A Computational Introduction to Number Theory and Algebra
(BETA version 1)

Victor Shoup

Preface

This book can be viewed in one of two ways: either as an introduction to number theory and algebra, with lots of algorithmic examples, or as an introduction to algorithmic number theory, but with all of the mathematical background material included. The book should be accessible to a very wide audience: the only formal mathematical prerequisites are those topics covered in a typical undergraduate Calculus course — all other mathematical material needed is developed in the text "from scratch."

The mathematical material covered includes the basics of number theory (including unique factorization, congruences, the distribution of primes, quadratic reciprocity), abstract algebra (including groups, rings, fields, and vector spaces), as well as discrete probability theory (which is needed for the treatment of probabilistic algorithms). The treatment of these topics is fairly standard, with perhaps the exception of groups: the text only deals with abelian groups, as this is all that is really needed for the purposes of this text, and the theory of abelian groups is much simpler and more transparent than for general groups. Even though it is mathematically fairly self contained, the text does presuppose that the reader is fairly proficient at reading and doing mathematical proofs — admittedly, this level of proficiency will typically only be attained by readers who have already had *some* exposure to *some* of the mathematical material covered here, but even such readers should find it convenient and useful to have all of the relevant mathematics conveniently available for study or review in one place. Since the mathematical concepts and notation are fairly standard, the reader who is already proficient in a certain area may safely skip, or quickly skim over, the relevant chapters or sections.

The computer science prerequisites for this text are fairly minimal: it is assumed that the reader is proficient in programming, and has had some exposure to the analysis of algorithms, essentially at the level of an undergraduate course on algorithms and data structures.

The choice of topics covered in this book was motivated primarily by their applicability to computing and communications, especially to the specific areas of cryptography and coding theory. The book may be useful, for example, for reference and self study to readers who want to learn about cryptography. The book could also be used, for example, as a textbook on a course on number theory and algebra, geared towards computer science students, either upper division undergraduates, or first year graduate students.

A few notes about the text:

- There are a few sections that are marked with a "♣," indicating that the material covered in that section is a bit technical, and is not needed in the sequel.
- In solving exercises, the reader is free to use any previously stated results in the text (including those in previous exercises).

Warning: This book is currently in BETA version. This means that it is essentially complete (except that it currently lacks an index), and should be fairly well polished; however, it has not yet been subjected to much independent reviewing or proof-reading. I would appreciate any feedback, especially feedback that identifies any errors or serious omissions. Please send you comments to `shoup@cs.nyu.edu`.

New York, August 2003 Victor Shoup

Contents

Chapter 1

Basic Properties of the Integers

This chapter reviews some of the basic properties of the integers, including notions of divisibility and primality, unique factorization into primes, greatest common divisors, and least common multiples.

1.1 Divisibility and Primality

Consider the integers $\mathbb{Z} = \{\ldots, -1, 0, 1, 2, \ldots\}$. For $a, b \in \mathbb{Z}$, we say that b **divides** a, and write $b \mid a$, if there exists $c \in \mathbb{Z}$ such that $a = bc$. If $b \mid a$, then b is called a **divisor** of a. If b does not divide a, then we write $b \nmid a$.

We first state some simple facts:

Theorem 1.1 *For all $a, b, c \in \mathbb{Z}$, we have*

1. *$a \mid a$, $1 \mid a$, and $a \mid 0$;*
2. *$0 \mid a$ if and only if $a = 0$;*
3. *$a \mid b$ and $b \mid c$ implies $a \mid c$;*
4. *$a \mid b$ implies $a \mid bc$;*
5. *$a \mid b$ and $a \mid c$ implies $a \mid (b + c)$;*
6. *$a \mid b$ and $b \mid a$ if and only if $a = \pm b$.*

Proof. Exercise. □

We say that an integer p is **prime** if $p > 1$ and the only divisors of p are ± 1 and $\pm p$. Conversely, and integer n is called **composite** if $n > 1$ and it is not prime. So an integer $n > 1$ is composite if and only if $n = ab$ for some integers a, b with $1 < a, b < n$.

A fundamental fact is that any integer can be written as a signed product of primes in an essentially unique way. More precisely:

Theorem 1.2 *Every non-zero integer n can be expressed as*

$$n = \pm \prod_p p^{\nu_p(n)},$$

where the product is over all primes, and all but a finite number of the exponents are zero. Moreover, the exponents and sign are uniquely determined by n.

To prove this theorem, we may clearly assume that n is positive, since otherwise, we may multiply n by -1 and reduce to the case where n is positive.

The proof of the existence part of Theorem 1.2 is easy. If n is 1 or prime, we are done; otherwise, there exist $a, b \in \mathbb{Z}$ with $1 < a, b < n$ and $n = ab$, and we apply an inductive argument with a and b.

The proof of the uniqueness part of Theorem 1.2 is not so simple, and most of the rest of this chapter is devoted to developing the ideas behind such a proof, along with a number of other very important tools. The essential ingredient in the proof is the following:

Theorem 1.3 (Division with Remainder Property) *For $a, b \in \mathbb{Z}$ with $b > 0$, there exist unique $q, r \in \mathbb{Z}$ such that $a = bq + r$ and $0 \leq r < b$.*

Proof. Consider the set S of non-negative integers of the form $a - zb$ with $z \in \mathbb{Z}$. This set is clearly non-empty, and so contains a minimum. Let $r = a - qb$ be the smallest integer in this set. By definition, we have $r \geq 0$. Also, we must have $r < b$, since otherwise, we would have $r - b \in S$, contradicting the minimality of r.

That proves the existence of r and q. For uniqueness, suppose that $a = bq + r$ and $a = bq' + r'$, where $0 \leq r, r' < b$. Then subtracting these two equations and rearranging terms, we obtain

$$r' - r = b(q - q'). \tag{1.1}$$

Now observe that by assumption, the left-hand side of (1.1) is less than b in absolute value. However, if $q \neq q'$, then the right-hand side of (1.1) would be at least b in absolute value; therefore, we must have $q = q'$. But then by (1.1), we must have $r = r'$. □

In the above theorem, it is easy to see that $q = \lfloor a/b \rfloor$, where for any real number x, $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. We shall write $r = a \text{ rem } b$. For $a \in \mathbb{Z}$ and a positive integer b, it is clear that $b \mid a$ if and only if $a \text{ rem } b = 0$.

Exercise 1.4 For integer n and real x, show that $n \leq x$ if and only if $n \leq \lfloor x \rfloor$. □

Exercise 1.5 For real x and positive integer n, show that $\lfloor \lfloor x \rfloor / n \rfloor = \lfloor x/n \rfloor$. In particular, for positive integers a, b, c, $\lfloor \lfloor a/b \rfloor / c \rfloor = \lfloor a/(bc) \rfloor$. □

Exercise 1.6 For real x, show that $2\lfloor x \rfloor \leq \lfloor 2x \rfloor \leq 2\lfloor x \rfloor + 1$. □

Exercise 1.7 For positive integers m and n, show that the number of multiples of m among $1, 2, \ldots, n$ is $\lfloor n/m \rfloor$. More generally, for integer $m \geq 1$ and real $x \geq 0$, show that the number of multiples of m in the interval $[1, x]$ is $\lfloor x/m \rfloor$. □

1.2 Ideals and Greatest Common Divisors

To carry on with the proof of Theorem 1.2, we introduce the notion of an **ideal** in $\mathbb{Z}$, which is a non-empty set of integers that is closed under addition and subtraction, and closed under multiplication by integers. That is, a non-empty set $I \subset \mathbb{Z}$ is an ideal if and only if for all $a, b \in I$ and all $z \in \mathbb{Z}$, we have

$$a + b \in I, \quad a - b \in I, \quad \text{and} \quad az \in I.$$

Note that in fact closure under addition and subtraction already implies closure under multiplication by integers, and so the definition is a bit redundant. However, we present the definition in this form, as it generalizes more nicely to other settings.

For $a_1, \ldots, a_k \in \mathbb{Z}$, define

$$a_1\mathbb{Z} + \cdots + a_k\mathbb{Z} := \{a_1z_1 + \cdots a_kz_k : z_1, \ldots, z_k \in \mathbb{Z}\}.$$

We leave it to the reader to verify that $a_1\mathbb{Z} + \cdots + a_k\mathbb{Z}$ is an ideal, and this ideal clearly contains $a_1, \ldots, a_k$. An ideal of the form $a\mathbb{Z}$ is called a **principal ideal**.

Example 1.8 Let $a = 3$ and consider the ideal $a\mathbb{Z}$. This consists of all integer multiples of 3; i.e., $a\mathbb{Z} = \{\ldots, -9, -6, -3, 0, 3, 6, 9, \ldots\}$. □

Example 1.9 Let $a_1 = 3$ and $a_2 = 5$, and consider the ideal $a_1\mathbb{Z} + a_2\mathbb{Z}$. This ideal contains $2a_1 - a_2 = 1$. Since it contains 1, it contains all integers; i.e., $a_1\mathbb{Z} + a_2\mathbb{Z} = \mathbb{Z}$. □

Theorem 1.10 *For any ideal $I \subset \mathbb{Z}$, there exists a unique non-negative integer d such that $I = d\mathbb{Z}$.*

Proof. We first prove the existence part of the theorem. If $I = \{0\}$, then $d = 0$ does the job, so let us assume that $I \neq \{0\}$. Since I contains non-zero integers, it must contain positive integers, since if $z \in I$ then so is $-z$. Let d be the smallest positive integer in I. We want to show that $I = d\mathbb{Z}$.

We first show that $I \subset d\mathbb{Z}$. To this end, let c be any element in I. It suffices to show that $d \mid c$. Using the Division with Remainder Property, write $c = qd + r$, where $0 \leq r < d$. Then by the closure properties of ideals, one sees that $r = c - qd$ is also an element of I, and by the minimality of the choice of d, we must have $r = 0$. Thus, $d \mid c$.

We next show that $d\mathbb{Z} \subset I$. This follows immediately from the fact that $d \in I$ and the closure properties of ideals.

That proves the existence part of the theorem. As for uniqueness, note that if $d\mathbb{Z} = d'\mathbb{Z}$, we have $d \mid d'$ and $d' \mid d$, from which it follows that $d' = \pm d$. □

For $a, b \in \mathbb{Z}$, we call $d \in \mathbb{Z}$ a **common divisor** of a and b if $d \mid a$ and $d \mid b$; moreover, we call such a d the **greatest common divisor** of a and b if d is non-negative and all other common divisors of a and b divide d. It is immediate from the definition of a greatest common divisor that it is unique if it exists at all.

Theorem 1.11 *For any $a, b \in \mathbb{Z}$, there exists a greatest common divisor d of a and b, and moreover, $a\mathbb{Z} + b\mathbb{Z} = d\mathbb{Z}$; in particular, $as + bt = d$ for some $s, t \in \mathbb{Z}$.*

Proof. We apply the previous theorem to the ideal $I = a\mathbb{Z} + b\mathbb{Z}$. Let $d \in \mathbb{Z}$ with $I = d\mathbb{Z}$, as in that theorem. Note that $a, b, d \in I$.

Since $a \in I = d\mathbb{Z}$, we see that $d \mid a$; similarly, $d \mid b$. So we see that d is a common divisor of a and b.

Since $d \in I = a\mathbb{Z} + b\mathbb{Z}$, there exist $s, t \in \mathbb{Z}$ such that $as + bt = d$. Now suppose $a = a'd'$ and $b = b'd'$ for $a', b', d' \in \mathbb{Z}$. Then the equation $as + bt = d$ implies that $d'(a's + b't) = d$, which says that $d' \mid d$. Thus, d is the greatest common divisor of a and b. □

For $a, b \in \mathbb{Z}$, we denote by $\gcd(a, b)$ the greatest common divisor of a and b. Note that as we have defined it, $\gcd(a, 0) = a$.

We say that a and b are **relatively prime** if $\gcd(a, b) = 1$. Notice that a and b are relatively prime if and only if $a\mathbb{Z} + b\mathbb{Z} = \mathbb{Z}$, i.e., if and only if there exist $s, t \in \mathbb{Z}$ such that $as + bt = 1$.

Theorem 1.12 *For $a, b, c \in \mathbb{Z}$ such that $c \mid ab$ and $\gcd(a, c) = 1$, we have $c \mid b$.*

Proof. Suppose that $c \mid ab$ and $\gcd(a, c) = 1$. Then since $\gcd(a, c) = 1$, by Theorem 1.11 we have $as + ct = 1$ for some $s, t \in \mathbb{Z}$. Multiplying this equation by b, we obtain

$$abs + cbt = b. \tag{1.2}$$

Since c divides ab by hypothesis, and since c clearly divides cbt, it follows that c divides the left-hand side of (1.2), and hence that c divides b. □

As a consequence of this theorem, we have:

Theorem 1.13 *Let p be prime, and let $a, b \in \mathbb{Z}$. Then $p \mid ab$ implies that $p \mid a$ or $p \mid b$.*

Proof. Assume that $p \mid ab$. The only divisors of p are ± 1 and $\pm p$. Thus, $\gcd(p, a)$ is either 1 or p. If $p \mid a$, we are done; otherwise, if $p \nmid a$, we must have $\gcd(p, a) = 1$, and by the previous theorem, we conclude that $p \mid b$. □

Theorem 1.13 is the key to proving the uniqueness part of Theorem 1.2. Indeed, suppose we have

$$p_1 \cdots p_r = p'_1 \cdots p'_s,$$

where the p_i and p'_i are primes (duplicates are allowed among the p_i and among the p'_i). If $r = 0$, we must have $s = 0$ and we are done. Otherwise, as p_1 divides the right-hand side, by inductively applying Theorem 1.13, one sees that p_1 is equal to some p'_i. We can cancel these terms and proceed inductively (on r). That proves the uniqueness part of Theorem 1.2.

Exercise 1.14 For two ideals $a\mathbb{Z}$ and $b\mathbb{Z}$, show that $a\mathbb{Z} \supset b\mathbb{Z}$ if and only if $a \mid b$, and that $a\mathbb{Z} = b\mathbb{Z}$ if and only if $a = \pm b$ □

Exercise 1.15 Let a, b, c be positive integers, with $\gcd(a, b) = 1$ and $c \geq ab$. Show that there exist *non-negative* integers s, t such that $c = as + bt$. □

Exercise 1.16 Let p be a prime and k an integer $0 < k < p$. Show that the binomial coefficient

$$\binom{p}{k} = \frac{p!}{k!(p-k)!},$$

which is an integer, of course, is divisible by p. □

1.3 More on Unique Factorization and Greatest Common Divisors

For non-zero integers a and b, it is easy to see that

$$\gcd(a, b) = \prod_p p^{\min(\nu_p(a), \nu_p(b))},$$

where the function $\nu_p(\cdot)$ is as implicitly defined in Theorem 1.2. If we make the notational conventions that $\nu_p(0) = \infty$ and $p^\infty = 0$ (see §A.4), then the above identity holds for all integers a and b.

For $a, b \in \mathbb{Z}$ a **common multiple** of a and b is an integer m such that $a \mid m$ and $b \mid m$; moreover, such an m is the **least common multiple** of a and b if m is non-negative and m divides all common multiples of a and b. In light of Theorem 1.2, it is clear that the least common multiple exists and is unique, and we denote the least common multiple of a and b as $\operatorname{lcm}(a, b)$. Note that as we have defined it, $\operatorname{lcm}(a, 0) = 0$. Also, for all integers a and b, we have (using same notational conventions as above)

$$\operatorname{lcm}(a, b) = \prod_p p^{\max(\nu_p(a), \nu_p(b))}.$$

Moreover, for all $a, b \in \mathbb{Z}$, we have

$$\gcd(a, b) \cdot \operatorname{lcm}(a, b) = ab.$$

It is easy to generalize the notions of greatest common divisor and least common multiple from two integers to many integers. For $a_1, \ldots, a_k \in \mathbb{Z}$, with $k \geq 1$, we call $d \in \mathbb{Z}$ a common divisor of $a_1, \ldots, a_k$ if $d \mid a_i$ for $1 \leq i \leq k$; moreover, we call such a d the greatest common divisor of $a_1, \ldots, a_k$ if d is non-negative and all other common divisors of $a_1, \ldots, a_k$ divide d. It is clear that the greatest common divisor of $a_1, \ldots, a_k$ exists and is unique and is given by the formula

$$\gcd(a_1, \ldots, a_k) = \prod_p p^{\min_i(\nu_p(a_i))}.$$

Analogously, for $a_1, \ldots, a_k \in \mathbb{Z}$, with $k \geq 1$, we call $m \in \mathbb{Z}$ a common multiple of $a_1, \ldots, a_k$ if $a_i \mid m$ for $1 \leq i \leq k$; moreover, such an m is called the least common multiple of $a_1, \ldots, a_k$ if m divides all common multiples of $a_1, \ldots, a_k$. It is clear that the least common multiple of $a_1, \ldots, a_k$ exists and is unique and is given by the formula

$$\operatorname{lcm}(a_1, \ldots, a_k) = \prod_p p^{\max_i(\nu_p(a_i))}.$$

Exercise 1.17 For $a_1, \ldots, a_k \in \mathbb{Z}$, with $k > 2$, show that

$$\gcd(a_1, \ldots, a_k) := \gcd(\gcd(a_1, \ldots, a_{k-1}), a_k)$$

and

$$\operatorname{lcm}(a_1, \ldots, a_k) := \operatorname{lcm}(\operatorname{lcm}(a_1, \ldots, a_{k-1}), a_k).$$

□

Exercise 1.18 Show that for any $a_1, \ldots, a_k \in \mathbb{Z}$, if $d = \gcd(a_1, \ldots, a_k)$, then $d\mathbb{Z} = a_1\mathbb{Z} + \cdots + a_k\mathbb{Z}$; in particular, there exist integers $s_1, \ldots, s_k$ such that

$$d = a_1 s_1 + \cdots + a_k s_k.$$

□

Because of the unique factorization property, given any rational number a/b, with $a, b \in \mathbb{Z}$ and $b \neq 0$, if we set $d := \gcd(a, b)$, and define the integers $a' := a/d$ and $b' := b/d$, then we have $a/b = a'/b'$ and $\gcd(a', b') = 1$. Moreover, if $\tilde{a}/\tilde{b} = a'/b'$, then we have $\tilde{a}b' = a'\tilde{b}$, and so $b' \mid a'\tilde{b}$, and since $\gcd(a', b') = 1$, we see that $b' \mid \tilde{b}$; if $\tilde{b} = \tilde{d}b'$, it follows that $\tilde{a} = \tilde{d}a'$. Thus, we can represent every rational number as a fraction in "lowest terms," and this representation is unique up to sign.

Exercise 1.19 For a prime p and a non-zero rational number $x = a/b$, let us define $\nu_p(x) := \nu_p(a) - \nu_p(b)$. As above, we define $\nu_p(0) := \infty$ (see §A.4).

(a) Show that this definition of $\nu_p(x)$ is unambiguous, in the sense that it does not depend on the particular choice of a and b.

(b) Show that for all rational numbers x, y, we have $\nu_p(xy) = \nu_p(x) + \nu_p(y)$.

(c) Show that for all rational numbers x, y, we have $\nu_p(x + y) \geq \min\{\nu_p(x), \nu_p(y)\}$, and that if $\nu_p(x) < \nu_p(y)$, then $\nu_p(x + y) = \nu_p(x)$.

□

Exercise 1.20 Let n be a positive integer, and let C_n denote the number of pairs of integers (a, b) such that $1 \leq a, b \leq n$ and $\gcd(a, b) = 1$, and let F_n be the number of *distinct* rational numbers a/b, where $0 \leq a < b \leq n$. Show (a) that $F_n = (C_n+1)/2$, and (b) that $C_n \geq n^2/4$.
□

Exercise 1.21 Show that if an integer cannot be expressed as a square of an integer, then it cannot be expressed as a square of any rational number. □

Chapter 2

Congruences

This chapter discusses the notion of congruences.

2.1 Definitions and Basic Properties

For positive integer n and for $a, b \in \mathbb{Z}$, we say that a **is congruent to** b **modulo** n if $n \mid (a - b)$, and we write $a \equiv b \pmod{n}$. If $n \nmid (a - b)$, then we write $a \not\equiv b \pmod{n}$. The number n appearing in such congruences is called the **modulus** of the congruence.

A trivial observation is that $a \equiv b \pmod{n}$ if and only if there exists an integer c such that $a = b + cn$. Another trivial observation is that if $a \equiv b \pmod{n}$ and $n' \mid n$, then $a \equiv b \pmod{n'}$.

A key property of congruences is that they are "compatible" with integer addition and multiplication, in the following sense:

Theorem 2.1 *For all positive integers n, and all $a, a', b, b' \in \mathbb{Z}$, if $a \equiv a' \pmod{n}$ and $b \equiv b' \pmod{n}$, then*

$$a + b \equiv a' + b' \pmod{n}$$

and

$$a \cdot b \equiv a' \cdot b' \pmod{n}.$$

Proof. Suppose that $a \equiv a' \pmod{n}$ and $b \equiv b' \pmod{n}$. This means that there exist integers c and d such that $a' = a + cn$ and $b' = b + dn$. Therefore,

$$a' + b' = a + b + (c + d)n,$$

which proves the first equality of the theorem, and

$$a'b' = (a + cn)(b + dn) = ab + (ad + bc + cdn)n,$$

which proves the second equality. □

2.2 Solving Linear Congruences

For a positive integer n, and $a \in \mathbb{Z}$, we say that a is a **unit modulo** n if there exists $a' \in \mathbb{Z}$ such that $aa' \equiv 1 \pmod{n}$, in which case we say that a' is a **multiplicative inverse of** a **modulo** n.

Theorem 2.2 *An integer a is a unit modulo n if and only if a and n are relatively prime.*

Proof. This follows immediately from the fact that a and n are relatively prime if and only if there exist $s, t \in \mathbb{Z}$ such that $as + bt = 1$. □

We now prove a simple a "cancellation law" for congruences:

Theorem 2.3 *If a is relatively prime to n, then $az \equiv az' \pmod{n}$ if and only if $z \equiv z' \pmod{n}$. More generally, if $d = \gcd(a, n)$, then $az \equiv az' \pmod{n}$ if and only if $z \equiv z' \pmod{n/d}$.*

Proof. For the first statement, assume that $\gcd(a, n) = 1$, and let a' be a multiplicative inverse of a modulo n. Then, $az \equiv az' \pmod{n}$ implies $a'az \equiv a'az' \pmod{n}$, which implies $z \equiv z' \pmod{n}$, since $a'a \equiv 1 \pmod{n}$. Conversely, if $z \equiv z' \pmod{n}$, then trivially $az \equiv az' \pmod{n}$. That proves the first statement.

For the second statement, let $d = \gcd(a, n)$. Simply from the definition of congruences, one sees that in general, $az \equiv az' \pmod{n}$ holds if and only if $(a/d)z \equiv (a/d)z' \pmod{n/d}$. Moreover, since a/d and n/d are relatively prime, the first statement of the theorem implies that $(a/d)z \equiv (a/d)z' \pmod{n}$ holds if and only if $z \equiv z' \pmod{n/d}$. That proves the second statement. □

One consequence of the above theorem is that multiplicative inverses, if they exist, are uniquely determined modulo n.

We next look at solutions z to congruences of the form $az \equiv b \pmod{n}$, for given integers n, a, b.

Theorem 2.4 *Let n be a positive integer and let $a, b \in \mathbb{Z}$. If a is relatively prime to n, then the congruence $az \equiv b \pmod{n}$ has a solution z; moreover, any integer z' is a solution if and only if $z \equiv z' \pmod{n}$.*

Proof. The integer $z = ba'$, where a' is a multiplicative inverse of a modulo n, is clearly a solution. For any integer z', we have $az' \equiv b \pmod{n}$ if and only if $az' \equiv az \pmod{n}$, which by Theorem 2.3 holds if and only if $z \equiv z' \pmod{n}$. □

In particular, this theorem implies that multiplicative inverses are uniquely determined modulo n.

More generally, we have:

Theorem 2.5 *Let n be a positive integer and let $a, b \in \mathbb{Z}$. Let $d = \gcd(a, n)$. If $d \mid b$, then the congruence $az \equiv b \pmod{n}$ has a solution z, and any integer z' is also a solution if and only if $z \equiv z' \pmod{n/d}$. If $d \nmid b$, then the congruence $az \equiv b \pmod{n}$ has no solution z.*

Proof. Let n, a, b, d be as defined above.

For the first statement, suppose that $d \mid b$. In this case, by Theorem 2.3, we have $az \equiv b \pmod{n}$ if and only if $(a/d)z \equiv (b/d) \pmod{n/d}$, and so the statement follows immediately from Theorem 2.4.

For the second statement, assume that $az \equiv b \pmod{n}$ for some integer z. Then since $d \mid n$, we have $az \equiv b \pmod{d}$. However, $az \equiv 0 \pmod{d}$, since $d \mid a$, and hence $b \equiv 0 \pmod{d}$, i.e., $d \mid b$. □

Example 2.6 The following table illustrates what the above theorem says for $n = 15$ and $a = 1, 2, 3, 4, 5, 6$.

z	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$2z$ rem 15	0	2	4	6	8	10	12	14	1	3	5	7	9	11	13
$3z$ rem 15	0	3	6	9	12	0	3	6	9	12	0	3	6	9	12
$4z$ rem 15	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11
$5z$ rem 15	0	5	10	0	5	10	0	5	10	0	5	10	0	5	10
$6z$ rem 15	0	6	12	3	9	0	6	12	3	9	0	6	12	3	9

In the second row, we are looking at the values $2z$ rem 15, and we see that this row is just a permutation of the first row. So for every b, there exists an z such that $2z \equiv b \pmod{15}$. We could have inferred this fact from the theorem, since $\gcd(2, 15) = 1$.

In the third row, the only numbers hit are the multiples of 3, which follows from the fact that $\gcd(3, 15) = 3$. Also note that the pattern in this row repeats every five columns; that is also implied by the theorem; i.e., $3z \equiv 3z' \pmod{15}$ if and only if $z \equiv z' \pmod{5}$.

In the fourth row, we again see a permutation of the first row, which follows from the fact that $\gcd(4, 15) = 1$.

In the fifth row, the only numbers hit are the multiples of 5, which follows from the fact that $\gcd(5, 15) = 5$. Also note that the pattern in this row repeats every three columns; that is also implied by the theorem; i.e., $5z \equiv 5z' \pmod{15}$ if and only if $z \equiv z' \pmod{3}$.

In the sixth row, since $\gcd(6, 15) = 3$, we see a permutation of the third row. The pattern repeats after five columns, although the pattern is a permutation of the pattern in the third row. □

Next, we consider systems of congruences with respect to moduli that that are relatively prime in pairs. The result we state here is known as the Chinese Remainder Theorem, and is extremely useful in a number of contexts.

Theorem 2.7 (Chinese Remainder Theorem) *Let $k > 0$, and let $a_1, \ldots, a_k \in \mathbb{Z}$, and let $n_1, \ldots, n_k$ be positive integers such that $\gcd(n_i, n_j) = 1$ for all $1 \leq i < j \leq k$. Then there exists an integer z such that*

$$z \equiv a_i \pmod{n_i} \quad (i = 1, \ldots, k).$$

Moreover, any other integer z' is also a solution of these congruences if and only if $z \equiv z' \pmod{n}$, where $n := \prod_{i=1}^{k} n_i$.

Proof. Let $n := \prod_{i=1}^{k} n_i$, as in the statement of the theorem. Let us also define

$$n_i' := n/n_i \quad (i = 1, \ldots, k).$$

It is clear that $\gcd(n_i, n_i') = 1$ for $1 \le i \le k$, and so let m_i be a multiplicative inverse of n_i' modulo n_i for $1 \le i \le k$, and define

$$w_i := n_i' m_i \quad (i = 1, \ldots, k).$$

By construction, one sees that for $1 \le i \le k$, we have

$$w_i \equiv 1 \pmod{n_i}$$

and

$$w_i \equiv 0 \pmod{n_j} \quad \text{for } 1 \le j \le k \text{ with } j \ne i.$$

That is to say, for $1 \le i, j \le k$, $w_i \equiv \delta_{ij} \pmod{n_j}$, where $\delta_{ij} := 1$ for $i = j$ and $\delta_{ij} := 0$ for $i \ne j$.

Now define

$$z := \sum_{i=1}^{k} w_i a_i.$$

One then sees that for $1 \le j \le k$,

$$z \equiv \sum_{i=1}^{k} w_i a_i \equiv \sum_{i=1}^{k} \delta_{ij} a_i \equiv a_j \pmod{n_j}.$$

Therefore, this z solves the given system of congruences.

Moreover, if $z' \equiv z \pmod{n}$, then since $n_i \mid n$ for $1 \le i \le k$, we see that $z' \equiv z \equiv a_i \pmod{n_i}$ for $1 \le i \le k$, and so z' also solves the system of congruences.

Finally, if z' solves the system of congruences, then $z' \equiv z \pmod{n_i}$ for $1 \le i \le k$. That is, $n_i \mid (z' - z)$ for $1 \le i \le k$. Since $\gcd(n_i, n_j) = 1$ for $i \ne j$, this implies that $n \mid (z' - z)$, i.e., $z' \equiv z \pmod{n}$. □

Example 2.8 The following table illustrates what the above theorem says for $n_1 = 3$ and $n_2 = 5$.

z	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
z rem 3	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
z rem 5	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4

We see that as z ranges from 0 to 15, the pairs $(z \text{ rem } 3, z \text{ rem } 5)$ range over all pairs (a_1, a_2) with $0 \le a_1 < 3$ and $0 \le a_2 < 5$, with every pair being hit exactly once. □

Exercise 2.9 Find an integer z such that $z \equiv -1 \pmod{100}$, $z \equiv 1 \pmod{33}$, and $z \equiv 2 \pmod{7}$. □

2.3 Residue Classes

It is easy to see that for a fixed value of n, the relation $\cdot \equiv \cdot \pmod{n}$ is an *equivalence relation* on the set $\mathbb{Z}$ (see §A.5). As such, this relation partitions the set $\mathbb{Z}$ into equivalence classes. We denote the equivalence class containing the integer a by $[a \bmod n]$, or when n is clear from context, we may simply write $[a]$. Historically, these equivalence classes are called **residue classes modulo** n, and we shall adopt this terminology here as well.

It is easy to see from the definitions that

$$[a \bmod n] = a + n\mathbb{Z} := \{a + nz : z \in \mathbb{Z}\}.$$

Note that a given residue class modulo n has many different "names"; e.g., the residue class $[1]$ is the same as the residue class $[1 + n]$. For any integer a in a residue class, we call a a **representative** of that class.

Theorem 2.10 *For a positive integer n, there are precisely n distinct residue classes modulo n, namely, $[a]$ for $0 \leq a < n$. Moreover, for any $k \in \mathbb{Z}$, the residue classes $[k + a]$ for $0 \leq a < n$ are distinct and therefore include all residue classes modulo n.*

Proof. Exercise. □

Fix a positive integer n. Let us define $\mathbb{Z}_n$ as the set of residue classes modulo n. We can "equip" $\mathbb{Z}_n$ with binary operators defining addition and multiplication in a natural way as follows: for $a, b \in \mathbb{Z}$, we define

$$[a] + [b] := [a + b],$$

and we define

$$[a] \cdot [b] := [a \cdot b].$$

Of course, one has to check this definition is unambiguous, i.e., that the addition and multiplication operators are well defined, in the sense that the sum or product of two residue classes does not depend on which particular representatives of the classes are chosen in the above definitions. More precisely, one must check that if $[a] = [a']$ and $[b] = [b']$, then $[a \text{ op } b] = [a' \text{ op } b']$, for $\text{op} \in \{+, \cdot\}$. However, this property follows immediately from Theorem 2.1.

These definitions of addition and multiplication operators on $\mathbb{Z}_n$ yield a very natural algebraic structure whose salient properties are as follows:

Theorem 2.11 *Let n be a positive integer, and consider the set $\mathbb{Z}_n$ of residue classes modulo n with addition and multiplication of residue classes as defined above.*

For all $a, b, c \in \mathbb{Z}$, we have

1. $[a] + [b] = [b] + [a]$ *(addition is commutative),*
2. $([a] + [b]) + [c] = [a] + ([b] + [c])$ *(addition is associative),*

3. $[a] + [0] = [a]$ *(existence of additive identity),*

4. $[a] + [-a] = [0]$ *(existence of additive inverses),*

5. $[a] \cdot [b] = [b] \cdot [a]$ *(multiplication is commutative),*

6. $([a] \cdot [b]) \cdot [c] = [a] \cdot ([b] \cdot [c])$ *(multiplication is associative),*

7. $[a] \cdot ([b] + [c]) = [a] \cdot [b] + [a] \cdot [c]$ *(multiplication distributes over addition)*

8. $[a] \cdot [1] = [a]$ *(existence of multiplicative identity).*

Proof. All of these properties follow trivially from the corresponding properties for the integers, together with the definition of addition and multiplication of residue classes. □

An algebraic structure satisfying the conditions in the above theorem is known more generally as a "commutative ring with unity," a notion that we will discuss in §9.

Note that while all elements of $\mathbb{Z}_n$ have an additive inverses, not all elements of $\mathbb{Z}_n$ have a multiplicative inverse; indeed, by Theorem 2.2, $[a \bmod n]$ has a multiplicative inverse if and only if $\gcd(a, n) = 1$, in which case, by Theorem 2.3, the inverse is unique. One denotes by $\mathbb{Z}_n^*$ the set of all residue classes $[a]$ of $\mathbb{Z}_n$ that have a multiplicative inverse; it is easy to see that $\mathbb{Z}_n^*$ is closed under multiplication.

2.4 Euler's ϕ-Function

Euler's ϕ-function is defined for positive integers n as the number of elements of $\mathbb{Z}_n^*$. Equivalently, $\phi(n)$ is equal to the number of integers between 0 and $n-1$ that are relatively prime to n. For example, $\phi(1) = 1$, $\phi(2) = 1$, $\phi(3) = 2$, and $\phi(4) = 2$.

A fact that is sometimes useful is the following:

Theorem 2.12 *For any positive integer n, we have*

$$\sum_{d|n} \phi(d) = n,$$

where the sum is over all positive divisors d of n.

Proof. Consider the list of n rational numbers $0/n, 1/n, \ldots, (n-1)/n$. For any divisor d of n and for any integer a with $0 \leq a < d$ and $\gcd(a, d) = 1$, the fraction a/d appears in the list exactly once, and moreover, every number in the sequence, when expressed as a fraction in lowest terms, is of this form. □

Using the Chinese Remainder Theorem, it is easy to get a nice formula for $\phi(n)$ in terms for the prime factorization of n.

Theorem 2.13 *For positive integers n, m with $\gcd(n, m) = 1$, we have*

$$\phi(nm) = \phi(n)\phi(m).$$

Proof. Consider the map

$$\begin{array}{rccc} \rho & : & \mathbb{Z}_{nm} & \to & \mathbb{Z}_n \times \mathbb{Z}_m \\ & & [a \bmod nm] & \mapsto & ([a \bmod n], [a \bmod m]). \end{array}$$

First, note that the definition of ρ is unambiguous, since $a \equiv a' \pmod{nm}$ implies $a \equiv a' \pmod{n}$ and $a \equiv a' \pmod{m}$. Second, according to the Chinese Remainder Theorem, the map ρ is is one-to-one and onto. Moreover, it is easy to see that $\gcd(a, nm) = 1$ if and only if $\gcd(a, n) = 1$ and $\gcd(a, m) = 1$ (verify). Therefore, the map ρ carries $\mathbb{Z}^*_{nm}$ injectively onto $\mathbb{Z}^*_n \times \mathbb{Z}^*_m$. □

Theorem 2.14 *For a prime p and a positive integer e, $\phi(p^e) = p^{e-1}(p - 1)$.*

Proof. The multiples of p among $0, 1, \ldots, p^e - 1$ are

$$0 \cdot p, 1 \cdot p, \ldots, (p^{e-1} - 1) \cdot p,$$

of which there are precisely p^{e-1}. Thus, $\phi(p^e) = p^e - p^{e-1} = p^{e-1}(p - 1)$. □

As an immediate consequence of the above two theorems, we have:

Theorem 2.15 *If $n = p_1^{e_1} \cdots p_r^{e_r}$ is the factorization of n into primes, then*

$$\phi(n) = p_1^{e_1-1}(p_1 - 1) \cdots p_r^{e_r-1}(p_r - 1).$$

The ϕ function is an example of a **multiplicative** function: a function f from the positive integers into the reals such that for all positive integers n, m with $\gcd(n, m) = 1$, we have $f(nm) = f(n)f(m)$.

Exercise 2.16 Show that if f is a multiplicative function, and if $n = p_1^{e_1} \cdots p_r^{e_r}$ is the prime factorization of n, then $f(n) = f(p_1^{e_1}) \cdots f(p_r^{e_r})$. □

Exercise 2.17 Let f be a polynomial with integer coefficients, and for positive integer n define $\omega_f(n)$ to be the number if integers $z \in \{0, \ldots, n - 1\}$ such that $f(z) \equiv 0 \pmod{n}$. Show that ω_f is multiplicative. □

Exercise 2.18 Show that $\phi(nm) = \phi(\gcd(n, m))\phi(\mathrm{lcm}(n, m))$. □

2.5 Other Arithmetic Functions

Let f and g be real-valued functions defined on the positive integers. The **Dirichlet product** of f and g, denoted $f \star g$, is defined by the formula $(f \star g)(n) := \sum f(d_1)g(d_2)$, the sum being over all pairs (d_1, d_2) of positive integers with $d_1 d_2 = n$. The product is clearly commutative (i.e., $f \star g = g \star f$), and is associate as well, which one can see by checking that

$$(f \star (g \star h))(n) = ((f \star g) \star h)(n) = \sum f(d_1)g(d_2)h(d_3),$$

the sum being over all triples (d_1, d_2, d_3) of positive integers with $d_1 d_2 d_3 = n$.

We now introduce three special functions: I, J, and μ. The function $I(n)$ is defined to be 1 when $n = 1$ and 0 when $n > 1$. The function $J(n)$ is defined to be 1 for all n. The **Möbius function** μ is defined for positive integers n as follows:

$$\mu(n) := \begin{cases} 0 & \text{if } n \text{ is divisible by a square other than 1;} \\ (-1)^r & \text{if } n \text{ is the product of } r \text{ distinct primes.} \end{cases}$$

It is easy to see (verify) that for any function f, $f \star I = f$, and that $(f \star J)(n) = \sum_{d|n} f(d)$. Also, the functions I, J, and μ are multiplicative (verify). A useful property of the Möbius function is the following:

Theorem 2.19 *For any multiplicative function f, if $n = p_1^{e_1} \cdots p_r^{e_r}$ is the prime factorization of n, we have*

$$\sum_{d|n} \mu(d) f(d) = (1 - f(p_1)) \cdots (1 - f(p_r)). \tag{2.1}$$

Proof. The non-zero terms in the sum on the left-hand side of (2.1) are those corresponding to divisors d of the form $p_{i_1} \cdots p_{i_\ell}$, where $p_{i_1}, \ldots, p_{i\ell}$ are distinct; the value contributed to the sum by such a term is $(-1)^\ell f(p_{i_1} \cdots p_{i_\ell}) = (-1)^\ell f(p_{i_1}) \cdots f(p_{i_\ell})$. These are the same as the terms in the expansion of the product on the right-hand side of (2.1). □

For example, suppose $f(d) = 1/d$ in the above theorem, and let $n = p_1^{e_1} \cdots p_r^{e_r}$ be the prime factorization of n. Then we obtain:

$$\sum_{d|n} \mu(d)/d = (1 - 1/p_1) \cdots (1 - 1/p_r). \tag{2.2}$$

As another example, suppose $f = J$. Then we obtain

$$\mu \star J = \sum_{d|n} \mu(d) = \prod_{i=1}^{r} (1 - 1),$$

which is 1 if $n = 1$, and is zero if $n > 1$. Thus, we have

$$\mu \star J = I. \tag{2.3}$$

Theorem 2.20 (Möbius Inversion Formula) *Let f and F be real-valued functions on the positive integers such that $F = J \star f$, i.e., $F(n) = \sum_{d|n} f(d)$. Then $f = \mu \star F$, i.e., $f(n) = \sum_{d|n} \mu(d)F(n/d)$.*

Proof. We have $F = f \star J$. Thus, using the associativity property of the Dirichlet product, along with (2.3), we have

$$F \star \mu = (f \star J) \star \mu = f \star (J \star \mu) = f \star I = f,$$

which proves the statement. □

As an application of the Möbius inversion formula, we can get a different proof of Theorem 2.15, based on Theorem 2.12. From the latter theorem, we have $\sum_{d|n} \phi(n) = n$. Applying Möbius inversion to this, with $F(n) = n$ and $f(n) = \phi(n)$, and using (2.2), we obtain

$$\begin{aligned} \phi(n) &= \sum_{d|n} \mu(d)n/d = n \sum_{d|n} \mu(d)/d \\ &= n(1 - 1/p_1) \cdots (1 - 1/p_r) = p_1^{e_1 - 1}(p_1 - 1) \cdots p_r^{e_r - 1}(p_r - 1). \end{aligned}$$

Exercise 2.21 Show that if f and g are multiplicative, then so is $f \star g$. □

Exercise 2.22 Show that if f is multiplicative, and if $n = p_1^{e_1} \cdots p_r^{e_r}$ is the prime factorization of n, then

$$\sum_{d|n} (\mu(d))^2 f(d) = (1 + f(p_1)) \cdots (1 + f(p_r)).$$

□

Exercise 2.23 Show that n is not divisible by a square other than 1 if and only if $\sum_{d|n} (\mu(d))^2 \phi(d) = n$. □

Exercise 2.24 Define $d(n)$ to be the number of divisors of n. Show that d is a multiplicative function, and moreover, that if $n = p_1^{e_1} \cdots p_r^{e_r}$ is the prime factorization of n, then

$$d(n) = (e_1 + 1) \cdots (e_r + 1).$$

□

Exercise 2.25 For $k \geq 1$, define $\sigma_k(n) := \sum_{d|n} d^k$. Show that σ_k is a multiplicative function, and moreover, that if $n = p_1^{e_1} \cdots p_r^{e_r}$ is the prime factorization of n, then

$$\sigma_k(n) = \prod_{i=1}^{r} \frac{p_i^{k(e_i+1)} - 1}{p_i^k - 1}.$$

□

Chapter 3

Computing with Large Integers

In this chapter, we review standard asymptotic notation, introduce the formal computational model we shall use throughout the rest of the text, and discuss basic algorithms for computing with large integers.

3.1 Asymptotic Notation

We review some standard notation for relating the rate of growth of functions.

Suppose that x is a variable taking positive integer or real values, and let g denote a real-valued function that is positive for all sufficiently large x; also, let f denote any real-valued function in x. Then

- $f = O(g)$ means that $|f(x)| \leq cg(x)$ for some positive constant c and all sufficiently large x,
- $f = \Omega(g)$ means that $f(x) \geq cg(x)$ for some positive constant c and all sufficiently large x,
- $f = \Theta(g)$ means that $cg(x) \leq f(x) \leq dg(x)$, for some positive constants c and d and all sufficiently large x,
- $f = o(g)$ means that $f(x)/g(x) \to 0$ as $x \to \infty$, and
- $f \sim g$ means that $f/g \to 1$ as $x \to \infty$, or equivalently, $f(x) = g(x)(1+\epsilon(x))$ for where $\epsilon(x) \to 0$ as $x \to \infty$.

One also may write $O(g)$ in an expression to denote an anonymous function f such that $f = O(g)$, e.g., $\sum_{i=1}^{n} i = n^2/2 + O(n)$. Similarly for $\Omega(g)$, $\Theta(g)$, and $o(g)$. The expression $O(1)$ denotes a function bounded in absolute value by a constant, while the expression $o(1)$ denotes a function that tends to zero in the limit.

One may also use the same notation in a setting where x is a real variable tending to some finite limit x_0, in which case, the phrases "for all sufficiently large x" and "as $x \to \infty$" are replaced by "for all x sufficiently close to x_0" and "as $x \to x_0$."

As an even further use (abuse?) of the notation, one may use the "O," "Ω," and "Θ" notation for functions on an arbitrary domain, in which case the relevant bound should hold throughout the entire domain.

Exercise 3.1 Let x be a variable tending to ∞. Order the following functions in x so that for each adjacent pair f, g in the ordering, we have $f = O(g)$, and indicate if $f = o(g)$, $f \sim g$, or $g = O(f)$:

$$x^3,\ e^x x^2,\ 1/x,\ \ x^2(x+100)+1/x,\ x+\sqrt{x},\ \log x,\ 2x^2,\ x,$$

$$e^{-x},\ 2x^2-10x+4,\ e^{x+\sqrt{x}},\ e^x,\ x^{-2},\ x^2(\log x)^{1000}.$$

□

Exercise 3.2 Repeat the previous exercise, but with x a real variable that tends to 0. □

Exercise 3.3 Give an example of two non-decreasing, functions f and g, both mapping positive integers to positive integers, such that $f \neq O(g)$ and $g \neq O(f)$. □

Exercise 3.4 Show that

(a) the relation "$\sim$" is an equivalence relation (see §A.5);

(b) $f_1 \sim f_2$ and $g_1 \sim g_2$ implies $f_1 \star g_1 \sim f_2 \star g_2$, where "$\star$" denotes addition, multiplication, or division;

(c) If $g \to \infty$, then $f_1 \sim f_2$ implies $f_1 \circ g \sim f_2 \circ g$, where "$\circ$" denotes function composition.

□

Exercise 3.5 Show that all of the claims in the previous exercise also hold when the relation "$\sim$" is replaced with the relation "$\cdot = \Theta(\cdot)$." □

Exercise 3.6 Show that if $f_1 \sim f_2$, then $\log(f_1) = \log(f_2) + o(1)$, and in particular, if $f_1 = \Omega(1)$, then $\log(f_1) \sim \log(f_2)$. □

Exercise 3.7 Suppose that $f(i)$ and $g(i)$ are functions defined on the integers $k, k+1, \ldots$, and that $g(i)$ takes positive values for all sufficiently large i. For $n \geq k$, define $F(n) := \sum_{i=k}^{n} f(i)$ and $G(n) := \sum_{i=k}^{n} g(i)$. Show that if $f = O(g)$ and $G(n) > 0$ for all sufficiently large n, then $F = O(G)$. □

Exercise 3.8 Suppose that $f(i)$ and $g(i)$ are functions defined on the integers $k, k+1, \ldots$, both of which take positive values for all sufficiently large i. For $n \geq k$, define $F(n) := \sum_{i=k}^{n} f(i)$ and $G(n) := \sum_{i=k}^{n} g(i)$. Show that if $f \sim g$ and $G(n) \to \infty$ as $n \to \infty$, then $F \sim G$. □

The following two exercises are continuous variants of the previous two exercises. To avoid unnecessary distractions, we shall only consider functions that are quite "well behaved." In particular, we restrict ourselves to piece-wise continuous functions (see §A.8).

Exercise 3.9 Suppose that $f(t)$ and $g(t)$ are piece-wise continuous on $[a, \infty)$, and that $g(t)$ takes positive values for all sufficiently large t. For $x \geq a$, define $F(x) := \int_a^x f(t)dt$ and $G(x) := \int_a^x g(t)dt$. Show that if $f = O(g)$ and $G(x) > 0$ for all sufficiently large x, then $F = O(G)$. □

Exercise 3.10 Suppose that $f(t)$ and $g(t)$ are piece-wise continuous $[a, \infty)$, both of which take positive values for all sufficiently large t. For $x \geq a$, define $F(x) := \int_a^x f(t)dt$ and $G(x) := \int_a^x g(t)dt$. Show that if $f \sim g$ and $G(x) \to \infty$ as $x \to \infty$, then $F \sim G$. □

3.2 Machine Models and Complexity Theory

When presenting an algorithm, we shall always use a high-level, and somewhat informal, notation. However, all of our high-level descriptions can be routinely translated into the machine-language of an actual computer. So that our theorems on the running times of algorithms have a precise mathematical meaning, we formally define an "idealized" computer: the **Random Access Machine** or **RAM**.

A RAM consists of an unbounded sequence of **memory cells**

$$m[0], m[1], m[2], \ldots$$

each of which can store an arbitrary integer, together with a **program**. A program consists of a finite sequence of instructions $I_0, I_1, \ldots$, where each instruction is of one of the following types:

arithmetic This type of instruction is of the form $\alpha \leftarrow \beta \circ \gamma$, where $\circ$ represents one of the operations addition, subtraction, multiplication, or integer division. The values β and γ are of the form c, $m[a]$, or $m[m[a]]$, and α is of the form $m[a]$ or $m[m[a]]$, where c is an integer constant and a is a nonnegative integer constant. Execution of this type of instruction causes the value $\beta \circ \gamma$ to be evaluated and then stored in α.

branching This type of instruction is of the form IF $\beta \sim \gamma$ GOTO i, where i is the index of an instruction, and where $\sim$ is one of the comparison operators $=, \neq, <, >, \leq, \geq$, and β and γ are as above. Execution of this type of instruction causes the "flow of control" to pass conditionally to instruction I_i.

halt The HALT instruction halts the execution of the program.

A RAM executes by executing instruction I_0, and continues to execute instructions, following branching instructions as appropriate, until a HALT instruction is executed.

We do not specify input or output instructions, and instead assume that the input and output are to be found in memory at some prescribed location, in some standardized format.

To determine the running time of a program on a given input, we charge 1 unit of time to each instruction executed.

This model of computation closely resembles a typical modern-day computer, except that we have abstracted away many annoying details. However, there are two details of real machines that cannot be ignored; namely, any real machine has a finite number of memory cells, and each cell can store numbers only in some fixed range.

The first limitation must be dealt with by either purchasing sufficient memory or designing more space-efficient algorithms.

The second limitation is especially annoying, as we will want to perform computations with quite large integers—much larger than will fit into any single memory cell of an actual machine. To deal with this limitation, we shall represent such large integers as vectors of digits to some base, so that each digit is bounded so as to fit into a memory cell. This is discussed in more detail in the next section. Using this strategy, the only other numbers we actually need to store in memory cells are "small" numbers representing array indices, addresses, and the like, which hopefully will fit into the memory cells of actual machines.

Thus, whenever we speak of an algorithm, we shall mean an algorithm that can be implemented on a RAM, such that all numbers stored in memory cells are "small" numbers, as discussed above. Admittedly, this is a bit imprecise. For the reader who demands more precision, we can make a restriction, such as the following: after the execution of m steps, all numbers stored in memory cells are bounded by $m^c + d$ in absolute value, for constants c and d — in making this formal requirement, we assume that the value m includes the number of memory cells of the input.

Even with these caveats and restrictions, the running time as we have defined it for a RAM is still only a rough predictor of performance on an actual machine. On a real machine, different instructions may take significantly different amounts of time to execute; for example, a division instruction may take much longer than an addition instruction. Also, on a real machine, the behavior of the cache may significantly affect the time it takes to load or store the operands of an instruction. However, despite all of these problems, it still turns out that measuring the running time on a RAM as we propose here is nevertheless a good "first order" predictor of performance on real machines in many cases.

If we have an algorithm for solving a certain class of problems, we expect that "larger" instances of the problem will require more time to solve than "smaller" instances. Theoretical computer scientists sometimes equate the notion of an "efficient" algorithm with that of a "polynomial-time" algorithm (although not everyone takes theoretical computer scientists very seriously, especially on this point). A polynomial-time algorithm is one whose running time on inputs of length n is bounded by $n^c + d$ for some constants c and d (a "real" theoretical computer scientist will write this as $n^{O(1)}$). To make this notion mathematically precise, one needs to define the *length* of an algorithm's input.

To define the length of an input, one chooses a "reasonable" scheme to encode all possible inputs as a string of symbols from some finite alphabet, and then defines the length of an input as the number of symbols in its encoding.

We will be dealing with algorithms whose inputs consist of arbitrary integers, or lists of such integers. We describe a possible encoding scheme using the alphabet consisting of the six symbols '0', '1', '-', ',', '(', and ')'. An integer is encoded in binary, with possibly a negative sign. Thus, the length of an integer x is approximately equal to $\log_2 |x|$. We can encode a list of integers $x_1, \ldots, x_n$ of numbers as "$(\bar{x}_1, \ldots, \bar{x}_n)$", where $\bar{x}_i$ is the encoding of x_i. We can also encode lists of lists, etc., in the obvious way. All of the mathematical objects we shall wish to compute with can be encoded in this way. For example, to encode an $n \times n$ matrix of rational numbers, we may encode each rational number as a pair of integers (the numerator and denominator), each row of the matrix as a list of n encodings of rational numbers, and the matrix as a list of n encodings of rows.

It is clear that other encoding schemes are possible, giving rise to different definitions of input length. For example, we could encode inputs in some base other than 2 (but not unary!) or use a different alphabet. Indeed, it is typical to assume, for simplicity, that inputs are encoded as bit strings. However, such an alternative encoding scheme would change the definition of input length by at most a constant multiplicative factor, and so would not affect the notion of a polynomial-time algorithm.

Note that algorithms may use data structures for representing mathematical objects that look quite different from whatever encoding scheme one might choose.

Also note that in defining the notion of polynomial time on a RAM, it is essential that we restrict the sizes of numbers that may be stored in the machine's memory cells, as we have done above.

3.3 Basic Integer Arithmetic

We will need algorithms to manipulate integers of arbitrary length. Since such integers will exceed the word-size of actual machines, we represent large integers as vectors of digits to some base B, along with a bit indicating the sign. Thus, for $x \in \mathbb{Z}$, we write

$$x = \pm(\sum_{i=0}^{k-1} x_i B^i) = \pm(x_{k-1} \cdots x_1 x_0)_B,$$

where $0 \leq x_i < B$ for $0 \leq i < k$, and usually, we shall have $x_{k-1} \neq 0$. The integer x will be represented in memory as a data structure consisting of a vector of digits and a sign-bit. For our purposes, we shall consider B to be a constant, and moreover, a power of 2. The choice of B as a power of 2 allows us to extract an arbitrary bit in the binary representation of a number in time $O(1)$.

We discuss basic arithmetic algorithms for positive integers; they can be very easily adapted to deal with signed integers. All of these algorithms can be implemented directly in a programming language that provides a "built-in" signed integer type that can represent

all integers whose absolute value is less than B^2, and that provides the basic arithmetic operations (addition, subtraction, multiplication, integer division). So, for example, using the `C` programming language's `int` type on a typical 32-bit computer, we could take $B = 2^{15}$. The resulting software would be reasonably efficient, but certainly not the best possible.

Suppose we have two positive integers a and b, represented with k and ℓ base-B digits, respectively, with $k \geq \ell$. So we have $a = (a_{k-1} \cdots a_0)_B$ and $b = (b_{\ell-1} \cdots b_0)_B$. We present algorithms to compute the base-B representation of $a + b$, $a - b$, $a \cdot b$, $\lfloor a/b \rfloor$, and a rem b.

3.3.1 Addition

The sum $c = a + b$ is of the form $c = (c_k c_{k-1} \cdots c_0)_B$. Using the standard "paper-and-pencil" method (adapted from base-10 to base-B, of course), we can compute the base-B representation of $a + b$ in time $O(k)$, as follows:

$carry \leftarrow 0$
for $i \leftarrow 0$ to $k - 1$ do
    if $i < \ell$ then $tmp \leftarrow a_i + b_i + carry$ else $tmp \leftarrow a_i + carry$
    $c_i \leftarrow tmp$ rem B
    $carry \leftarrow \lfloor tmp/B \rfloor$
$c_k \leftarrow carry$

Note that in every loop iteration, the value of *carry* is 0 or 1, and the value *tmp* lies between 0 and $2B - 1$.

3.3.2 Subtraction

To compute the difference $c = a - b$, assuming that $a \geq b$, we may use the same algorithm as above, except replacing the expression "$a_i + b_i$" in the inner loop by "$a_i - b_i$." In every loop iteration, the value of *carry* is 0 or -1, and the value of *tmp* lies between $-B$ and $B - 1$. Moreover, since we are assuming that $a \geq b$, we have $c_k = 0$; that is, there is no carry out of the last loop iteration.

3.3.3 Multiplication

The product $c = a \cdot b$ is of the form $(c_{k+\ell-1} \cdots c_0)_B$, and may be computed in time $O(k\ell)$ as follows:

for $i \leftarrow 0$ to $k + \ell - 1$ do $c_i \leftarrow 0$
for $i \leftarrow 0$ to $k - 1$ do
    $carry \leftarrow 0$
    for $j \leftarrow 0$ to $\ell - 1$ do
        $tmp \leftarrow a_i b_j + c_{i+j} + carry$
        $c_{i+j} \leftarrow tmp \text{ rem } B$
        $carry \leftarrow \lfloor tmp/B \rfloor$
    $c_{i+\ell} \leftarrow carry$

Note that at every step in the above algorithm, the value of *carry* lies between 0 and $B - 1$, and the value of *tmp* lies between 0 and $B^2 - 1$.

3.3.4 Division with remainder

We now consider the problem of computing q and r such that $a = bq + r$ and $0 \leq r < b$. Let us assume that $a \geq b$; otherwise, we can just set $q = 0$ and $r = a$. Also, let us assume that $b_{\ell-1} \neq 0$. The quotient q will have at most $m = k - \ell + 1$ base-B digits. Write $q = (q_{m-1} \cdots q_0)_B$.

At a high level, the strategy we shall use to compute q and r is the following:

$r \leftarrow a$
for $i \leftarrow m - 1$ down to 0 do
    $q_i \leftarrow \lfloor r/B^i b \rfloor$
    $r \leftarrow r - B^i \cdot q_i b$

One easily verifies by induction that in each loop iteration, $0 \leq r < B^{i+1} b$, and hence each q_i will be between 0 and $B - 1$, as required.

To turn the above strategy into a detailed algorithm takes a bit of work. In particular, we want an easy way to compute $\lfloor r/B^i b \rfloor$. Now, we could in theory just try all possible choices for q_i — this would take time $O(B\ell)$, and viewing B as a constant, this is $O(\ell)$. However, this is not really very desirable from either a practical or theoretic point of view, and we can do much better with just a little effort.

We shall first consider a special case; namely, the case where $\ell = 1$. In this case, the computation of $\lfloor r/B^i b \rfloor$ is facilitated by the following:

Theorem 3.11 *Let $x \geq 0$ and $y > 0$ be integers such that $x = x'2^n + r$ for some $n \geq 0$ and $0 \leq r < 2^n$ and $y = y'2^n$. Then $\lfloor x/y \rfloor = \lfloor x'/y' \rfloor$.*

Proof. We have

$$\frac{x}{y} = \frac{x'}{y'} + \frac{r}{y'2^n} \geq \frac{x'}{y'}.$$

It follows immediately that $\lfloor x/y \rfloor \geq \lfloor x'/y' \rfloor$.

We also have

$$\frac{x}{y} = \frac{x'}{y'} + \frac{r}{y'2^n} < \frac{x'}{y'} + \frac{1}{y'} \leq \left(\left\lfloor \frac{x'}{y'} \right\rfloor + \frac{y'-1}{y'}\right) + \frac{1}{y'}.$$

Thus, we have $x/y < \lfloor x'/y' \rfloor + 1$, and hence, $\lfloor x/y \rfloor \leq \lfloor x'/y' \rfloor$. □

From this theorem, one sees that the following algorithm correctly computes the quotient and remainder in time $O(k)$:

carry $\leftarrow 0$
for $i \leftarrow k-1$ down to 0 do
    tmp $\leftarrow$ *carry* $\cdot\, B + a_i$
    $q_i \leftarrow \lfloor tmp/b_0 \rfloor$
    carry $\leftarrow$ *tmp* rem b_0
output the quotient $q = (q_{k-1} \cdots q_0)_B$ and the remainder *carry*

Note that in every loop iteration, the value of *carry* lies between 0 and $b_0 \leq B-1$, and the value of *tmp* lies between 0 and $B \cdot b_0 + (B-1) \leq B^2 - 1$.

That takes care of the special case where $\ell = 1$. Now we turn to the general case $\ell \geq 1$. In this case, we cannot so easily get the digits q_i of the quotient, but we can still fairly easily estimate these digits, using the following:

Theorem 3.12 *Let $x \geq 0$ and $y > 0$ be integers such that $x = x'2^n + r$ for some $n \geq 0$ and $0 \leq r < 2^n$ and $y = y'2^n + s$ for $0 \leq s < 2^n$. Further suppose that $2y' \geq x/y$. Then we have*

$$\lfloor x/y \rfloor \leq \lfloor x'/y' \rfloor \leq \lfloor x/y \rfloor + 2.$$

Proof. For the first inequality, note that $x/y \leq x/(y'2^n)$, and so $\lfloor x/y \rfloor \leq \lfloor x/(y'2^n) \rfloor$, and by the previous theorem, $\lfloor x/(y'2^n) \rfloor = \lfloor x'/y' \rfloor$. That proves the first inequality.

For the second inequality, first note that from the definitions, $x/y \geq x'/(y'+1)$, which is equivalent to $x'y - xy' - x \leq 0$. Now, the inequality $2y' \geq x/y$ is equivalent to $2yy' - x \geq 0$, and combining this with the inequality $x'y - xy' - x \leq 0$, we obtain $2yy' - x \geq x'y - xy' - x$, which is equivalent to $x/y \geq x'/y' - 2$. It follows that $\lfloor x/y \rfloor \geq \lfloor x'/y' \rfloor - 2$. That proves the second inequality. □

Based on this theorem, we first present an algorithm that works assuming that b is appropriately "normalized," meaning that $b_{\ell-1} \geq 2^{w-1}$, where $B = 2^w$.

It is fairly easy to normalize b, by simply multiplying both a and b by an appropriate value $2^{w'}$, where $0 \leq w' < w$; alternatively, we can use a more efficient, special-purpose "left shift" algorithm. Let $a' = a2^{w'}$ and $b' = b2^{w'}$, where b' is normalized. If we compute q and r' such that $a' = b'q + r'$, then $q = \lfloor a'/b' \rfloor = \lfloor a/b \rfloor$, and $r' = r2^{w'}$, where $r = a$ rem b. To recover r, we simply divide r' by $2^{w'}$, which we can do either using the above "single

precision" division algorithm, or by using a special-purpose "right shift" algorithm. All of the normalizing and denormalizing takes time $O(k + \ell)$.

So let us now assume that b is normalized. We obtain the quotient q and remainder r as follows:

1. for $i \leftarrow 0$ to $k - 1$ do $r_i \leftarrow a_i$
2. $r_k \leftarrow 0$
3. for $i \leftarrow k - \ell$ down to 0 do
4. $\quad q_i \leftarrow \lfloor (r_{i+\ell}B + r_{i+\ell-1})/b_{\ell-1} \rfloor$
5. $\quad$ if $q_i \geq B$ then $q_i \leftarrow B - 1$
6. $\quad carry \leftarrow 0$
7. $\quad$ for $j \leftarrow 0$ to $\ell - 1$ do
8. $\quad\quad tmp \leftarrow r_{i+j} - q_i b_j + carry$
9. $\quad\quad r_{i+j} \leftarrow tmp \text{ rem } B$
10. $\quad\quad carry \leftarrow \lfloor tmp/B \rfloor$
11. $\quad r_{i+\ell} \leftarrow carry$
12. $\quad$ while $r_{i+\ell} < 0$ do
13. $\quad\quad (r_{i+\ell} \cdots r_i)_B \leftarrow (r_{i+\ell} \cdots r_i)_B + (b_{\ell-1} \cdots b_0)_B$
14. $\quad\quad q_i \leftarrow q_i - 1$
15. output the quotient $q = (q_{k-\ell} \cdots q_0)_B$ and the remainder $r = (r_{\ell-1} \cdots r_0)_B$

Some remarks are in order:

1. In line 4, we compute q_i, which by Theorem 3.12 is greater than or equal to the true quotient digit, but exceeds this value by at most two.

2. In line 5, we reduce q_i if it is obviously too big.

3. In lines 6–10, we essentially compute

$$(r_{i+\ell} \cdots r_i)_B \leftarrow (r_{i+\ell} \cdots r_i) - q_i b.$$

   In each loop iteration, the value of *tmp* lies between $-(B^2 - B)$ and $B - 1$, and the value *carry* lies between $-(B - 1)$ and 0.

4. If the estimate q_i is too large, this is manifested by a negative value of $r_{i+\ell}$ at line 11. Lines 12–14 detect and correct this condition: the loop body here executes at most twice; the addition step in line 13 can be implemented using the same algorithm described above in §3.3.1, except that we ignore the carry out of that algorithm.

Exercise 3.13 Work out the details of an algorithm that computes the quotient and remainder for signed integers, using an algorithm for unsigned integers as a subroutine. □

Exercise 3.14 Suppose that we run the above division with remainder algorithm for $\ell > 1$ without normalizing b, but instead, we compute the value q_i in line 4 as follows:

$$q_i \leftarrow \lfloor (r_{i+\ell}B^2 + r_{i+\ell-1}B + r_{i+\ell-2})/(b_{\ell-1}R + b_{\ell-2}) \rfloor.$$

Show that q_i is either equal to the correct quotient digit, or the correct quotient digit plus 1. Note that a limitation of this approach is that the numbers involved in the computation are larger than B^2. □

Exercise 3.15 This exercise is for *C* programmers. Suppose that values of type `int` are stored using a 32-bit two's complement representation, and that all basic arithmetic operations are computed correctly modulo 2^{32}, even if an "overflow" happens to occur. Also assume that double precision floating point has 53 bits of precision, and that all basic arithmetic operations give a result with a relative error of at most 2^{-53}. Also assume that conversion from type `int` to `double` is exact, and that conversion from `double` to `int` truncates the fractional part. These assumptions reflect very typical implementations, in fact.

Now, suppose we are given `int` variables `a`, `b`, and `n`, such that $1 < \mathtt{n} < 2^{30}$ and $0 \leq \mathtt{a}, \mathtt{b} < \mathtt{n}$. Show that the when the following code sequence is executed, the value of `r` is equal to $(\mathtt{a} \cdot \mathtt{b}) \bmod \mathtt{n}$:

```
int q;

q  = (int) ((((double) a) * ((double) b)) / ((double) n));
r = a*b - q*n;

if (r >= n)
   r = r - n;
else if (r < 0)
   r = r + n;
```

□

3.3.5 Summary

We now summarize the above observations. For an integer n, we define $\mathrm{len}(n)$ to be the number of bits in the binary representation of $|n|$; more precisely,

$$\mathrm{len}(n) = \begin{cases} \lfloor \log_2 |n| \rfloor + 1 & \text{if } n \neq 0, \\ 1 & \text{if } n = 0. \end{cases}$$

Notice that for $n > 0$, we have $\log_2 n < \mathrm{len}(n) \leq \log_2 n + 1$.

Theorem 3.16 *Let a and b be arbitrary integers, represented using the data structures described above.*

(i) We can determine an arbitrary bit in the binary representation of $|a|$ in time $O(1)$.

(ii) We can compute $a \pm b$ in time $O(\mathrm{len}(a) + \mathrm{len}(b))$.

(iii) We can compute $a \cdot b$ in time $O(\text{len}(a)\,\text{len}(b))$.

(iv) If $b > 0$, we can compute q and r such that $a = bq + r$ and $0 \leq r < b$ in time $O(\text{len}(b)\,\text{len}(q))$.

From now on, we shall not worry about the implementation details of long-integer arithmetic, and will just refer directly this theorem.

Note the bound $O(\text{len}(b)\,\text{len}(q))$ in part (iv) of this theorem, which may be significantly less than the bound $O(\text{len}(a)\,\text{len}(b))$.

This theorem does not refer to the base B in the underlying implementation. The choice of B affects the values of the implied big-'O' constants; while in theory, this is of no significance, it does have a significant impact in practice.

A note on notation. In expressing the running times of algorithms, we generally prefer to write, for example, $O(\text{len}(a)\,\text{len}(b))$, rather than $O((\log a)(\log b))$. There are two reasons for this. The first is esthetic: the function "len" stresses the fact that running times should be expressed in terms of the bit length of the inputs. The second is technical: O-estimates involving expressions containing several independent parameters, like $O(\text{len}(a)\,\text{len}(b))$, should be valid for *all* possible values of the parameters, since the notion of "sufficiently large" does not make sense in his setting; because of this, it is very inconvenient to have functions, like log, that vanish or are undefined on some inputs.

Exercise 3.17 Show that the product n of integers $n_1, \ldots, n_k$, with each $n_i > 1$, can be computed in time $O(\text{len}(n)^2)$. Do not assume that k is a constant. □

Exercise 3.18 Show that given integers $n_1, \ldots, n_k$, with each $n_i > 1$, and an integer $0 \leq z < n$, where $n = \prod_i n_i$, we can compute the k integers $z \bmod n_i$, for $1 \leq i \leq k$, in time $O(\text{len}(n)^2)$. □

Exercise 3.19 The quadratic-time algorithms presented here for integer multiplication and division are by no means the fastest possible. This exercise develops a faster multiplication algorithm, originally invented by Karacuba and Ofman. Suppose we have two positive, ℓ-bit integers integers a and b such that $a = a_1 2^k + a_0$ and $b = b_1 2^k + b_0$, where $0 \leq a_0 < 2^k$ and $0 \leq b_0 < 2^k$. Then

$$ab = a_1 b_1 2^{2k} + (a_0 b_1 + a_1 b_0) 2^k + a_0 b_0.$$

Show how to compute the product ab in time $O(\ell)$, given the products $a_0 b_0$, $a_1 b_1$, and $(a_0 - a_1)(b_0 - b_1)$. From this, design a recursive algorithm that computes ab in time $O(\ell^{\log_2 3})$, where $\log_2 3 \approx 1.6$. □

In the following exercises, assume that we have an algorithm that multiplies two integers of at most ℓ bits in time $M(\ell)$. It is convenient (and reasonable) to assume that M is a **well-behaved complexity function**. By this, we mean that M maps non-negative integers to non-negative real numbers, and

- for all non-negative integers a and b, $M(a+b) \geq M(a) + M(b)$, and
- for all positive integers a, there exists a positive integer b, such that for all non-negative integers n, $M(an) \leq bM(n)$.

The first condition says that M grows at least linearly in n, while the second says that M does not grow "too fast." The reader may verify that these conditions imply that M is a non-decreasing function, that $M(0) = 0$, and that if $M(n) > 0$ for any n, then $M(n) > 0$ for all $n > 0$.

Exercise 3.20 Give an algorithm for Exercise 3.17 that runs in time $O(M(\text{len}(n))\,\text{len}(k))$. □

Exercise 3.21 We can represent a "floating point" number $\hat{z}$ as a pair (a, e), where a and e are integers — the value of $\hat{z}$ is the number $a2^e$, and we call $\text{len}(a)$ the **precision** of $\hat{z}$. We say that $\hat{z}$ is a k**-bit approximation** of a real number z if $\hat{z}$ has precision k and $\hat{z} = z(1+\epsilon)$ for some $|\epsilon| \leq 2^{-k+1}$. Show how to compute — given positive integers b and k — a k-bit approximation to $1/b$ in time $O(M(k))$. Hint: using Newton iteration, show how to go from a t-bit approximation of $1/b$ to a $(2t-2)$-bit approximation of $1/b$, making use of just the high-order $O(t)$ bits of b, in time $O(M(t))$. □

Exercise 3.22 Using the result of the previous exercise, given positive integers a and b of bit length at most ℓ, show how to compute $\lfloor a/b \rfloor$ and $a \bmod b$ in time $O(M(\ell))$. From this, we see that up to a constant factor, division with remainder is no harder that multiplication. □

Exercise 3.23 Using the result of the previous exercise, give an algorithm for Exercise 3.18 that runs in time $O(M(\text{len}(n))\,\text{len}(k))$. □

Exercise 3.24 Show that for integer $n \geq 0$, we can compute $\lfloor n^{1/2} \rfloor$ in time $O(M(\text{len}(n)))$. Hint: Newton iteration. □

Exercise 3.25 Suppose we have an algorithm that computes the square of an ℓ-bit integer in time $S(\ell)$, where S is a well-behaved complexity function. Show how to use this algorithm to compute the product of two arbitrary integers of at most ℓ bits in time $O(S(\ell))$. □

3.4 Computing in $\mathbb{Z}_n$

Let $n > 1$. For computational purposes, we may represent elements of $\mathbb{Z}_n$ as elements of the set $\{0, \ldots, n-1\}$.

Addition and subtraction in $\mathbb{Z}_n$ can be performed in time $O(\text{len}(n))$. Multiplication can be performed in time $O(\text{len}(n)^2)$ with an ordinary integer multiplication, followed by a division with remainder.

A note on notation. In describing algorithms, as well as in other contexts, if α, β are elements of $\mathbb{Z}_n$, we may write, e.g., $\gamma \leftarrow \alpha + \beta$ or $\gamma \leftarrow \alpha\beta$, and it is understood that elements of $\mathbb{Z}_n$ are represented as discussed above, as integers between 0 and $n-1$, and the arithmetic on the representations is done modulo n. Thus, we have in mind a "strongly typed" language for our pseudo-code that makes a clear distinction between integers in the set $\{0, \ldots, n-1\}$ and elements of $\mathbb{Z}_n$. If $a \in \mathbb{Z}$, we can convert a to an object $\alpha \in \mathbb{Z}_n$ by writing $\alpha \leftarrow [a \bmod n]$, and if $a \in \{0, \ldots, n-1\}$, this type conversion is purely conceptual, involving no actual computation. Conversely, if $\alpha \in \mathbb{Z}_n$, we can convert α to an object $a \in \{0, \ldots, n-1\}$, by writing $a \leftarrow \mathrm{rep}(\alpha)$; again, this type conversion is purely conceptual, and involves no actual computation.

Another interesting problem is exponentiation in $\mathbb{Z}_n$: given $\alpha \in \mathbb{Z}_n$ and a non-negative integer e, compute $\alpha^e \in \mathbb{Z}_n$. Perhaps the most obvious way to do this is to iteratively multiply by α a total of e times, requiring time $O(e\ \mathrm{len}(n)^2)$. A much faster algorithm, the **repeated-squaring algorithm**, computes α^e using just $O(\mathrm{len}(e))$ multiplications in $\mathbb{Z}_n$, thus taking time $O(\mathrm{len}(e)\ \mathrm{len}(n)^2)$.

This method works as follows. Let $e = (b_{\ell-1} \cdots b_0)_2$ be the binary expansion of e (where b_0 is the low-order bit). For $0 \le i \le \ell$, define $e_i = \lfloor e/2^i \rfloor$; the binary expansion of e_i is $e_i = (b_{\ell-1} \cdots b_i)_2$. Also define, for $0 \le i \le \ell$, $\beta_i = \alpha^{e_i}$, so $\beta_\ell = 1$ and $\beta_0 = \alpha^e$. Then we have

$$e_i = 2e_{i+1} + b_i \quad (0 \le i < \ell),$$

and hence

$$\beta_i = \beta_{i+1}^2 \cdot \alpha^{b_i} \quad (0 \le i < \ell).$$

This idea yields the following algorithm:

$\beta \leftarrow 1$
for $i \leftarrow \ell - 1$ down to 0 do
    $\beta \leftarrow \beta^2$
    if $b_i = 1$ then $\beta \leftarrow \beta \cdot \alpha$
output y

It is clear that when this algorithm terminates, $\beta = \alpha^e$, and that the running-time estimate is as claimed above. Indeed, the algorithm uses ℓ squarings in $\mathbb{Z}_n$, and at most ℓ additional multiplications in $\mathbb{Z}_n$.

The following exercises develop some important efficiency improvements to the basic repeated squaring algorithm.

Exercise 3.26 By using a "2^k-ary" approach, instead of a binary approach, show how to modify the repeated squaring so as to compute α^e using at most ℓ squarings in $\mathbb{Z}_n$, and an additional $2^k + \ell/k + O(1)$ multiplications in $\mathbb{Z}_n$. As above, $\alpha \in \mathbb{Z}_n$ and $\mathrm{len}(e) = \ell$, while k is a parameter that we are free to choose. Hint: first build a table of powers $1, \alpha, \ldots, \alpha^{2^k-1}$. Also show that by appropriately choosing the parameter k, we can make the number of

additional multiplications $O(\ell/\operatorname{len}(\ell))$. Thus, the cost of exponentiation is essentially the cost of ℓ squarings. □

Exercise 3.27 Suppose we are given $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}_n$, along with non-negative integers $e_1, \ldots, e_k$, where $\operatorname{len}(e_i) \leq \ell$ for $1 \leq i \leq k$. Show how to compute

$$\beta := \alpha_1^{e_k} \cdots \alpha_k^{e_k}$$

using at most ℓ squarings and an additional $\ell + 2^k + O(1)$ multiplications. Your algorithm should work in two phases: in the first phase, the algorithm uses just the values $\alpha_1, \ldots, \alpha_k$ and ℓ to build a table, performing $2^k + O(1)$ multiplications; in the second phase, the algorithm computes β, using the exponents $e_1, \ldots, e_k$, and the table computed in the first phase. computes a table of information □

Exercise 3.28 Suppose that we are to compute α^e, where $\alpha \in \mathbb{Z}_n$, for many ℓ-bit exponents e, but with α fixed. Show that for any positive integer parameter k, we can make a precomputation, depending on α, that uses $O(\ell+2^k)$ multiplications in $\mathbb{Z}_n$, so that after the precomputation, we can compute α^e for any ℓ-bit exponent e using just $O(\ell/k)$ multiplications in $\mathbb{Z}_n$. Hint: use the previous exercise. □

3.5 Notes

The "classical" algorithms presented here for integer multiplication and division are by no means the best possible. The most practical algorithms take advantage of low-level "assembly language" codes specific to a particular machine's architecture (e.g., the GNU Multi-Precision library GMP, available as `http://www.swox.com/gmp`). Moreover, there are algorithms whose running time is asymptotically faster. We saw this already with Karacuba and Ofman's algorithm [39] in Exercise 3.19, which allows us to multiply two ℓ-bit integers in time $O(\ell^{\log_2 3})$. If a and b are two integers whose length in bits is bounded by ℓ, then the fastest known algorithm for computing ab on a RAM runs in time $O(\ell)$. This algorithm is due to Schönhage, and actually works on a very restricted type of RAM called a "pointer machine" (see Problem 12, Section 4.3.3 of Knuth [41]).

Another model of computation is that of **boolean circuits**. In this model of computation, one considers families of boolean circuits (with, say, the usual "and," "or," and "not" gates) that compute a particular function — for every input length, there is a different circuit in the family that computes the function on inputs of that length. One natural notion of complexity for such circuit families is the **size**, i.e., number of gates and wires, of the circuit, which is measured as a function of the input length. The smallest known boolean circuit that multiplies two ℓ-bit numbers has size $O(\ell \operatorname{len}(\ell) \operatorname{len}(\operatorname{len}(\ell)))$. This result is due to Schönhage and Strassen [63].

It is hard to say which model of computation, the RAM or circuits, is "better." On the one hand, the RAM very naturally models computers as we know them today. On the other hand, one can "cheat" a bit in the RAM model by stuffing $O(\operatorname{len}(\ell))$-bit integers into

"words" on the RAM that would not fit into words on a real machine. For example, even with the simple quadratic-time algorithms discussed above, we can choose the base B to have $\text{len}(\ell)$ bits, in which case these algorithms would run in time $O((\ell/\text{len}(\ell))^2)$.

In the remainder of this text, unless otherwise specified, we shall always use the classical $O(\ell^2)$ bounds for integer multiplication and division, which have the advantage of being both simple and reasonably reliable predictors of actual performance for small to moderately sized inputs. For relatively large numbers, experience shows that the classical algorithms are definitely not the best — Karacuba and Ofman's multiplication algorithm does significantly better than the classical algorithms on inputs of a thousand bits or so (the exact crossover depends on myriad implementation details). Thus, the reader should bear in mind that for serious computations involving very large numbers, the faster algorithms are very important, even though this text does not discuss them at great length.

For a good survey of asymptotically fast algorithms for integer arithmetic, see Chapter 9 of Crandall and Pomerance [23], as well as Chapter 4 of Knuth [41].

Chapter 4

Euclid's Algorithm

In this chapter, we discuss Euclid's algorithm for computing greatest common divisors. It turns out that Euclid's algorithm has a number of very nice properties, and has applications far beyond that of just computing greatest common divisors.

4.1 The Basic Euclidean Algorithm

We consider the following problem: given two non-negative integers a and b, compute $\gcd(a, b)$. We can do this using the well-known algorithm of Euclid, which is described in the following theorem.

Theorem 4.1 *Let $a \geq b \geq 0$, with $a > 0$. Define the integers $r_0, r_1, \ldots, r_{\ell+1}$, and $q_1, \ldots, q_\ell$, where $\ell \geq 0$, as follows:*

$$\begin{aligned}
r_0 &= a, \\
r_1 &= b, \\
r_0 &= r_1 q_1 + r_2 \quad (0 < r_2 < r_1), \\
&\vdots \\
r_{i-1} &= r_i q_i + r_{i+1} \quad (0 < r_{i+1} < r_i), \\
&\vdots \\
r_{\ell-2} &= r_{\ell-1} q_{\ell-1} + r_\ell \quad (0 < r_\ell < r_{\ell-1}), \\
r_{\ell-1} &= r_\ell q_\ell \quad (r_{\ell+1} = 0).
\end{aligned}$$

Then $r_\ell = \gcd(a, b)$. Moreover, if $b > 0$, then then $\ell \leq \log b / \log \phi + 1$, where $\phi = (1 + \sqrt{5})/2 \approx 1.62$, and if $b = 0$, then $\ell = 0$.

Proof. For the first statement, one sees that for $1 \leq i \leq \ell$, the common divisors of r_{i-1} and r_i are the same as the common divisors of r_i and r_{i+1}, and hence $\gcd(r_{i-1}, r_i) = \gcd(r_i, r_{i+1})$. From this, it follows that $\gcd(a, b) = \gcd(r_0, r_1) = \gcd(r_\ell, 0) = r_\ell$.

To prove the second statement, assume that $b > 0$. We claim that for $0 \leq i \leq \ell - 1$, $r_{\ell-i} \geq \phi^i$. The statement will then follow by setting $i = \ell - 1$ and taking logarithms.

If $\ell = 1$, the claim is obviously true, so assume $\ell > 1$. We have $r_\ell \geq 1 = \phi^0$ and $r_{\ell-1} \geq r_\ell + 1 \geq 2 \geq \phi^1$. For $2 \leq i \leq \ell - 1$, using induction and applying the fact the $\phi^2 = \phi + 1$, we have

$$r_{\ell-i} \geq r_{\ell-(i-1)} + r_{\ell-(i-2)} \geq \phi^{i-1} + \phi^{i-2} = \phi^{i-2}(1 + \phi) = \phi^i,$$

which proves the claim. □

Example 4.2 Suppose $a = 100$ and $b = 35$. Then the numbers appearing in Theorem 4.1 are easily computed as follows:

i	0	1	2	3	4
r_i	100	35	30	5	0
q_i		2	1	6	

So we have $\gcd(a, b) = r_3 = 5$. □

We can easily turn the scheme described in Theorem 4.1 into a simple algorithm, taking as input integers a, b, with $a \geq b$ and $a > 0$:

while $b \neq 0$ do
    Compute q, r such that $a = bq + r$, with $0 \leq r < b$
    $(a, b) \leftarrow (b, r)$
output a

By Theorem 4.1, this algorithm, known as **Euclid's algorithm**, outputs the greatest common divisor of a and b.

Theorem 4.3 *Euclid's algorithm runs in time* $O(\text{len}(a)\,\text{len}(b))$.

Proof. We may assume that $b > 0$. The running time is $O(\tau)$, where $\tau = \sum_{i=1}^{\ell} \text{len}(r_i)\,\text{len}(q_i)$. We have

$$\tau \leq \text{len}(b) \sum_i \text{len}(q_i) \leq \text{len}(b) \sum_i (\log_2 q_i + 1) = \text{len}(b)(\ell + \log_2(\prod_i q_i)).$$

Note that

$$a = r_0 \geq r_1 q_1 \geq r_2 q_2 q_1 \geq \cdots \geq r_\ell q_\ell \cdots q_1 \geq q_\ell \cdots q_1.$$

We also have $\ell \leq \log b / \log \phi + 1$. Combining this with the above, we have

$$\tau \leq \text{len}(b)(\log b / \log \phi + 1 + \log_2 a) = O(\text{len}(a)\,\text{len}(b)),$$

which proves the theorem. □

Exercise 4.4 This exercise looks at an alternative algorithm for computing $\gcd(a, b)$, called the **binary gcd algorithm**, which can be directly implemented using just additions, subtraction, and "shift" operations, which on real-world computers, are often very efficiently implemented. In practice, this algorithm is usually faster than Euclid's algorithm.

For integer $n = 2^e m$, with m odd, let $\text{EvenPart}(n) := 2^e$ and $\text{OddPart}(n) := m$. The algorithm takes positive integers a and b as input, and runs as follows:

$c \leftarrow \min(\text{EvenPart}(a), \text{EvenPart}(b))$
$a \leftarrow \text{OddPart}(a)$, $b \leftarrow \text{OddPart}(b)$
$(a, b) \leftarrow (\max(a, b), \min(a, b))$
$v \leftarrow a - b$
while $v \neq 0$ do
    $v \leftarrow \text{OddPart}(v)$
    $(a, b) \leftarrow (\max(v, b), \min(v, b))$
    $v \leftarrow a - b$
output $c \cdot a$

Show that this algorithm correctly computes $\gcd(a, b)$, and runs in time $O(\ell^2)$, where $\ell := \max(\text{len}(a), \text{len}(b))$. □

4.2 The Extended Euclidean Algorithm

Let $d = \gcd(a, b)$. We know that there exist integers s and t such that $as + bt = d$. The **extended Euclidean algorithm** allows us to compute s and t. The following theorem describes the algorithm, and also states a number of important facts about the relative sizes of the numbers that arise during the computation — these size estimates will play a crucial role, both in the analysis of the running time of the algorithm, as well as in applications of the algorithm that we will discuss later.

Theorem 4.5 *Let a, b, $r_0, r_1, \ldots, r_{\ell+1}$, and $q_1, \ldots, q_\ell$ be as in Theorem 4.1. Define integers $s_0, s_1, \ldots, s_{\ell+1}$ and $t_0, t_1, \ldots, t_{\ell+1}$ as follows:*

$$s_0 := 1, \quad t_0 := 0,$$

$$s_1 := 0, \quad t_1 := 1,$$

and for $1 \leq i \leq \ell$,

$$s_{i+1} := s_{i-1} - s_i q_i, \quad t_{i+1} := t_{i-1} - t_i q_i.$$

Then

(i) for $0 \leq i \leq \ell + 1$, we have $s_i a + t_i b = r_i$; in particular, $s_\ell a + t_\ell b = \gcd(a, b)$;

(ii) for $0 \leq i \leq \ell$, we have $s_i t_{i+1} - t_i s_{i+1} = (-1)^i$;

(iii) *for* $0 \le i \le \ell + 1$, *we have* $\gcd(s_i, t_i) = 1$;

(iv) *we have* $|s_{\ell+1}| \le b$ *and* $|t_{\ell+1}| \le a$;

(v) *for* $0 \le i \le \ell$, *we have* $t_i t_{i+1} \le 0$ *and* $|t_i| \le |t_{i+1}|$; *for* $1 \le i \le \ell$, *we have* $s_i s_{i+1} \le 0$ *and* $|s_i| \le |s_{i+1}|$;

(vi) *for* $1 \le i \le \ell + 1$, *we have* $|s_i| \le b$, *and for* $0 \le i \le \ell + 1$, *we have* $|t_i| \le a$;

(vii) *for* $1 \le i \le \ell + 1$, *we have* $|s_i| \le b/r_{i-1}$ *and* $|t_i| \le a/r_{i-1}$.

Proof. (i) is easily proved by induction on i. For $i = 0, 1$, the statement is clear. For $1 \le i \le \ell$, we have

$$\begin{aligned} s_{i+1}a + t_{i+1}b &= (s_{i-1} - s_i q_i)a + (t_{i-1} - t_i q_i)b \\ &= (s_{i-1}a + t_{i-1}b) - (s_i a + t_i b)q_i \\ &= r_{i-1} - r_i q_i \quad \text{(by induction)} \\ &= r_{i+1}. \end{aligned}$$

(ii) is also easily proved by induction on i. For $i = 0$, the statement is clear. For $1 \le i \le \ell$, we have

$$\begin{aligned} s_i t_{i+1} - t_i s_{i+1} &= s_i(t_{i-1} - t_i q_i) - t_i(s_{i-1} - s_i q_i) \\ &= -(s_{i-1}t_i - t_{i-1}s_i) \quad \text{(after expanding and simplifying)} \\ &= -(-1)^{i-1} \;=\; (-1)^i \quad \text{(by induction)}. \end{aligned}$$

(iii) follows directly from (ii).

To prove (iv), note that $s_{\ell+1}a + t_{\ell+1}b = r_{\ell+1} = 0$. We have $t_{\ell+1} \neq 0$, since otherwise, both $s_{\ell+1}$ and $t_{\ell+1}$ would be zero, contradicting (iii). So (iv) follows from the fact that the fractions $-b/a$ and $s_{\ell+1}/t_{\ell+1}$ are equal, and the fact that, again by (iii), the latter fraction is in lowest terms.

For (v), one can easily prove both statements about by induction on i. Both statements are clearly true for $i = 0$. For $1 \le i \le \ell$, since $s_{i+1} = s_{i-1} - s_i q_i$, and since by the induction hypothesis s_{i-1} and s_i have opposite sign and $|s_i| \ge |s_{i-1}|$, it is clear that $|s_{i+1}| = |s_{i-1}| + |s_i|q_i \ge |s_i|$, and that the sign of s_{i+1} is the opposite of that of s_i. The proof of the corresponding statement for t_{i+1} is the same.

(vi) follows immediately from (iv) and (v).

For (vii), one considers the two equations:

$$\begin{aligned} s_{i-1}a + t_{i-1}b &= r_{i-1} \\ s_i a + t_i b &= r_i. \end{aligned}$$

Subtracting t_{i-1} times the second equation from t_i times the first, applying (ii), and using the fact from (v) that t_i and t_{i-1} have opposite sign, we obtain

$$a = |t_i r_{i-1} - t_{i-1} r_i| \ge |t_i| r_{i-1},$$

from which the bound for t_i follows. The bound for s_i follows similarly, subtracting s_i times the first equation from s_{i-1} times the second. □

Example 4.6 We continue with Example 4.2. The numbers s_i and t_i are easily computed from the q_i:

i	0	1	2	3	4
r_i	100	35	30	5	0
q_i		2	1	6	
s_i	1	0	1	-1	7
t_i	0	1	-2	3	-20

□

We can easily turn the scheme described in Theorem 4.5 into a simple algorithm, taking as input integers a, b, such that $a \geq b$ and $a > 0$:

$s \leftarrow 1,\ t \leftarrow 0$
$s' \leftarrow 0,\ t' \leftarrow 1$
while $b \neq 0$ do
    Compute q, r such that $a = bq + r$, with $0 \leq r < b$
    $(s, t, s', t') \leftarrow (s', t', s - s'q, t - t'q)$
    $(a, b) \leftarrow (b, r)$
output a, s, t

This algorithm outputs (d, s, t) such that $d = \gcd(a, b)$ and $as + bt = d$.

Theorem 4.7 *The extended Euclidean algorithm runs in time $O(\mathrm{len}(a)\,\mathrm{len}(b))$.*

Proof. We may assume that $b > 0$. It suffices to analyze the cost of computing the sequences $\{s_i\}$ and $\{t_i\}$. Consider first the cost of computing all of the t_i, which is $O(\tau)$, where $\tau = \sum_{i=1}^{\ell} \mathrm{len}(t_i)\,\mathrm{len}(q_i)$. By Theorem 4.5 part (vi), and arguing as in the proof of Theorem 4.3, we have

$$\begin{aligned} \tau &= \mathrm{len}(q_1) + \sum_{i=2}^{\ell} \mathrm{len}(t_i)\,\mathrm{len}(q_i) \leq \mathrm{len}(q_1) + \mathrm{len}(a)(\ell - 1 + \log_2(\prod_{i=2}^{\ell} q_i)) \\ &= O(\mathrm{len}(a)\,\mathrm{len}(b)), \end{aligned}$$

using the fact that $\prod_{i=2}^{\ell} q_i \leq b$. An analogous argument shows that one can compute all of the s_i also in time $O(\mathrm{len}(a)\,\mathrm{len}(b))$, and in fact, in time $O(\mathrm{len}(b)^2)$. □

Another, instructive way to view Theorem 4.5 is as follows.
For $1 \leq i \leq \ell$, we have

$$\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix}.$$

Recursively expanding the right-hand side of this equation, we have for $0 \leq i \leq \ell$

$$\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = M_i \begin{pmatrix} a \\ b \end{pmatrix},$$

where for $1 \leq i \leq \ell$, M_i is defined as

$$M_i := \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix}.$$

If we define M_0 to be the identity matrix, then it is easy to see that

$$M_i = \begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix},$$

for $0 \leq i \leq \ell$. From this observation, part (i) of Theorem 4.5 is immediate, and part (ii) follows from the fact that M_i is the product of i matrices, each of determinant -1, and the determinant of M_i is evidently $s_i t_{i+1} - t_i s_{i+1}$.

Exercise 4.8 Develop an "extended" binary gcd algorithm; i.e., a variation of the binary gcd algorithm in Exercise 4.4 that efficiently computes $d = \gcd(a, b)$, along with integers s and t such that $as + bt = d$, and which uses only addition, subtraction, and "shift" operations. □

4.3 Computing Modular Inverses and Chinese Remaindering

One application of the Extended Euclidean algorithm is to the problem of computing multiplicative inverses in $\mathbb{Z}_n$.

Given $a \in \{0, \ldots, n-1\}$, we can determine if $[a \bmod n]$ has a multiplicative inverse in $\mathbb{Z}_n$, and if so, determine this inverse, in time $O(\text{len}(n)^2)$, as follows. We run the extended Euclidean algorithm on input (n, a) to determine integers d, s, and t, such that $d = \gcd(n, a)$ and $ns + at = d$. If $d \neq 1$, then $[a \bmod n]$ is not invertible; otherwise, $[a \bmod n]$ is invertible, and $[t \bmod n]$ is its inverse. In the latter case, by part (vi) of Theorem 4.5, we know that $|t| \leq n$; we cannot have $t = \pm n$, and so either $t \in \{0, \ldots, n-1\}$, or $t + n \in \{0, \ldots, n-1\}$.

We also observe that Theorem 2.7 (Chinese Remainder Theorem) can be made computationally effective as well.

Theorem 4.9 *Given integers $n_1, \ldots, n_k$, and $a_1, \ldots, a_k$, with $n_i > 1$, $\gcd(n_i, n_j) = 1$ for $i \neq j$, and $0 \leq a_i < n_i$, we can compute z such that $0 \leq z < n$ and $z \equiv a_i \pmod{n_i}$ in time $O(\text{len}(n)^2)$, where $n = \prod_i n_i$.*

Proof. Exercise (just use the formulas in the proof of Theorem 2.7, and see Exercises 3.17 and 3.18). □

Exercise 4.10 Suppose that we are given two distinct k-bit primes, p and q, an element $\alpha \in \mathbb{Z}_n$, where $n = pq$, and an integer e, where $0 \leq e < n$. Show how to compute $\alpha^e \in \mathbb{Z}_n$ using at most k squarings modulo p, k squarings modulo q, and additional computations whose running time is $o(k^3)$. □

Exercise 4.11 We are given a positive integer n, two elements $\alpha, \beta \in \mathbb{Z}_n$, and integers e and f such that $\alpha^e = \beta^f$ and $\gcd(e, f) = 1$. Show how to efficiently find some $\gamma \in \mathbb{Z}_n$ such that $\gamma^e = \beta$. □

Exercise 4.12 In this exercise and the next, you are to analyze an "incremental Chinese Remaindering" algorithm. Consider the following algorithm, which takes as input integers z, n, z', n', where n and n' are positive integers such that

$$n' > 1, \quad \gcd(n, n') = 1, \quad 0 \leq z < n, \quad \text{and} \quad 0 \leq z' < n'.$$

It outputs integers z'', n'', such that

$$n'' = nn', \quad 0 \leq z'' < n'', \quad z'' \equiv z \pmod{n}, \quad \text{and} \quad z'' \equiv z' \pmod{n'}.$$

It runs as follows:

1. Compute $\tilde{n}$ such that $n\tilde{n} \equiv 1 \pmod{n'}$ and $0 \leq \tilde{n} < n'$.
2. Set $h \leftarrow ((z' - z)\tilde{n}) \text{ rem } n'$.
3. Set $z'' \leftarrow z + nh$.
4. Set $n'' \leftarrow nn'$.
5. Output z'', n''.

Show that the output z'', n'' of the algorithm satisfies the conditions stated above, and estimate its running time. □

Exercise 4.13 Using the algorithm in the previous exercise as a subroutine, give a simple $O(\text{len}(n)^2)$ algorithm that takes as input integers $n_1, \ldots, n_k$, and $a_1, \ldots, a_k$, with $n_i > 1$, $\gcd(n_i, n_j) = 1$ for $i \neq j$, and $0 \leq a_i < n_i$, and outputs z, n such that $0 \leq z < n$, $z \equiv a_i \pmod{n_i}$, and $n = \prod_i n_i$. The algorithm should be "incremental," in that it processes the pairs (n_i, a_i) one at a time, using time $O(\text{len}(n)\,\text{len}(n_i))$ to process each such pair. □

4.4 Speeding up Algorithms via Modular Computation

An important practical application of the above "computational" version (Theorem 4.9) of the Chinese Remainder Theorem is a general algorithmic technique that can significantly speed up certain types of computations involving long integers. Instead of trying to describe

the technique in some general form, we simply illustrate the technique by means of a specific example: integer matrix multiplication.

Suppose we have two $\ell \times \ell$ matrices A and B whose entries are large integers, and we want to compute the product matrix $C = AB$. If the entries of A are (a_{rs}) and the entries of B are (b_{st}), then the entries (c_{rt}) of C are given by the usual rule for matrix multiplication:

$$c_{rt} = \sum_{s=1}^{\ell} a_{rs} b_{st}.$$

Suppose further that M is the maximum absolute of the entries in A and B, so that the entries in C are bounded in absolute value by $M' := M^2\ell$. Then by just applying the above formula, we can compute the entries of C using ℓ^3 multiplications of numbers of length at most $\mathrm{len}(M)$, and ℓ^3 additions of numbers at length at most $\mathrm{len}(M')$, where $\mathrm{len}(M') \leq 2\,\mathrm{len}(M) + \mathrm{len}(\ell)$. This yields a running time of

$$O(\ell^3\,\mathrm{len}(M)^2 + \ell^3\,\mathrm{len}(\ell)). \tag{4.1}$$

If the entries of A and B are large relative to ℓ, specifically, if $\mathrm{len}(\ell) = O(\mathrm{len}(M)^2)$, then the running time is dominated by the first term above, i.e., it is

$$O(\ell^3\,\mathrm{len}(M)^2).$$

Using the Chinese Remainder Theorem, we can actually do much better than this, as follows.

For any integer $n > 1$, and for all $1 \leq r, t \leq \ell$, we have

$$c_{rt} \equiv \sum_{s=1}^{\ell} a_{rs} b_{st} \pmod{n}. \tag{4.2}$$

Moreover, if we compute integers c'_{rt} such that

$$c'_{rt} \equiv \sum_{s=1}^{\ell} a_{rs} b_{st} \pmod{n} \tag{4.3}$$

and if we also have

$$-n/2 \leq c'_{rt} < n/2 \quad \text{and} \quad n > 2M', \tag{4.4}$$

then we must have

$$c_{rt} = c'_{rt}. \tag{4.5}$$

To see why (4.5) follows from (4.3) and (4.4), observe that (4.2) and (4.3) imply that $c_{rt} \equiv c'_{rt} \pmod{n}$, i.e., that n divides $(c_{rt} - c'_{rt})$. Then from the bound $|c_{rt}| \leq M'$ and from (4.4), we obtain

$$|c_{rt} - c'_{rt}| \leq |c_{rt}| + |c'_{rt}| \leq M' + n/2 < n/2 + n/2 = n.$$

So we see that the quantity $(c_{rt} - c'_{rt})$ is a multiple of n, while at the same time this quantity is strictly less than n in absolute value; hence, this quantity must be zero. That proves (4.5).

So from the above discussion, to compute C, it suffices to compute the entries of C modulo n, where we have to make sure that we compute "balanced" remainders in the interval $[-n/2, n/2)$, rather than the more usual "least non-negative" remainders.

To compute C modulo n, we choose a number of small integers $n_1, \ldots, n_k$, relatively prime in pairs, and such that the product $n := n_1 \cdots n_k$ is strictly greater than $2M'$. In practice, one would choose the n_i's to be small primes, and a table of such primes could easily be computed in advance, so that all problems problems up to a given size could be handled. For example, the product of all primes of at most 16 bits is a number that has more than $90,000$ bits. Thus, by simply pre-computing and storing such a table of small primes, we can handle input matrices with quite large entries (up to about $45,000$ bits).

Let us assume that we have pre-computed appropriate small primes $n_1, \ldots, n_k$. Further, we shall assume that addition and multiplication modulo any of the n_i's can be done in *constant* time. This is reasonable, both from a practical and theoretical point of view, since such primes easily "fit" into a memory cell.

To compute C, we execute the following steps:

1. For each $i = 1, \ldots, k$, do the following:

   (a) compute $\hat{a}_{rs}^{(i)} \leftarrow a_{rs} \text{ rem } n_i$ for $1 \leq r, s \leq \ell$,

   (b) compute $\hat{b}_{st}^{(i)} \leftarrow b_{st} \text{ rem } n_i$ for $1 \leq s, t \leq \ell$,

   (c) For $1 \leq r, t \leq \ell$, compute

   $$\hat{c}_{rt}^{(i)} \leftarrow \sum_{s=1}^{\ell} \hat{a}_{rs}^{(i)} \hat{b}_{st}^{(i)} \text{ rem } n_i.$$

2. For each $1 \leq r, t \leq \ell$, apply the Chinese Remainder Theorem to $\hat{c}_{rt}^{(1)}, \hat{c}_{rt}^{(2)}, \ldots, \hat{c}_{rt}^{(k)}$, obtaining an integer c_{rt}, which should be computed as a balanced remainder modulo n, i.e., $n/2 \leq c_{rt} < n/2$.

3. Output $(c_{rt} : 1 \leq r, t \leq \ell)$.

Note that in Step 2, if our Chinese Remainder algorithm happens to be implemented to return an integer z with $0 \leq z < n$, we can easily get a balanced remainder by just subtracting n from z if $z \geq n/2$.

The correctness of the above algorithm has already been established. Let us now analyze its running time. The running time of Step 1 is easily seen (c.f., Exercise 3.18) to be $O(\ell^2 \operatorname{len}(M) \operatorname{len}(M'))$. Under our assumption about the cost of arithmetic modulo small primes, the cost of Step 2 is $O(\ell^3 k)$, and since $k = O(\operatorname{len}(M')) = O(\operatorname{len}(M) + \operatorname{len}(\ell))$, the cost of this step is $O(\ell^3(\operatorname{len}(M) + \operatorname{len}(\ell)))$. Finally, the cost of Step 3 is also $O(\ell^2 \operatorname{len}(M')^2)$.

Thus, the total running time of this algorithm is easily calculated (discarding terms that are dominated by others) as

$$O(\ell^2 \operatorname{len}(M)^2 + \ell^3 \operatorname{len}(M) + \ell^3 \operatorname{len}(\ell)).$$

Compared to (4.1), we have essentially replaced the term $\ell^3 \operatorname{len}(M)^2$ by $\ell^2 \operatorname{len}(M)^2 + \ell^3 \operatorname{len}(M)$. This is a significant improvement: for example, if $\operatorname{len}(M) \approx \ell$, then the running time of the original algorithm is $O(\ell^5)$, while the running time of the modular algorithm is $O(\ell^4)$.

Exercise 4.14 Apply the ideas above to the problem of computing the product of two polynomials whose coefficients are large integers. First, determine the running time of the "obvious" algorithm for multiplying two such polynomials, then design and analyze a "modular" algorithm. □

4.5 Rational Reconstruction and Applications

We next state a theorem whose immediate utility may not be entirely obvious, but we quickly follow up with several very neat applications. The general problem we consider here, called *rational reconstruction*, is as follows. Suppose that there is some rational number $\hat{y}$ that we would like to get our hands on, but the only information we have about $\hat{y}$ is the following:

- First, suppose that we know that $\hat{y}$ may be expressed as r/t for integers r, t, with $|r| \leq r^*$ and $|t| \leq t^*$ — we do not know r, t, but we do know the bounds r^*, t^*.
- Second, suppose that we know integers y, n such that

  $$r \equiv ty \pmod{n},$$

  where r, t are the unknown integers above.

It turns out that if n is sufficiently large relative to the bounds r^*, t^*, then we can virtually "pluck" $\hat{y}$ out of the Extended Euclidean Algorithm applied to n and y.

Theorem 4.15 *Let r^*, t^*, n, y be integers such that $r^* > 0$, $t^* > 0$, $n \geq 4r^*t^*$, and $0 \leq y < n$. Suppose we run the Extended Euclidean Algorithm with inputs $a := n$ and $b := y$. Then, adopting the notation of Theorem 4.5, the following hold:*

1. *There exists a unique index i, with $1 \leq i \leq \ell + 1$, such that $r_i \leq 2r^* < r_{i-1}$, and for this i, $t_i \neq 0$; let $r' := r_i$, $s' := s_i$, and $t' := t_i$.*
2. *Furthermore, for any integers r, s, t such that*

   $$r = sn + ty, \quad |r| \leq r^*, \quad 0 < |t| \leq t^*, \tag{4.6}$$

   we have

   $$r = r'\alpha, \ s = s'\alpha, \ t = t'\alpha,$$

   for some non-zero integer α.

Proof. By hypothesis, $2r^* < n = r_0$. Moreover, since $r_0, \ldots, r_\ell, r_{\ell+1} = 0$ is a decreasing sequence, and $1 = |t_1|, |t_2|, \ldots, |t_{\ell+1}|$ is a non-decreasing sequence, the first statement of the theorem is clear.

Now let i be defined as in the first statement of the theorem. Also, let r, s, t be as in (4.6).

From part (vii) of Theorem 4.5, we have

$$|t_i| \leq \frac{n}{r_{i-1}} < \frac{n}{2r^*}.$$

From the equalities $r_i = s_i n + t_i y$ and $r = sn + ty$, we have the two congruences:

$$\begin{aligned} r &\equiv ty \pmod{n}, \\ r_i &\equiv t_i y \pmod{n}. \end{aligned}$$

Subtracting t_i times the first from t times the second, we obtain

$$rt_i \equiv r_i t \pmod{n}.$$

This says that n divides $rt_i - r_i t$; however, using the bounds $|r| \leq r^*$, $|t_i| < n/(2r^*)$, $|r_i| \leq 2r^*$, $|t| \leq t^*$, and $4r^*t^* \leq n$, we obtain (verify)

$$|rt_i - r_i t| \leq |rt_i| + |r_i t| < n.$$

Since n divides $rt_i - r_i t$ and $|rt_i - r_i t| < n$, the only possibility is that

$$rt_i - r_i t = 0. \tag{4.7}$$

Now consider the two equations:

$$\begin{aligned} r &= sn + ty \\ r_i &= s_i n + t_i y. \end{aligned}$$

Subtracting t_i times the first from t times the second, and using the identity (4.7), we obtain $n(st_i - s_i t) = 0$, and hence

$$st_i - s_i t = 0. \tag{4.8}$$

From (4.8), we see that $t_i \mid s_i t$, and since from part (iii) of Theorem 4.5, we know that $\gcd(s_i, t_i) = 1$, we must have $t_i \mid t$. So $t = t_i \alpha$ for some α, and we must have $\alpha \neq 0$ since $t \neq 0$. Substituting $t_i \alpha$ for t in equations (4.7) and (4.8) yields $r = r_i \alpha$ and $s = s_i \alpha$. That proves the second statement of the theorem. □

4.5.1 Application: Chinese Remaindering with Errors

One interpretation of the Chinese Remainder Theorem is that if we "encode" an integer z, with $0 \leq z < n$, as the sequence $(a_1, \ldots, a_k)$, where $a_i = z \bmod n_i$, then we can efficiently recover z from this encoding. Here, of course, $n = n_1 \cdots n_k$ where the n_i's are pairwise relatively prime.

But now suppose that Alice encodes z as $(a_1, \ldots, a_k)$, and sends this encoding to Bob; however, during the transmission of the encoding, some (but hopefully not too many) of the a_i's may be corrupted. The question is, can Bob still efficiently recover the original z from its corrupted encoding?

To make the problem more precise, suppose that the original, correct encoding of z is $(a_1, \ldots, a_k)$, and the corrupted encoding is $(\tilde{a}_1, \ldots, \tilde{a}_k)$, where we shall assume that at most ℓ of the $\tilde{a}_i$'s differ from the corresponding a_i's.

Of course, if Bob hopes to recover z, we need to build some redundancy into the system; that is, we must require that $0 \leq z \leq Z$ for some Z that is somewhat smaller than n. Now, if Bob knew the positions where the errors actually occurred, and if the product of the n_i's at the non-error positions exceed Z, then Bob could simply discard the errors, and reconstruct z by applying the Chinese Remainder Theorem to the a_i's and n_i's at the non-error positions. However, in general, Bob will not know *a priori* the positions of the errors, and so this approach will not work.

Despite these apparent difficulties, Theorem 4.15 may be used to solve the problem quite easily, as follows. Let us suppose that $n_1, \ldots, n_k$ are arranged in decreasing order, and let us set $P := n_1 \cdots n_\ell$; that is, P is the product of the ℓ largest n_i's, and in particular, any product of any ℓ of the n_i's is at most P. Further, let us assume that $n \geq 4P^2Z$.

Now, suppose Bob obtains the corrupted enconding $(\tilde{a}_1, \ldots, \tilde{a}_k)$. Here is what Bob does to recover z:

1. Apply the Chinese Remainder Theorem, obtaining an integer y, with $0 \leq y < n$ and $y \equiv \tilde{a}_i \pmod{n}$ for $1 \leq i \leq k$.

2. Run the Extended Euclidean Algorithm on $a := n$ and $b := y$, and let r', t' be the values obtained from Theorem 4.15 applied with $r^* := XP$ and $t^* := P$.

3. If $t' \mid r'$, output r'/t'; otherwise, output "error."

We claim that the above procedure outputs z, assuming the number of errors is at most ℓ. To see this, let t be the product of the n_i's for those values of i where an error occurred. Now, assuming at most ℓ errors occurred, we have $1 \leq t \leq P$. Also, let $r := tz$, and note that $0 \leq r \leq XP$. We claim that

$$r \equiv ty \pmod{n}. \tag{4.9}$$

To show that (4.9) holds, it suffices to show that

$$tz \equiv ty \pmod{n_i} \tag{4.10}$$

for all $1 \leq i \leq k$. To show this, consider first an index i at which no error occurred, so that $a_i = \tilde{a}_i$. Then $tz \equiv ta_i \pmod{n_i}$ and $ty \equiv t\tilde{a}_i \equiv ta_i \pmod{n_i}$, and so (4.10) holds for this i. Next, consider an index i for which an error occurred. Then by construction, $tz \equiv 0 \pmod{n_i}$ and $ty \equiv 0 \pmod{n_i}$, and so (4.10) holds for this i. Thus, (4.9) holds, from which it follows that the values r', t' obtained from Theorem 4.15 satisfy

$$\frac{r'}{t'} = \frac{r}{t} = \frac{tz}{t} = z.$$

One easily checks that both the procedures to encode and decode a value z run in time $O(\mathrm{len}(n)^2)$. If one wanted a practical implementation, one might choose the n_i's to be, say, 16-bit primes, so that the encoding of a value z consisted of a sequence of k 16-bit words.

The above scheme is an example of an *error correcting code*, and is actually the integer analog of a *Reed-Solomon code*.

4.5.2 Application: recovering fractions from their decimal expansion

Suppose Alice knows a rational number $z = s/t$, where s and t are integers with $0 \leq s < t$, and tells Bob some of the high order digits in the decimal expansion of z. Can Bob determine z? The answer is yes, provided Bob knows an upper bound M on t, and provided Alice gives Bob enough digits. Of course, from grade school, Bob knows that the decimal expansion of z is ultimately periodic, and that given enough digits of z so as to include the periodic part, he can recover z; however, this technique is quite useless in practice, as the length of the period can be huge — $\Theta(M)$ in the worst case. The method we discuss here requires only $O(\mathrm{len}(M))$ digits.

To be a bit more general, suppose that Alice gives Bob the high-order k digits in the d-ary expansion of z, for some base $d > 1$. Now, we can express z in base d as

$$z = z_1 d^{-1} + z_2 d^{-2} + z_3 d^{-3} + \cdots,$$

and the sequence of digits $z_1, z_2, z_3, \ldots$ is uniquely determined if we require that the sequence does not terminate with an infinite run of $(d-1)$-digits. Suppose Alice gives Bob the first k digits $z_1, \ldots, z_k$. Then

$$y := \sum_{i=1}^{k} z^i d^{k-1-i} = \lfloor z d^k \rfloor.$$

Let us define $n := d^k$, so that $y = \lfloor zn \rfloor$.

Now, if n is much smaller than M^2, the number z is not even uniquely determined by y, since there are $\Omega(M^2)$ distinct rational numbers of the form s/t, with $0 \leq s < t \leq M$ (see Exercise 1.20). However, if $n \geq 4M^2$, then not only is z uniquely determined by y, but using Theorem 4.15, we can compute it as follows:

1. Run the Extended Euclidean Algorithm on inputs $a := n$ and $b := y$, and let let s', t' be as in Theorem 4.15, using $r^* := t^* := M$.

2. Output s', t'.

We claim that $z = -s'/t'$.

To prove this, let $z = s/t$ as above, and note that by definition

$$\frac{s}{t} = \frac{y}{n} + w, \tag{4.11}$$

where $0 \leq w < 1/n$. Clearing denominators, we see that

$$sn = ty + wnt.$$

Thus we see that $r := wnt$ is an integer, and moreover,

$$r = sn - ty \text{ and } 0 \leq r < t \leq t^*.$$

It follows that the integers s', t' from Theorem 4.15 satisfy $s = s'\alpha$ and $-t = t'\alpha$ for some non-zero integer α. Thus, $s'/t' = -s/t$, which proves the claim.

We may further observe that since the extended Euclidean algorithm guarantees that $\gcd(s', t') = 1$, not only do we obtain z, but we obtain z expressed as a fraction in lowest terms.

It is clear that the running time of this algorithm is $O(\text{len}(n)^2)$.

Example 4.16 Alice is chooses numbers $0 \leq s < t \leq 1000$, and tells Bob the high order 7 digits y in the decimal expansion of $z := s/t$, from which Bob should be able to compute z. Suppose $s = 511$ and $t = 710$. Then $s/t \approx 0.71971830985915492957$, and so $y = 7197183$. We also have $n = 10^7$. Running the Extended Euclidean Algorithm on inputs $a := n$ and $b := y$, Bob obtains the following data:

i	r_i	q_i	s_i	t_i
0	10000000		1	0
1	7197183	1	0	1
2	2802817	2	1	-1
3	1591549	1	-2	3
4	1211268	1	3	-4
5	380281	3	-5	7
6	70425	5	18	-25
7	28156	2	-95	132
8	14113	1	208	-289
9	14043	1	-303	421
10	70	200	511	-710
11	43	1	-102503	142421
12	27	1	103014	-143131
13	16	1	-205517	285552
14	11	1	308531	-428683
15	5	2	-514048	714235
16	1	5	1336627	-1857153
17	0		-7197183	10000000

The first r_i which falls below the threshold 2000 is at $i = 10$, and we read off $s' = 511$ and $t' = -710$, from which Bob obtains $z = -s'/t' = 511/710$. □

Exercise 4.17 Show that given integers s, t, k, with $0 \leq s < t$, and $k > 0$, we can compute the kth digit in the decimal expansion of s/t in time $O(\text{len}(k)\,\text{len}(t)^2)$. □

4.5.3 Applications to symbolic algebra

Without going into many of the details, we discuss how Theorem 4.15 is used in some algorithms in symbolic algebra. The discussion in the section is deliberately quite sketchy — we just want to give the reader the flavor of this application.

Suppose, for example, that we want to find the solution v to a matrix-vector equation

$$Mv = w,$$

where we are given a non-singular square integer matrix M and the integer vector w. Now, the solution vector v has rational entries, and by Cramer's Rule, each entry in v can be expressed as a fraction with denominator $\det(M)$. We stress that we want to compute the exact solution v, and not some floating point approximation to it. Now, we could solve for v directly using Gaussian elimination; however, the intermediate quantities computed by that algorithm would be rational numbers whose numerators and denominators might get somewhat large, leading to a rather lengthy computation (however, it is possible to show that the overall running time is still polynomial in the input length).

Another approach is to compute a solution vector modulo n, where n is a prime or a power of a prime that does not divide $\det(M)$. We will not go into the details of how one computes such a solution, but suffice it to say that such a computation will be much more efficient than one over the rational numbers, because all of the intermediate quantities will be reduced modulo n. If y is one of the entries in the solution vector modulo n, then we will have $r \equiv ty \pmod{n}$, where r/t is the corresponding entry in the rational solution vector. Using well-known bounds on r and t (in terms of the inputs M and w), we can make sure that n is large enough relative to these bounds so that Theorem 4.15 yields r and t.

4.6 Notes

The Euclidean algorithm as we have presented it here is not the fastest known algorithm for computing greatest common divisors. The asymptotically fastest known algorithm for computing the greatest common divisor of two numbers of bit length at most ℓ runs in time $O(\ell\,\text{len}(\ell))$ on a RAM, and the smallest boolean circuits are of size $O(\ell\,\text{len}(\ell)^2\,\text{len}(\text{len}(\ell)))$. The same complexity results also hold for the extended Euclidean algorithm, as well as Chinese remaindering and rational reconstruction. See Chapter 9 of Crandall and Pomerance [23] (and also the discussion in §3.5).

Experience suggests that such fast algorithms for greatest common divisors are not of much practical value, unless the integers involved are *very* large — at least several tens

of thousands of bits in length. The extra "log" factor and the rather large multiplicative constants seem to slow things down too much.

Our exposition of Theorem 4.15 is loosely based on Bach [10]. A somewhat "tighter" result is proved, with significantly more effort, by Wang, Guy, and Davenport [76]. However, for most practical purposes, the result proved here is just as good. The application of Euclid's algorithm to computing a rational number from the first digits of its decimal expansion was observed by Blum, Blum, and Shub [15].

Chapter 5

The Distribution of Primes

This chapter concerns itself with the question: how many primes are there? This chapter has a bit more of an "analytical" flavor than other chapters in this text. However, we shall not make use of any mathematics beyond that of elementary calculus.

We first state a (truly) classical result:

Theorem 5.1 *There are infinitely many primes.*

Proof. Suppose that there were only finitely many primes, call them $p_1, \ldots, p_k$. Then set $x = 1 + \prod_{i=1}^{k} p_i$, and consider any prime p that divides x. Clearly, p cannot equal any of the p_i, since if it did, we would would have $p \mid 1$, which is impossible. Therefore, the prime p is not among $p_1, \ldots, p_k$, which contradicts our assumption that these are the only primes. □

5.1 Chebyshev's Theorem on the Density of Primes

In addition to the fact that there are infinitely many primes, one would like to know how "dense" prime numbers are. The natural way of measuring the density of primes is to count the number of primes up to a bound x, where x is a real number. For a real number $x \geq 0$, the function $\pi(x)$ is defined to be the number of primes up to x. Thus, $\pi(1) = 0$, $\pi(2) = 1$, $\pi(7.5) = 4$, and so on. The function π is an example of a "step function," i.e., a function that changes values only at a discrete set of points. It might seem more natural to define π only on the integers, but it is the tradition (and there are some technical benefits) to defining it over the real numbers.

Let us first take a look at some values of $\pi(x)$. Table 5.1 shows values of $\pi(x)$ for $x = 10^{3i}$, for $i = 1, \ldots, 6$. The third column of this table shows the value of $x/\pi(x)$ (to five decimal places). One can see that the differences between successive rows of this third column are roughly the same, which suggests that the function $x/\pi(x)$ grows logarithmically in x. Indeed, as $\log(10^3) \approx 6.9$, it would not be unreasonable to guess that $x/\pi(x) \approx \log x$, i.e., $\pi(x) \approx x/\log x$.

x	$\pi(x)$	$x/\pi(x)$
10^3	168	5.95238
10^6	78498	12.73918
10^9	50847534	19.66664
10^{12}	37607912018	26.59015
10^{15}	29844570422669	33.50693
10^{18}	24739954287740860	40.42045

Table 5.1: Some values of $\pi(x)$

The following theorem is a first — and important — step towards making the above guess-work more rigorous:

Theorem 5.2 (Chebyshev's Theorem) *We have*

$$\pi(x) = \Theta(x/\log x).$$

It is not too difficult to prove this theorem, which we now proceed to do in several steps. Recalling that $\nu_p(n)$ denotes the power to which a prime p divides an integer n, we begin with the following observation:

Theorem 5.3 *Let n be a positive integer. For any prime p, we have*

$$\nu_p(n!) = \sum_{k\geq 1} \lfloor n/p^k \rfloor.$$

Proof. This follows immediately from the observation that the numbers $1, 2, \ldots, n$ include exactly $\lfloor n/p \rfloor$ multiples of p, $\lfloor n/p^2 \rfloor$ multiples of p^2, and so on (see Exercise 1.7). □

The following theorem gives a lower bound on $\pi(x)$.

Theorem 5.4 *$\pi(n) \geq (\log 2/2)n/\log n$ for all integers $n \geq 2$.*

Proof. For positive integer m, let

$$N := \binom{2m}{m} = \frac{(2m)!}{(m!)^2}.$$

Note that

$$N = \left(\frac{m+1}{1}\right)\left(\frac{m+2}{2}\right)\cdots\left(\frac{m+m}{m}\right),$$

from which it is clear that $N \geq 2^m$ and that N is divisible only by primes p not exceeding $2m$. Applying Theorem 5.3 to the identity $N = (2m)!/(m!)^2$, we have

$$\nu_p(N) = \sum_{k\geq 1} (\lfloor 2m/p^k \rfloor - 2\lfloor m/p^k \rfloor).$$

Each term in this sum is either 0 or 1 (see Exercise 1.6), and for $k > \log(2m)/\log p$, each term is zero. Thus, $\nu_p(N) \leq \log(2m)/\log p$.

So we have

$$\pi(2m)\log(2m) = \sum_{p \leq 2m} \frac{\log(2m)}{\log p} \log p \geq \sum_{p \leq 2m} \nu_p(N) \log p = \log N \geq m \log 2.$$

Therefore,

$$\pi(2m) \geq (\log 2/2)(2m)/\log(2m).$$

That proves the theorem for even n. Now consider odd $n \geq 3$, so $n = 2m-1$ for $m \geq 2$. Since the function $x/\log x$ is increasing for $x \geq 3$ (verify), and since $\pi(2m-1) = \pi(2m)$ for $m \geq 2$, we have

$$\pi(2m-1) = \pi(2m) \geq (\log 2/2)(2m)/\log(2m) \geq (\log 2/2)(2m-1)/\log(2m-1).$$

That proves the theorem for odd n. □

To obtain a corresponding upper bound for $\pi(x)$, we introduce an auxiliary function, called Chebyshev's ϑ-function:

$$\vartheta(x) := \sum_{p \leq x} \log p,$$

where the sum is over all primes p up to x. The next theorem relates $\pi(x)$ and $\vartheta(x)$.

Theorem 5.5 *We have:*

$$\pi(x) \sim \frac{\vartheta(x)}{\log x}.$$

Proof. On the one hand, we have

$$\vartheta(x) = \sum_{p \leq x} \log p \leq \log x \sum_{p \leq x} 1 = \pi(x) \log x.$$

So we have

$$\pi(x) \geq \frac{\vartheta(x)}{\log x}.$$

On the other hand, for every $\epsilon > 0$, we have

$$\begin{aligned} \vartheta(x) &\geq \sum_{x^{1-\epsilon} < p \leq x} \log p \geq (1-\epsilon)\log x \sum_{x^{1-\epsilon} < p \leq x} 1 = (1-\epsilon)\log x\,(\pi(x) - \pi(x^{1-\epsilon})) \\ &\geq (1-\epsilon)\log x\,(\pi(x) - x^{1-\epsilon}). \end{aligned}$$

Hence,

$$\pi(x) \leq x^{1-\epsilon} + \frac{\vartheta(x)}{(1-\epsilon)\log x}.$$

Since by the previous theorem, the term $x^{1-\epsilon}$ is $o(\pi(x))$, we have for all sufficiently large x (depending on ϵ), $x^{1-\epsilon} \leq \epsilon\pi(x)$, and so

$$\pi(x) \leq \frac{\vartheta(x)}{(1-\epsilon)^2 \log x}.$$

By making ϵ sufficiently small, we can make $1/(1-\epsilon)^2$ arbitrarily close to 1, and the theorem follows. □

Theorem 5.6 *$\vartheta(x) < 2x \log 2$ for all real numbers $x \geq 1$.*

Proof. It suffices to prove that $\vartheta(n) < 2n \log 2$ for integers $n \geq 1$, since then $\vartheta(x) = \vartheta(\lfloor x \rfloor) < 2\lfloor x \rfloor \log 2 \leq 2x \log 2$.

For positive integer m, let

$$M := \binom{2m+1}{m} = \frac{(2m+1)!}{m!(m+1)!}.$$

One sees that M is divisible by all primes p with $m+1 < p \leq 2m+1$. As M occurs twice in the binomial expansion of $(1+1)^{2m+1}$, one sees that $M < 2^{2m+1}/2 = 2^{2m}$. It follows that

$$\vartheta(2m+1) - \vartheta(m+1) = \sum_{m+1<p\leq 2m+1} \log p \leq \log M < 2m \log 2.$$

We now prove the theorem by induction. For $n = 1$ and $n = 2$, the theorem is trivial. Now let $n > 2$. If n is even, then we have

$$\vartheta(n) = \vartheta(n-1) < 2(n-1)\log 2 < 2n \log 2.$$

If $n = 2m+1$ is odd, then we have

$$\vartheta(n) = \vartheta(2m+1) - \vartheta(m+1) + \vartheta(m+1) < 2m \log 2 + 2(m+1)\log 2 < 2n \log 2.$$

□

Another way of stating the above theorem is:

$$\prod_{p \leq x} < 4^x.$$

Theorem 5.2 follows immediately from Theorems 5.4, 5.5 and 5.6. Note that we have also proved:

Theorem 5.7 *We have*

$$\vartheta(x) = \Theta(x).$$

Exercise 5.8 If p_n denotes the nth prime, show that $p_n = \Theta(n \log n)$. □

Exercise 5.9 For integer $n > 1$, let $\omega(n)$ denote the number of distinct primes dividing n. Show that $\omega(n) = O(\log n / \log\log n)$. □

Exercise 5.10 Show that for positive integers a and b,

$$\binom{a+b}{b} \geq 2^{\min(a,b)}.$$

□

5.2 Bertrand's Postulate

Suppose we want to know how many primes there are of a given bit length, or more generally, how many primes there are between m and $2m$ for a given integer m. Neither the statement, nor the proof, of Chebyshev's Theorem imply that there are *any* primes between m and $2m$, let alone a useful density estimate of such primes.

Bertrand's Postulate is the assertion that for all positive integers m, there exists a prime between m and $2m$. We shall in fact prove a stronger result, namely, that not only is there one prime, but the number of primes between m and $2m$ is $\Omega(m/\log m)$.

Theorem 5.11 (Bertrand's Postulate) *For any integer $m \geq 2$, we have*

$$\pi(2m) - \pi(m) > \frac{m}{3\log(2m)}.$$

The proof uses Theorem 5.6, along with a more careful re-working of the proof of Theorem 5.4. The theorem is clearly true for $m = 2$, so we may assume that $m \geq 3$. As in the proof of the Theorem 5.4, define $N := \binom{2m}{m}$, and recall that N is divisible only by primes strictly less than $2m$, and that we have the identity

$$\nu_p(N) = \sum_{k \geq 1} (\lfloor 2m/p^k \rfloor - 2\lfloor m/p^k \rfloor), \tag{5.1}$$

where each term in the sum is either 0 or 1. We can characterize the values $\nu_p(M)$ a bit more precisely, as follows:

Lemma 5.12 *With m and N as above, for all primes p, we have*

$$p^{\nu_p(N)} \leq 2m; \tag{5.2}$$

$$\text{if } p > \sqrt{2m}, \text{ then } \nu_p(N) \leq 1; \tag{5.3}$$

$$\text{if } 2m/3 < p \leq m, \text{ then } \nu_p(N) = 0; \tag{5.4}$$

$$\text{if } m < p < 2m, \text{ then } \nu_p(N) = 1. \tag{5.5}$$

Proof.

For (5.2), all terms with $k > \log(2m)/\log p$ in (5.1) vanish, and hence $\nu_p(N) \leq \log(2m)/\log p$, from which it follows that $p^{\nu_p(N)} \leq 2m$.

(5.3) follows immediately from (5.2).

For (5.4), if $2m/3 < p \leq m$, then $2m/p < 3$, and we must also have $p \geq 3$, since $p = 2$ implies $m < 3$. We have $p^2 > p(2m/3) = 2m(p/3) \geq 2m$, and hence all terms with $k > 1$ in (5.1) vanish. The term with $k = 1$ also vanishes, since $1 \leq m/p < 3/2$, from which it follows that $2 \leq 2m/p < 3$, and hence $\lfloor m/p \rfloor = 1$ and $\lfloor 2m/p \rfloor = 2$.

For (5.5), if $m < p < 2m$, it follows that $1 < 2m/p < 2$, so $\lfloor 2m/p \rfloor = 1$. Also, $m/p < 1$, so $\lfloor m/p \rfloor = 0$. It follows that term with $k = 1$ in (5.1) is 1, and it is clear that $2m/p^k < 1$ for all $k > 1$, and so all the other terms vanish. □

We need one more technical fact, namely, a somewhat better lower bound on N than that used in the proof of Theorem 5.4:

Lemma 5.13 *With m and N as above, we have*

$$N > 4^m/(2m). \tag{5.6}$$

Proof. We prove this for all $m \geq 2$ by induction on m. One checks by direct calculation that it holds for $m = 2$. For $m > 2$, by induction we have

$$\binom{2(m+1)}{m+1} = 2\frac{2m+1}{m+1}\binom{2m}{m} > \frac{(2m+1)4^m}{m(m+1)} = \frac{2m+1}{2m}\frac{4^{m+1}}{2(m+1)} > \frac{4^{m+1}}{2(m+1)}.$$

□

We now have the necessary technical ingredients to prove Theorem 5.11. Define

$$P_m := \prod_{m<p<2m} p,$$

and define Q_m so that

$$N = Q_m P_m.$$

By (5.4) and (5.5), we see that

$$Q_m = \prod_{p \leq 2m/3} p^{\nu_p(N)}.$$

Moreover, by (5.3), $\nu_p(N) > 1$ for at most those $p \leq \sqrt{2m}$, so there are at most $\sqrt{2m}$ such primes, and by (5.2), the contribution of each such prime to the above product is at most $2m$. Combining this with Theorem 5.6, we obtain

$$Q_m < (2m)^{\sqrt{2m}} \cdot 4^{2m/3}.$$

We now apply (5.6), obtaining

$$P_m = NQ_m^{-1} > 4^m (2m)^{-1} Q_m^{-1} > 4^{m/3} (2m)^{-(1+\sqrt{2m})}.$$

It follows that

$$\begin{aligned} \pi(2m) - \pi(m) &\geq \log P_m / \log(2m) > \frac{m \log 4}{3 \log(2m)} - (1 + \sqrt{2m}) \\ &= \frac{m}{3 \log(2m)} + \frac{m(\log 4 - 1)}{3 \log(2m)} - (1 + \sqrt{2m}). \end{aligned} \tag{5.7}$$

Clearly, the term $(m(\log 4 - 1))/(3 \log(2m))$ in (5.7) dominates the term $1 + \sqrt{2m}$, and so Theorem 5.11 holds for all sufficiently large m. Indeed, a simple calculation shows that (5.7) implies the theorem for $m \geq 13,000$, and one can verify by brute force (with the aid of a computer) that the theorem holds for $m < 13,000$.

5.3 The Sum $\sum_{p \le x} 1/p$

Our next goal is to prove the following theorem, which turns out to have a number of applications.

Theorem 5.14 *We have*

$$\sum_{p \le x} \frac{1}{p} = \log \log x + O(1).$$

The proof of this theorem, while not difficult, is a bit technical, and we proceed in several steps.

Theorem 5.15 *We have*

$$\sum_{p \le x} \frac{\log p}{p} = \log x + O(1).$$

Proof. Let $n = \lfloor x \rfloor$. By Theorem 5.3, we have

$$\log(n!) = \sum_{p \le n} \sum_{k \geq 1} \lfloor n/p^k \rfloor \log p = \sum_{p \le n} \lfloor n/p \rfloor \log p + \sum_{k \geq 2} \sum_{p \le n} \lfloor n/p^k \rfloor \log p.$$

We next show that the last sum is $O(n)$. We have

$$\begin{aligned} \sum_{p \le n} \log p \sum_{k \geq 2} \lfloor n/p^k \rfloor &\leq n \sum_{p \le n} \log p \sum_{k \geq 2} p^{-k} \\ &= n \sum_{p \le n} \frac{\log p}{p^2} \cdot \frac{1}{1 - 1/p} = n \sum_{p \le n} \frac{\log p}{p(p-1)} \\ &\leq n \sum_{i \geq 2} \frac{\log k}{k(k-1)} = O(n). \end{aligned}$$

Thus, we have shown that

$$\log(n!) = \sum_{p \le n} \lfloor n/p \rfloor \log p + O(n).$$

Further, since $\lfloor n/p \rfloor = n/p + O(1)$, applying Theorem 5.6, we have

$$\log(n!) = \sum_{p \le n} (n/p) \log p + O(\sum_{p \le n} \log p) + O(n) = n \sum_{p \le n} \frac{\log p}{p} + O(n). \tag{5.8}$$

We can also estimate $\log(n!)$ using a little calculus (see §A.7). We have

$$\log(n!) = \sum_{k=1}^{n} \log k = \int_1^n \log t \, dt + O(\log n) = n \log n - n + O(\log n). \tag{5.9}$$

Combining (5.8) and (5.9), and noting that $\log x - \log n = o(1)$, we obtain

$$\sum_{p \le x} \frac{\log p}{p} = \log n + O(1) = \log x + O(1),$$

which proves the theorem. □

We shall also need the following theorem, which is a very useful tool in its own right:

Theorem 5.16 (Abel's Identity) *Suppose that $c_k, c_{k+1}, \ldots$ is a sequence of numbers, that*

$$C(t) := \sum_{k \le i \le t} c_i,$$

and that $f(t)$ has a continuous derivative $f'(t)$ on the interval $[k, x]$. Then

$$\sum_{k \le i \le x} c_i f(i) = C(x) f(x) - \int_k^x C(t) f'(t) \, dt.$$

Note that since $C(t)$ is a step function, the integrand $C(t)f'(t)$ piece-wise continuous on $[k, x]$, and hence the integral is well defined (see §A.8).

Proof. Let $n = \lfloor x \rfloor$. We have

$$\begin{aligned}
\sum_{i=k}^{n} c_i f(i) &= C(k)f(k) + [C(k+1) - C(k)]f(k+1) + \cdots + [C(n) - C(n-1)]f(n) \\
&= C(k)[f(k) - f(k+1)] + \cdots + C(n-1)[f(n-1) - f(n)] + C(n)f(n) \\
&= C(k)[f(k) - f(k+1)] + \cdots + C(n-1)[f(n-1) - f(n)] + \\
&\quad C(n)[f(n) - f(x)] + C(x)f(x).
\end{aligned}$$

Observe that for $k \le i < n$, we have $C(t) = C(i)$ for $i \le t < i+1$, and so

$$C(i)[f(i) - f(i+1)] = -\int_i^{i+1} C(t)f'(t)\,dt;$$

likewise,

$$C(n)[f(n) - f(x)] = -\int_n^x C(t)f'(t)\,dt,$$

from which the theorem directly follows. □

Proof of Theorem 5.14. For $i \ge 2$, set $c_i = \log i/i$ if i is prime, and 0 otherwise. By Theorem 5.15, we have

$$C(t) := \sum_{2 \le i \le t} c_i = \sum_{p \le t} \frac{\log p}{p} = \log t + O(1).$$

Applying Theorem 5.16 with $f(t) = 1/\log t$, we obtain

$$\begin{aligned}
\sum_{p \le x} \frac{1}{p} &= \frac{C(x)}{\log x} + \int_2^x \frac{C(t)}{t(\log t)^2} dt \\
&= \Big(1 + O(1/\log x)\Big) + \Big(\int_2^x \frac{dt}{t \log t} + O(\int_2^x \frac{dt}{t(\log t)^2})\Big) \\
&= 1 + O(1/\log x) + (\log\log x - \log\log 2) + O(1/\log 2 - 1/\log x).
\end{aligned}$$

□

Using Theorem 5.14, we can easily show the following:

Theorem 5.17 (Mertens' Theorem) *Let $U(x) := \prod_{p \le x}(1 - 1/p)$, where the product is over all primes p up to x. Then*

$$U(x) = \Theta(1/\log x).$$

Proof. From calculus, we have

$$-\log(1 - 1/p) = \sum_{i \ge 1} \frac{1}{ip^i} < \frac{1}{p} + \frac{1}{2}\sum_{i \ge 2} \frac{1}{p^i} = \frac{1}{p} + \frac{1}{2p(p-1)}.$$

It follows that

$$-\frac{1}{2p(p-1)} < \frac{1}{p} + \log(1 - 1/p) < 0.$$

Moreover, $\sum_{p \le x} \frac{1}{p(p-1)} \le \sum_{i \ge 2} \frac{1}{i(i-1)} < \infty$. Therefore,

$$\sum_{p \le x} \frac{1}{p} + \log U(x) = O(1).$$

From this, and from Theorem 5.14, we obtain

$$\log\log x + \log U(x) = O(1).$$

Now exponentiate both sides, and the theorem follows. □

Exercise 5.18 Let $\omega(n)$ be the number of distinct prime factors of n, and define $\overline{\omega}(x) = \sum_{n\le x}\omega(n)$, so that $\overline{\omega}(x)/x$ represents the "average" value of ω. First, show that $\overline{\omega}(x) = \sum_{p\le x}\lfloor x/p\rfloor$. From this, show that $\overline{\omega}(x) \sim x\log\log x$. □

Exercise 5.19 Define the sequence of numbers $n_1, n_2, \ldots$, where n_k is the product of all the primes up to k. Show that as $k\to\infty$, $\phi(n_k) = O(n_k/\log\log n_k)$. Hint: you will want to use Mertens' Theorem, and also Theorem 5.7. □

Exercise 5.20 The previous exercise showed that $\phi(n)$ could be as small as (about) $n/\log\log n$ for infinitely many n. Show that this is the "worst case," in the sense that $\phi(n) = \Omega(n/\log\log n)$ as $n\to\infty$. □

Exercise 5.21 Show that for any positive integer constant k,

$$\int_2^x \frac{dt}{(\log t)^k} = \frac{x}{(\log x)^k} + O\left(\frac{x}{(\log x)^{k+1}}\right).$$

□

Exercise 5.22 Use Chebyshev's Theorem and Abel's Identity to show that

$$\sum_{p\le x}\frac{1}{\log p} = \frac{\pi(x)}{\log x} + O(x/(\log x)^3).$$

□

Exercise 5.23 Use Chebyshev's Theorem and Abel's Identity to prove a stronger version of Theorem 5.5:

$$\vartheta(x) = \pi(x)\log x + O(x/\log x).$$

□

Exercise 5.24 Define

$$U_2(x) := \prod_{2<p\le x}(1-2/p),$$

where the product is over all primes between 3 and x. Show that

$$U_2(x) = \Theta(1/(\log x)^2).$$

□

Exercise 5.25 Show that if $\pi(x) \sim cx/\log x$ for some constant c, then we must have $c = 1$. Hint: use either Theorem 5.14 or 5.15. □

Exercise 5.26 Strengthen Theorem 5.14, showing that $\sum_{p \leq x} 1/p \sim \log\log x + A$ for some constant A. (Note: $A \approx 0.261497212847643$.) □

Exercise 5.27 Strengthen Mertens' Theorem, showing that $U(x) \sim B_1/(\log x)$ for some constant B_1. Hint: use the result from the previous exercise. (Note: $B_1 \approx 0.561459483566885$.) □

Exercise 5.28 Strengthen the result of Exercise 5.24, showing that $U_2(x) \sim B_2/(\log x)^2$ for some constant B_2. (Note: $B_2 \approx 0.832429065662$.) □

5.4 The Sieve of Eratosthenes

As an application of Theorem 5.14, consider the Sieve of Eratosthenes. This is an algorithm for generating all the primes up to a given bound k. It uses an array $A[2 \ldots k]$, and runs as follows.

for $n \leftarrow 2$ to k do $A[n] \leftarrow 1$
for $n \leftarrow 2$ to $\lfloor \sqrt{k} \rfloor$ do
    if $A[n] = 1$ then
        $i \leftarrow 2n$; while $i \leq k$ do { $A[i] \leftarrow 0$; $i \leftarrow i + n$ }

When the algorithm finishes, we have $A[n] = 1$ if and only if n is prime, for $2 \leq n \leq k$. This can easily be proven using the fact that a composite number n between 2 and k must be divisible by a prime that is at most $\sqrt{k}$, and by proving by induction on n that at the beginning of the nth iteration of the main loop, $A[i] = 0$ iff i is divisible by a prime less than n, for $n \leq i \leq k$. We leave the details of this to the reader.

We are more interested in the running time of the algorithm. To analyze the running time, we assume that all arithmetic operations take constant time; this is reasonable, since all the quantities computed in the algorithm are bounded by k, and we need to at least be able to index all entries of the array A, which has size k.

Every time we execute the inner loop of the algorithm, we perform $O(k/n)$ steps to clear the entries of A whose indices are multiples of n. Naively, we could bound the running time by a constant times

$$\sum_{n \leq \sqrt{k}} k/n,$$

which is $O(k \operatorname{len}(k))$, where we use a little calculus (see §A.7) to derive that

$$\sum_{n=1}^{\ell} 1/n = \int_1^{\ell} \frac{dy}{y} + O(1) \sim \log \ell.$$

However, the inner loop is executed only for prime values of n; thus, the running time is proportional to

$$\sum_{p \leq \sqrt{k}} k/p,$$

and so by Theorem 5.14 is $\Theta(k \operatorname{len}(\operatorname{len}(k)))$.

Exercise 5.29 Give a detailed proof of the correctness of the above algorithm. □

Exercise 5.30 One drawback of the above algorithm is its use of space: it requires an array of size k. Show how to modify the algorithm, without substantially increasing its running time, so that one can enumerate all the primes up to k, using an auxiliary array of size just $O(\sqrt{k})$. □

5.5 The Prime Number Theorem . . . and Beyond

In this section, we survey a number of theorems and conjectures related to the distribution of primes. This is a vast area of mathematical research, with a number very deep results. We shall be stating a number of theorems from the literature in this section without proof; while our intent is to keep the text as self contained as possible, and to avoid degenerating into "mathematical tourism," it nevertheless is a good idea to occasionally have a somewhat broader perspective. In the following chapters, we shall not make any critical use of the theorems in this section.

5.5.1 The Prime Number Theorem

The main theorem in the theory of the density of primes is the following.

Theorem 5.31 (Prime Number Theorem) *We have*

$$\pi(x) \sim x/\log x.$$

Proof. Literature — see §5.6. □

As we saw in Exercise 5.25, if $\pi(x)/(x/\log x)$ tends to a limit as $x \to \infty$, then the limit must be 1, so in fact the hard part of proving the Prime Number Theorem is to show that $\pi(x)/(x/\log x)$ does indeed tend to some limit.

One simple consequence of the Prime Number Theorem, together with Theorem 5.5, is the following:

Theorem 5.32 *We have*

$$\vartheta(x) \sim x.$$

Exercise 5.33 Using the Prime Number Theorem, show that if p_n denotes the nth prime, then $p_n \sim n \log n$. □

Exercise 5.34 Show that using the Prime Number Theorem, Theorem 5.11 (Bertrand's Postulate) can be strengthened (asymptotically) as follows: for all $\epsilon > 0$, there exist positive constants c and x_0, such that for all $x \geq x_0$, we have

$$\pi((1+\epsilon)x) - \pi(x) \geq c\frac{x}{\log x}.$$

□

Sometimes, it is useful to have explicit estimates for $\pi(x)$, as well as related functions, like $\vartheta(x)$ and the nth prime function p_n.

Theorem 5.35 *We have*

$$\frac{x}{\log x}\left(1 + \frac{1}{2\log x}\right) < \pi(x) < \frac{x}{\log x}\left(1 + \frac{3}{2\log x}\right), \quad \textit{for } x \geq 59;$$

$$n(\log n + \log\log n - 3/2) < p_n < n(\log n + \log\log -1/2), \quad \textit{for } n \geq 20;$$

$$x(1 - 1/(2\log x)) < \vartheta(x) < x(1 + 1/(2\log x)), \quad \textit{for } x \geq 563;$$

$$\log\log x + A - 1/(2(\log x)^2) < \sum_{p \leq x} 1/p < \log\log x + A + 1/(2(\log x)^2), \quad \textit{for } x \geq 286,$$

where A is the constant in Exercise 5.26;

$$\frac{B_1}{\log x}\left(1 - \frac{1}{2(\log x)^2}\right) < \prod_{p \leq x}\left(1 - \frac{1}{p}\right) < \frac{B_1}{\log x}\left(1 + \frac{1}{2(\log x)^2}\right), \quad \textit{for } x \geq 285,$$

where B_1 is the constant in Exercise 5.27.

Proof. Literature — see §5.6. □

5.5.2 The Error Term in the Prime Number Theorem

The Prime Number Theorem says that

$$|\pi(x) - x/\log x| \leq \delta(x),$$

where $\delta(x) = o(x/\log x)$. A natural question is: how small is "error term" $\delta(x)$? It turns out that:

Theorem 5.36 *We have*

$$\pi(x) = x/\log x + O(x/(\log x)^2).$$

x	$\pi(x)$	$\mathrm{li}(x)$	$x/\log x$
10^3	168	176.6	144.8
10^6	78498	78626.5	72382.4
10^9	50847534	50849233.9	48254942.4
10^{12}	37607912018	37607950279.8	36191206825.3
10^{15}	29844570422669	29844571475286.5	28952965460216.8
10^{18}	24739954287740860	24739954309690414.0	24127471216847323.8

Table 5.2: Values of $\pi(x)$, $\mathrm{li}(x)$, and $x/\log x$

The above bound on the error term is not very impressive. The reason is that $x/\log x$ is not really the best "simple" function that approximates $\pi(x)$. It turns out that a better approximation to $\pi(x)$ is the **logarithmic integral**, defined by

$$\mathrm{li}(x) := \int_2^x \frac{dt}{\log t}.$$

It is not hard to show (see Exercise 5.21) that

$$\mathrm{li}(x) = x/\log x + O(x/(\log x)^2).$$

Thus, $\mathrm{li}(x) \sim x/\log x \sim \pi(x)$. However, the error term in the approximation of $\pi(x)$ by $\mathrm{li}(x)$ is much better. This is illustrated numerically in Table 5.2 — notice how much better $\mathrm{li}(x)$ approximates $\pi(x)$ than does $x/\log x$; for example, at $x = 10^{18}$, $\mathrm{li}(x)$ approximates $\pi(x)$ with a relative error just under 10^{-9}, while $x/\log x$ approximates $\pi(x)$ with a relative error of about 0.025.

The sharpest proven result is the following:

Theorem 5.37 *Let $\kappa(x) := (\log x)^{3/5}(\log\log x)^{-1/5}$. Then for some $c > 0$, we have*

$$\pi(x) = \mathrm{li}(x) + O(xe^{-c\kappa(x)}).$$

Proof. Literature — see §5.6. □

Note that the error term $xe^{-c\kappa(x)}$ is $o(x/(\log x)^k)$ for every fixed $k \geq 0$. Also note that Theorem 5.36 follows directly from the above theorem and Exercise 5.21; it also follows from Theorem 5.35.

Although the above estimate on the error term in the approximation of $\pi(x)$ by $\mathrm{li}(x)$ is pretty good, it is conjectured that the actual error term is much smaller:

Conjecture 5.38 *For all $x \geq 2.01$, we have*

$$|\pi(x) - \mathrm{li}(x)| < x^{1/2}\log x.$$

Conjecture 5.38 is equivalent to a famous conjecture called the **Riemann Hypothesis**, which is an assumption about the location of the zeros of a certain function, called Riemann's "zeta" function. For real $s > 1$, this function is defined as

$$\zeta(s) := \sum_{n=1}^{\infty} \frac{1}{n^s}.$$

A simple, but important, connection between the zeta function and the theory of prime numbers is that for real $s > 1$, the following identity holds:

$$\zeta(s) = \prod_{p} (1 - p^{-s})^{-1}. \tag{5.10}$$

Proving this identity is straightforward, and the reader is invited to do so. Various closed-form evaluations of the zeta function are known, e.g.,

$$\zeta(2) = \pi^2/6, \quad \zeta(4) = \pi^4/90,$$

and more generally, $\zeta(n)$ for even integers n is known.

There is a certain way to extend the domain of definition of the zeta function to all complex numbers other than 1 (we cannot get into the details of how this is done here); the Riemann Hypothesis is the conjecture that the zeta function does not vanish at any complex points of the form $z = x + yi$, where $0 \leq x \leq 1$ and $x \neq 1/2$.

5.5.3 Primes in Arithmetic Progressions

The arithmetic progression of odd numbers $1, 3, 5, \ldots$ contains infinitely many primes. It is natural to ask if other arithmetic progressions do as well. An arithmetic progression with first term a and common difference d consists of all integers of the form

$$md + a, \quad m = 0, 1, 2, \ldots.$$

If d and a have a common factor $c > 1$, then every term in the progression is divisible by c, and so there can be no more than one prime in the progression. So a necessary condition for the existence of infinitely many primes p with $p \equiv a \pmod{d}$ is that $\gcd(d, a) = 1$. A famous theorem due to Dirichlet states that this is a sufficient condition as well.

Theorem 5.39 (Dirichlet's Theorem) *For any positive integer d and any integer a relatively prime to d, there are infinitely many primes p with $p \equiv a \pmod{d}$.*

Proof. Literature — see §5.6. □

We can also ask about the density of primes in arithmetic progressions. One might expect that for a fixed value of d, the primes are distributed in roughly equal measure among the $\phi(d)$ different residue classes $[a \bmod d]$ with $\gcd(a, d) = 1$. This is in fact the case. To formulate such assertions, we define $\pi(x; d, a)$ to be the number of primes p up to x with $p \equiv a \pmod{d}$.

Theorem 5.40 *Let $d > 0$ be fixed, and let a be relatively prime to d. Then*

$$\pi(x; d, a) \sim \frac{x}{\phi(d) \log x}.$$

Proof. Literature — see §5.6. □

The above theorem is only applicable in the case where d is fixed and $x \to \infty$. But what if we want an estimate on the number of primes p up to x with $p \equiv a \pmod{d}$, where x is, say, a fixed power of d? Theorem 5.40 does not help us here. The following conjecture does, however:

Conjecture 5.41 *For any $x \geq 2$, $d \geq 2$, and a relatively prime to d, we have*

$$\left|\pi(x; d, a) - \frac{\mathrm{li}(x)}{\phi(d)}\right| \leq x^{1/2}(\log x + 2 \log d).$$

The above conjecture is in fact a consequence of a generalization of the Riemann Hypothesis — see §5.6.

Exercise 5.42 Assuming Conjecture 5.41, show that for all $0 < \alpha < 1/2$, there exists an x_0, such that for all $x > x_0$, for all $2 \leq d \leq x^\alpha$, and for all a relatively prime to d, there are at least $\mathrm{li}(x)/(2\phi(d))$ primes $p \leq x$ such that $p \equiv a \pmod{d}$. □

It is an open problem to prove an unconditional density result analogous to Exercise 5.42 for any positive exponent α. The following, however, is known:

Theorem 5.43 *There exists a constant c such that for all $d \geq 2$ and a relatively prime to d, the least prime p with $p \equiv a \pmod{d}$ is at most $cd^{11/2}$.*

Proof. Literature — see §5.6. □

5.5.4 Sophie Germain Primes

A **Sophie Germain prime** is a prime p such that $2p + 1$ is also prime. Such primes are actually useful in a number of practical applications, and so we discuss them briefly here.

It is an open problem to prove (or disprove) that there are infinitely many Sophie Germain primes. However, numerical evidence, and heuristic arguments, strongly suggest not only that there are infinitely many such primes, but also a fairly precise estimate on the density of such primes.

Let $\pi^*(x)$ denote the number of Sophie Germain primes up to x.

Conjecture 5.44 *We have*

$$\pi^*(x) \sim C\frac{x}{(\log x)^2},$$

where C is the constant

$$C := 2\prod_{q>2}\frac{q(q-2)}{(q-1)^2} \approx 1.32032,$$

and the product is over all primes $q > 2$.

The above conjecture is a special case of a more general conjecture, known as **Hypothesis H**. We can formulate a special case of Hypothesis H (which includes Conjecture 5.44), as follows:

Conjecture 5.45 *Let $(a_1, b_1), \ldots, (a_k, b_k)$ be distinct pairs of integers such that $a_i > 0$ and for all primes p, there exists an integer m such that*

$$\prod_{i=1}^{k}(ma_i + b_i) \not\equiv 0 \pmod{p}.$$

Let $P(x)$ be the number of integers m up to x such that $ma_i + b_i$ are simultaneously prime for $1 \leq i \leq k$. Then

$$P(x) \sim D\frac{x}{(\log x)^k},$$

where

$$D := \prod_{p}\left\{\left(1-\frac{1}{p}\right)^{-k}\left(1-\frac{\omega(p)}{p}\right)\right\},$$

the product being over all primes p, and $\omega(p)$ being the number of distinct solutions m modulo p to the congruence

$$\prod_{i=1}^{k}(ma_i + b_i) \equiv 0 \pmod{p}.$$

The above conjecture also includes (a strong version of) the famous **twin primes conjecture** as a special case: the number of primes p up to x such that $p + 2$ is also prime is $\sim Cx/(\log x)^2$, where C is the same constant as in Conjecture 5.44.

Exercise 5.46 Show that the constant C appearing in Conjecture 5.44 satisfies

$$2C = B_2/B_1^2,$$

where B_1 and B_2 are the constants from Exercises 5.27 and 5.28. □

Exercise 5.47 Show that the quantity D appearing in Conjecture 5.45 is well defined, and satisfies $0 < D < \infty$. □

5.6 Notes

The Prime Number Theorem was conjectured by Gauss in 1791. It was proven independently in 1896 by Hadamard and de la Vallée Poussin. A proof of the Prime Number theorem may be found in the book by Hardy and Wright [33].

Theorem 5.35, as well as the estimates for the constants A, B_1, and B_2 mentioned in Exercises 5.26, 5.27, and 5.28, are from Rosser and Schoenfeld [61].

Theorem 5.37 is from Walfisz [75].

The identity (5.10), which made the first connection between the theory of prime numbers and the zeta function, was discovered in the 18th century by Euler. The Riemann Hypothesis was made by Riemann in 1859, and to this day, remains one of the most vexing conjectures in mathematics. Riemann in fact showed that his conjecture about the zeros of the zeta function is equivalent to the conjecture that for each fixed $\epsilon > 0$, $\pi(x) = \mathrm{li}(x) + O(x^{1/2+\epsilon})$. This was strengthened by Koch in 1901, who showed that the Riemann Hypothesis is true if and only if $\pi(x) = \mathrm{li}(x) + O(x^{1/2} \log x)$. See Chapter 1 of the book by Crandall and Pomerance [23] for more on the connection between the Riemann Hypothesis and the theory of prime numbers; in particular, see Exercise 1.36 in that book for an outline of a proof that Conjecture 5.38 follows from Riemann Hypothesis.

A warning: some authors (and software packages) define the logarithmic integral using the interval of integration $(0, x)$, rather than $(2, x)$, which increases its value by by a constant $c \approx 1.0452$.

Theorem 5.39 was proved by Dirichlet in 1837, while Theorem 5.40 was proved by de la Vallée Poussin in 1896. Conjecture 5.41 was proved by Oesterlé [53] to be a consequence of an assumption about the location of the zeros of certain generalizations of Riemann's zeta function. Theorem 5.43 is from Heath-Brown [34].

Hypothesis H is from Hardy and Littlewood [32].

For the reader who is interested in learning more on the topics discussed in this chapter, we recommend the books by Apostol [7] and Hardy and Wright [33]; indeed, many of the proofs presented in this chapter are minor variations on proofs from these two books. See also Bach and Shallit [11] (especially Chapter 8), Crandall and Pomerance [23] (especially Chapter 1) for a more detailed overview of these topics.

Chapter 6

Discrete Probability Distributions

This chapter introduces concepts from discrete probability theory. We begin with a discussion of finite probability distributions, and then towards the end of the chapter we discuss the more general notion of a discrete probability distribution.

6.1 Finite Probability Distributions: Basic Definitions

A **finite probability distribution** $\mathbf{D} = (\mathcal{U}, \mathsf{P})$ is a *finite* set $\mathcal{U}$, together with a function P that maps $u \in \mathcal{U}$ to $0 \leq \mathsf{P}[u] \leq 1$, such that

$$\sum_{u \in \mathcal{U}} \mathsf{P}[u] = 1. \tag{6.1}$$

The set $\mathcal{U}$ is called the **sample space** and the function P is called the **probability function**.

Intuitively, the elements of $\mathcal{U}$ represent the possible outcomes of a random experiment, where the probability of outcome $u \in \mathcal{U}$ is $\mathsf{P}[u]$.

Throughout this chapter, unless otherwise stated, we shall assume some particular finite probability distribution $\mathbf{D} = (\mathcal{U}, \mathsf{P})$ is under consideration. Also, up until §6.9, we shall use the phrase "probability distribution" to mean "finite probability distribution."

Example 6.1 If we think of rolling a fair die, then $\mathcal{U} = \{1, 2, 3, 4, 5, 6\}$, and $\mathsf{P}[u] = 1/6$ for all $u \in \mathcal{U}$ gives a probability distribution describing the possible outcomes of the experiment. □

Example 6.2 More generally, if $\mathcal{U}$ is a finite set, and $\mathsf{P}[u] = 1/|\mathcal{U}|$ for all $u \in \mathcal{U}$, then $\mathbf{D}$ is called the **uniform distribution on** $\mathcal{U}$. □

Example 6.3 A coin flip is an example of a **Bernoulli trial**, which is in general an experiment with only two possible outcomes: *success*, which occurs with probability p, and *failure*, which occurs with probability $q = 1 - p$. □

Example 6.4 Suppose we perform an experiment by executing n Bernoulli trials, where each trial succeeds with the same probability p, independently of the outcomes of all of the other trials. Let the outcome u of the experiment denote the total number of successes among the n trials. To model this as a probability distribution, we set $\mathcal{U} = \{0, \ldots, n\}$, and for each $0 \leq u \leq n$, we associate the probability

$$\mathsf{P}[u] = \binom{n}{u} p^u q^{n-u},$$

where $q = 1 - p$, since there are n choose u ways to pick which of the n trials succeeds. Such a distribution is called a **binomial distribution**. The reader may verify that the probabilities sum to one. □

An **event** is a subset $\mathcal{A}$ of $\mathcal{U}$, and the **probability** of $\mathcal{A}$ is defined to be

$$\mathsf{P}[\mathcal{A}] := \sum_{u \in \mathcal{A}} \mathsf{P}[u]. \tag{6.2}$$

Thus, we extend the domain of definition of P from $\mathcal{U}$ to the set of all subsets of $\mathcal{U}$.

For an event $\mathcal{A}$, let $\overline{\mathcal{A}}$ denote the complement of $\mathcal{A}$ in $\mathcal{U}$. We have $\mathsf{P}[\emptyset] = 0$, $\mathsf{P}[\mathcal{U}] = 1$, $\mathsf{P}[\overline{\mathcal{A}}] = 1 - \mathsf{P}[\mathcal{A}]$.

For any events $\mathcal{A}, \mathcal{B}$, if $\mathcal{A} \subset \mathcal{B}$, then $\mathsf{P}[\mathcal{A}] \leq \mathsf{P}[\mathcal{B}]$. Also, for any events $\mathcal{A}, \mathcal{B}$, we have

$$\mathsf{P}[\mathcal{A} \cup \mathcal{B}] = \mathsf{P}[\mathcal{A}] + \mathsf{P}[\mathcal{B}] - \mathsf{P}[\mathcal{A} \cap \mathcal{B}] \leq \mathsf{P}[\mathcal{A}] + \mathsf{P}[\mathcal{B}]; \tag{6.3}$$

in particular, if $\mathcal{A}$ and $\mathcal{B}$ are disjoint,

$$\mathsf{P}[\mathcal{A} \cup \mathcal{B}] = \mathsf{P}[\mathcal{A}] + \mathsf{P}[\mathcal{B}]. \tag{6.4}$$

More generally, for any events $\mathcal{A}_1, \ldots, \mathcal{A}_n$ we have

$$Pr[\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_n] \leq \mathsf{P}[\mathcal{A}_1] + \cdots + \mathsf{P}[\mathcal{A}_n], \tag{6.5}$$

and if the $\mathcal{A}_i$'s are pairwise disjoint, then

$$\mathsf{P}[\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_n] = \mathsf{P}[\mathcal{A}_1] + \cdots + \mathsf{P}[\mathcal{A}_n]. \tag{6.6}$$

If $\mathbf{D}_1 = (\mathcal{U}_1, \mathsf{P}_1)$ and $\mathbf{D}_2 = (\mathcal{U}_2, \mathsf{P}_2)$ are probability distributions, we can form the **product distribution** $\mathbf{D} = (\mathcal{U}, \mathsf{P})$, where $\mathcal{U} := \mathcal{U}_1 \times \mathcal{U}_2$, and $\mathsf{P}[(u_1, u_2)] := \mathsf{P}_1[u_1]\mathsf{P}_2[u_2]$. It is easy to verify that the product distribution is also a probability distribution.

Intuitively, the elements (u_1, u_2) of $\mathcal{U}_1 \times \mathcal{U}_2$ denote the possible outcomes of two separate experiments.

More generally, if $\mathbf{D}_i = (\mathcal{U}_i, \mathsf{P}_i)$ for $1 \leq i \leq n$, we can define the product distribution $\mathbf{D} = (\mathcal{U}, \mathsf{P})$, where $\mathcal{U} := \mathcal{U}_1 \times \cdots \times \mathcal{U}_n$, and $\mathsf{P}[(u_1, \ldots, u_n)] := \mathsf{P}[u_1] \ldots \mathsf{P}[u_n]$.

Example 6.5 Continuing with Example 6.1, the probability of an "odd roll" $\mathcal{A} = \{1, 3, 5\}$ is $1/2$. □

Example 6.6 More generally, if $\mathbf{D}$ is the uniform distribution of a set $\mathcal{U}$ of cardinality n, and $\mathcal{A}$ is a subset of $\mathcal{U}$ of cardinality k, then $\mathsf{P}[\mathcal{A}] = k/n$. □

Example 6.7 Alice rolls two dice, and asks Bob (without looking) to guess a value that appears on either of the two dice. Let us model this situation by considering the uniform distribution on $\{(x, y) : 1 \leq x, y \leq 6\}$, where x represents the value of the first die, and y the value of the second, which is the product distribution of two copies of the distribution from Example 6.1.

For $1 \leq x \leq 6$, let $\mathcal{A}_x$ be the event that the first die is x, and $\mathcal{B}_x$ the event that the second die is x, Let $\mathcal{C}_x = \mathcal{A}_x \cup \mathcal{B}_x$ be the event that x appears on either of the two dice. No matter what value $1 \leq x \leq 6$ Bob chooses, the probability that this choice is correct is

$$\mathsf{P}[\mathcal{C}_x] = \mathsf{P}[\mathcal{A}_x \cup \mathcal{B}_x] = \mathsf{P}[\mathcal{A}_x] + \mathsf{P}[\mathcal{B}_x] - \mathsf{P}[\mathcal{A}_x \cap \mathcal{B}_x] = 1/6 + 1/6 - 1/36 = 11/36.$$

□

Exercise 6.8 Using Equation 6.3, prove the *inclusion/exclusion principle:* for events $\mathcal{A}_1, \ldots, \mathcal{A}_n$,

$$\mathsf{P}[\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_n] = \sum_{\ell=1}^{n} (-1)^{\ell-1} \sum_{i_1,\ldots,i_\ell} \mathsf{P}[\mathcal{A}_{i_1} \cap \cdots \cap \mathcal{A}_{i_\ell}],$$

where the inner sum is over all subsets of ℓ distinct indices between 1 and n. □

6.2 Conditional Probability and Independence

For events $\mathcal{A}$ and $\mathcal{B}$ with $\mathsf{P}[\mathcal{B}] \neq 0$, the **conditional probability of $\mathcal{A}$ given $\mathcal{B}$** is defined as

$$\mathsf{P}[\mathcal{A} \mid \mathcal{B}] \;:=\; \mathsf{P}[\mathcal{A} \cap \mathcal{B}]/\mathsf{P}[\mathcal{B}].$$

Intuitively, $\mathsf{P}[\mathcal{A} \mid \mathcal{B}]$ is the probability that event $\mathcal{A}$ occurred, given that event $\mathcal{B}$ occurred; that is, if a random experiment produces an outcome according to the given probability distribution $\mathbf{D}$, and we know that the outcome lies in $\mathcal{B}$ — but nothing more about the outcome — then $\mathsf{P}[\mathcal{A} \mid \mathcal{B}]$ represents the probability that the outcome lies in $\mathcal{A}$, given this partial knowledge about the outcome.

The function $\mathsf{P}[\cdot \mid \mathcal{B}]$ defines another probability distribution on $\mathcal{U}$, namely, $\mathbf{D}_{\mathcal{B}} = (\mathcal{U}, \mathsf{P}[\cdot \mid \mathcal{B}])$, called the **conditional distribution given by $\mathcal{B}$**.

For events $\mathcal{A}$ and $\mathcal{B}$, if $\mathsf{P}[\mathcal{A} \cap \mathcal{B}] = \mathsf{P}[\mathcal{A}] \cdot \mathsf{P}[\mathcal{B}]$, then $\mathcal{A}$ and $\mathcal{B}$ are called **independent** events. If $\mathsf{P}[\mathcal{B}] \neq 0$, a simple calculation shows that $\mathcal{A}$ and $\mathcal{B}$ are independent if and only if $\mathsf{P}[\mathcal{A} \mid \mathcal{B}] = \mathsf{P}[\mathcal{A}]$.

A collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of events is called **pairwise independent** of $\mathsf{P}[\mathcal{A}_i \cap \mathcal{A}_j] = \mathsf{P}[\mathcal{A}_i]\mathsf{P}[\mathcal{A}_j]$ for all $i \neq j$, and is called **mutually independent** if every subset $\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_k}$ of the collection satisfies

$$\mathsf{P}[\mathcal{A}_{i_1} \cap \cdots \cap \mathcal{A}_{i_k}] = \mathsf{P}[\mathcal{A}_{i_1}] \cdots \mathsf{P}[\mathcal{A}_{i_k}].$$

Example 6.9 In Example 6.7, suppose that Alice tells Bob that the sum of the two dice before Bob makes his guess. For example, suppose Alice tells Bob the sum is 4. Then what is Bob's best strategy in this case? Let $\mathcal{S}_z$ be the event that the sum is z, for $2 \leq z \leq 12$, and consider the conditional probability distribution determined by $\mathcal{S}_4$. This is the uniform distribution on the three pairs $(1,3),(2,2),(3,1)$. The numbers 1 and 3 both appear in two pairs, while the number 2 appears in just one pair. Therefore,

$$\mathsf{P}[\mathcal{C}_1 \mid \mathcal{S}_4] = \mathsf{P}[\mathcal{C}_3 \mid \mathcal{S}_4] = 2/3,$$

while

$$\mathsf{P}[\mathcal{C}_2 \mid \mathcal{S}_4] = 1/3$$

and

$$\mathsf{P}[\mathcal{C}_4 \mid \mathcal{S}_4] = \mathsf{P}[\mathcal{C}_5 \mid \mathcal{S}_4] = \mathsf{P}[\mathcal{C}_6 \mid \mathcal{S}_4] = 0.$$

Thus, if the sum is 4, Bob's best strategy is to guess either 1 or 3.

Note that the events $\mathcal{A}_1$ and $\mathcal{B}_2$ are independent, while the events $\mathcal{A}_1$ and $\mathcal{S}_4$ are not. □

Example 6.10 Suppose we toss three fair coins. Let $\mathcal{A}_1$ be the event that the first coin is "heads," let $\mathcal{A}_2$ be the event that the second coin is "heads," and let $\mathcal{A}_3$ be the event that the third coin is "heads." Then the collection of events $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ is mutually independent.

Now let $\mathcal{B}_{12}$ be the event that the first and second coins agree (i.e., both "heads" or both "tails"), let $\mathcal{B}_{13}$ be the event that the first and third coins agree, and let $\mathcal{B}_{23}$ be the event that the second and third coins agree. Then the collection of events $\{\mathcal{B}_{12}, \mathcal{B}_{13}, \mathcal{B}_{23}\}$ is pairwise independent, but not mutually independent. Indeed, the probability that any one of the events occurs is $1/2$, and the probability that any two of the three events occurs is $1/4$; however, the probability that all three occurs is also $1/4$, since if any two events occur, then so does the third. □

Suppose we have a collection $\mathcal{B}_1, \ldots, \mathcal{B}_n$ of events that partitions $\mathcal{U}$ (i.e., the $\mathcal{B}_i$ are non-empty, pairwise disjoint, and their union is $\mathcal{U}$), then it is easy to see that for any event $\mathcal{A}$,

$$\mathsf{P}[\mathcal{A}] = \sum_{i=1}^{n} \mathsf{P}[\mathcal{A} \cap \mathcal{B}_i] = \sum_{i=1}^{n} \mathsf{P}[\mathcal{A} \mid \mathcal{B}_i] \cdot \mathsf{P}[\mathcal{B}_i]. \tag{6.7}$$

Furthermore, if $\mathsf{P}[\mathcal{A}] \neq 0$, then for any $1 \leq j \leq n$, we have

$$\mathsf{P}[\mathcal{B}_j \mid \mathcal{A}] = \frac{\mathsf{P}[\mathcal{A} \cap \mathcal{B}_j]}{\mathsf{P}[\mathcal{A}]} = \frac{\mathsf{P}[\mathcal{A} \mid \mathcal{B}_j]\mathsf{P}[\mathcal{B}_j]}{\sum_{i=1}^{n} \mathsf{P}[\mathcal{A} \mid \mathcal{B}_i]\mathsf{P}[\mathcal{B}_i]}. \tag{6.8}$$

This equality, known as **Bayes' Theorem**, allows us to compute the conditional probability $\mathsf{P}[\mathcal{B}_j \mid \mathcal{A}]$ in terms of the conditional probabilities $\mathsf{P}[\mathcal{A} \mid \mathcal{B}_i]$.

The equation (6.7) is useful for computing or estimating probabilities by conditioning on specific events $\mathcal{B}_i$ in such a way that the conditional probabilities $\mathsf{P}[\mathcal{A} \mid \mathcal{B}_i]$ are easy to

compute or estimate. Also, if we want to compute a conditional probability $\mathsf{P}[\mathcal{A} \mid \mathcal{C}]$, we can do so by partitioning $\mathcal{C}$ into sets $\mathcal{B}_1, \ldots, \mathcal{B}_n$, and use the following simple fact:

$$\mathsf{P}[\mathcal{A} \mid \mathcal{C}] = \sum_{i=1}^{n} \mathsf{P}[\mathcal{A} \mid \mathcal{B}_i]\mathsf{P}[\mathcal{B}_i]/\mathsf{P}[\mathcal{C}]. \tag{6.9}$$

Example 6.11 This example is based on the TV game show "Let's make a deal," which was popular in the 1970's. In this game, a contestant chooses one of three doors. Behind two doors is a "zonk," e.g., something of little or no value, and behind one of the doors is a "grand prize," e.g., a car or vacation package. We may assume that the door behind which the grand prize is placed is chosen at random from among the three doors, with equal probability. After the contestant chooses a door, the host of the show, Monty Hall, always reveals a zonk behind one of the two doors not chosen by the contestant. The contestant is then given a choice: either stay with his initial choice of door, or switch to the other unopened door. After the contestant finalizes his decision on which door to choose, that door is opened and he wins whatever is behind the chosen door. The question is, which strategy for is better for the contestant: to stay or to switch?

Let us evaluate the two strategies. If the contestant always stays with his initial selection, then it is clear that his probability of success is exactly $1/3$.

Now consider the strategy of always switching. Let $\mathcal{B}$ be the event that the contestant's initial choice was correct, and let $\mathcal{A}$ be the event that the contestant wins the grand prize. On the one hand, if the contestant's initial choice was correct, then switching will certainly lead to failure. That is, $\mathsf{P}[\mathcal{A} \mid \mathcal{B}] = 0$. On the other hand, suppose that the contestant's initial choice was incorrect, so that one of the zonks is behind the initially chosen door. Since Monty reveals the other zonk, switching will lead with certainty to success. That is, $\mathsf{P}[\mathcal{A} \mid \overline{\mathcal{B}}] = 1$. Furthermore, it is clear that $\mathsf{P}[\mathcal{B}] = 1/3$. So we compute

$$\mathsf{P}[\mathcal{A}] = \mathsf{P}[\mathcal{A} \mid \mathcal{B}]\mathsf{P}[\mathcal{B}] + \mathsf{P}[\mathcal{A} \mid \overline{\mathcal{B}}]\mathsf{P}[\overline{\mathcal{B}}] = 0 \cdot (1/3) + 1 \cdot (2/3) = 2/3.$$

Thus, the "stay" strategy has a success probability of $1/3$, while the "switch" strategy has a success probability of $2/3$. So it is better to switch than to stay.

Of course, real life is a bit more complicated. Monty did not always reveal a zonk and offer a choice to switch. Indeed, if Monty *only* revealed a zonk when the contestant had chosen the correct door, then switching would certainly be the wrong strategy. However, if Monty's choice itself was a random decision made independent of the contestant's initial choice, then switching is again the preferred strategy. □

Example 6.12 Suppose that the rate of incidence of disease X in the overall population is 1%. Also suppose that there is a test for disease X; however, the test is not perfect: it has a 5% false positive rate, and a 2% false negative rate. A doctor gives the test to a patient and it comes out positive. How should the doctor advise his patient? In particular, what is the probability that the patient actually has disease X, given a positive test result?

Amazingly, many trained doctors will say the probability is 95%, since the test has a false positive rate of 5%. However, this conclusion is completely wrong.

Let $\mathcal{A}$ be the event that the test is positive and let $\mathcal{B}$ be the event that the patient has disease X. The relevant quantity that we need to estimate is $\mathsf{P}[\mathcal{B} \mid \mathcal{A}]$; that is, the probability that the patient has disease X, given a positive test result. We use Bayes' Theorem to do this:

$$\mathsf{P}[\mathcal{B} \mid \mathcal{A}] = \frac{\mathsf{P}[\mathcal{A} \mid \mathcal{B}]\mathsf{P}[\mathcal{B}]}{\mathsf{P}[\mathcal{A} \mid \mathcal{B}]\mathsf{P}[\mathcal{B}] + \mathsf{P}[\mathcal{A} \mid \overline{\mathcal{B}}]\mathsf{P}[\overline{\mathcal{B}}]} = \frac{0.98 \cdot 0.01}{0.98 \cdot 0.01 + 0.05 \cdot 0.99} \approx 0.17.$$

Thus, the chances that the patient has disease X given a positive test result is just 17%. The correct intuition here is that it is much more likely to get a false positive than it is to actually have the disease.

Of course, the real world is a bit more complicated than this example suggests: the doctor may be giving the patient the test because other risk factors or symptoms may suggest that the patient is more likely to have the disease than a random member of the population, in which case the above analysis does not apply. □

Exercise 6.13 Show that if two events $\mathcal{A}$ and $\mathcal{B}$ are independent, then so are $\mathcal{A}$ and $\overline{\mathcal{B}}$. □

Exercise 6.14 Suppose we roll two dice, and let (x, y) denote the outcome (as in Example 6.7). For each of the following pairs of events $\mathcal{A}$ and $\mathcal{B}$, determine if they are independent or not:

- $\mathcal{A}$: $x = y$; $\mathcal{B}$: $y = 1$.
- $\mathcal{A}$: $x \geq y$; $\mathcal{B}$: $y = 1$.
- $\mathcal{A}$: $x \geq y$; $\mathcal{B}$: $y^2 = 7y - 6$.
- $\mathcal{A}$: $xy = 6$; $\mathcal{B}$: $y = 3$.

□

6.3 Random Variables

Let $\mathbf{D} = (\mathcal{U}, \mathsf{P})$ be a probability distribution. It is sometimes convenient to associate a real number, or other mathematical object, with each outcome $u \in \mathcal{U}$. Such an association is called a **random variable**; more formally, a random variable X is a function from $\mathcal{U}$ into a set $\mathcal{X}$. If $\mathcal{X}$ is a subset of the real numbers, then X is called a **real random variable**. For a random variable $X : \mathcal{U} \to \mathcal{X}$, we define $\mathrm{im}(X) := X(\mathcal{U}) = \{X(u) : u \in \mathcal{U}\}$.

One may define any number of random variables on a given probability distribution. If $X : \mathcal{U} \to \mathcal{X}$ is a random variable, and $f : \mathcal{X} \to \mathcal{Y}$ is a function, then $f(X) := f \circ X$ is also a random variable.

Example 6.15 Suppose we flip n fair coins. Then we may define a random variable X that maps each outcome to a bit string of length n, where a "head" is encoded as a 1-bit, and a "tail" is encoded as a 0-bit. We may define another random variable Y that is the number of "heads." The variable Y is a real random variable. □

Let $X : \mathcal{U} \to \mathcal{X}$ be a random variable. For $x \in \mathcal{X}$, we write "$X = x$" as shorthand for the event $\{u \in \mathcal{U} : X(u) = x\}$. More generally, for and any predicate ϕ, we may write "$\phi(X)$" as shorthand for the event $\{u \in \mathcal{U} : \phi(X)\}$.

A random variable X defines a probability distribution on $\mathrm{im}(X)$, where the probability associated with $x \in \mathrm{im}(X)$ is $\mathsf{P}[X = x]$. We call this the **distribution of** X. For two random variables X, Y defined on a probability distribution, $Z = (X, Y)$ is also a random variable whose distribution is called the **joint distribution of** X **and** Y.

If X is a random variable, and $\mathcal{A}$ an an event, then the **conditional distribution of** X **given by** $\mathcal{A}$ is the probability distribution on $\mathrm{im}(X)$, where the probability associated with $x \in \mathrm{im}(X)$ is $\mathsf{P}[X = x \mid \mathcal{A}]$.

We say two random variables X, Y are **independent** if for all $x \in \mathrm{im}(X)$ and $y \in \mathrm{im}(Y)$, the events $X = x$ and $Y = y$ are independent, i.e.,

$$\mathsf{P}[X = x \wedge Y = y] = \mathsf{P}[X = x]\mathsf{P}[Y = y].$$

Equivalently, X and Y are independent if and only if their joint distribution is equal to the product of their individual distributions. Alternatively, X and Y are independent if and only if for all $x \in \mathrm{im}(X)$ the conditional distribution of Y given by the event $X = x$ is the same as the distribution of Y.

A collection $X_1, \ldots, X_n$ of random variables is called **pairwise independent** if for all $1 \leq i < j \leq n$, X_i and X_j are independent. We say that $X_1, \ldots, X_n$ are **mutually independent** if for all $x_1 \in \mathrm{im}(X_1), \ldots, x_n \in \mathrm{im}(X_n)$, we have

$$\mathsf{P}[X_1 = x_n \wedge \cdots \wedge X_n = x_n] = \mathsf{P}[X_1 = x_1] \cdots \mathsf{P}[X_n = x_n].$$

More generally, for $2 \leq k \leq n$, we say that the random variables $X_1, \ldots, X_n$ are k**-wise independent** if any k of them are mutually independent.

Example 6.16 We toss 3 coins, and set $X_i = 0$ if the ith coin is "tails," and $X_i = 1$ otherwise. The variables X_1, X_2, X_3 are mutually independent. Let us set $Y_{12} = X_1 \oplus X_2$, $Y_{13} = X_1 \oplus X_3$, and $Y_{23} = X_2 \oplus X_3$, where "$\oplus$" denotes "exclusive or," i.e., addition modulo 2. Then the variables Y_{12}, Y_{13}, Y_{23} are pairwise independent, but not mutually independent — observe that $Y_{12} \oplus Y_{13} = Y_{23}$. □

The following is a simple but useful fact:

Theorem 6.17 *Let $X_i : \mathcal{U} \to \mathcal{X}_i$, for $1 \leq i \leq n$, be random variables, and suppose that there exist functions $f_i : \mathcal{X}_i \to [0, 1]$, for $1 \leq i \leq n$, such that*

$$\sum_{x_i \in \mathcal{X}_i} f_i(x_i) = 1 \quad (i = 1 \ldots n),$$

and

$$\mathsf{P}[X_1 = x_1 \wedge \cdots \wedge X_n = x_n] = f_1(x_1) \cdots f_n(x_n) \quad (\textit{for all } x_1 \in \mathcal{X}_1, \ldots, x_n \in \mathcal{X}_n).$$

Then for any subset of indices $1 \leq i_1 < i_2 < \cdots < i_\ell \leq n$*, we have*

$$\mathsf{P}[X_{i_1} = x_{i_1} \wedge \cdots \wedge X_{i_\ell} = x_{i_\ell}] = f(x_{i_1}) \cdots f(i_\ell) \quad (\textit{for all } x_{i_1} \in \mathcal{X}_{i_1}, \ldots, x_{i_\ell} \in \mathcal{X}_{i_\ell}).$$

Proof. We may assume that $\{i_1, \ldots, i_\ell\} = \{1, \ldots, \ell\}$ — otherwise, just reorder the X_i's. Now fix $x_1, \ldots, x_\ell$. We have

$$\begin{aligned}
&\mathsf{P}[X_1 = x_1 \wedge \cdots \wedge X_\ell = x_\ell] \\
&\quad = \sum_{x_{\ell+1}} \cdots \sum_{x_n} \mathsf{P}[X_1 = x_1 \wedge \cdots \wedge X_\ell = x_\ell \wedge X_{\ell+1} = x_{\ell+1} \wedge \cdots \wedge X_n = x_n] \\
&\quad = \sum_{x_{\ell+1}} \cdots \sum_{x_n} f_1(x_1) \cdots f(x_\ell) f(x_{\ell+1}) \cdots f(x_n) \\
&\quad = f(x_1) \cdots f(x_\ell) \Big(\sum_{x_{\ell+1}} f(x_{\ell+1}) \Big) \cdots \Big(\sum_{x_n} f(x_n) \Big) \\
&\quad = f(x_1) \cdots f(x_\ell).
\end{aligned}$$

□

The following three theorems are immediate consequences of the above theorem:

Theorem 6.18 *Let* $X_i : \mathcal{U} \to \mathcal{X}_i$*, for* $1 \leq i \leq n$*, be random variables such that*

$$\mathsf{P}[X_1 = x_1 \wedge \cdots \wedge X_n = x_n] = \frac{1}{|\mathcal{X}_1|} \cdots \frac{1}{|\mathcal{X}_n|} \qquad (\textit{for all } x_1 \in \mathcal{X}_1, \ldots, x_n \in \mathcal{X}_n).$$

Then the X_i*'s are mutually independent with each* X_i *uniformly distributed over* $\mathcal{X}_i$*.*

Theorem 6.19 *If* $X_1, \ldots, X_n$ *are mutually independent random variables, then they are* k*-wise independent for all* $2 \leq k \leq n$*.*

Theorem 6.20 *If* $\mathbf{D}_i = (\mathcal{U}_i, \mathsf{P}_i)$ *are probability distributions for* $1 \leq i \leq n$*, then the projection functions* $\pi_i : \mathcal{U}_1 \times \cdots \times \mathcal{U}_n \to \mathcal{U}_i$*, where* $\pi_i(u_1, \ldots, u_n) = u_i$*, are mutually independent random variables on the product distribution* $\mathbf{D}_1 \times \cdots \times \mathbf{D}_n$*.*

We also have:

Theorem 6.21 *If* $X_1, \ldots, X_n$ *are mutually independent random variables, and* $g_1, \ldots, g_n$ *are functions, then* $g_1(X_1), \ldots, g_n(X_n)$ *are also mutually independent random variables.*

Proof. Exercise. □

Example 6.22 If we toss n dice, and let X_i denote the value of the ith die for $1 \leq i \leq n$, then the X_i's are mutually independent random variables. If we set $Y_i = X_i^2$ for $1 \leq i \leq n$, then the Y_i's are also mutually independent random variables. □

Example 6.23 This example again illustrates the notion of pairwise independence. Let X and Y be independent and uniformly distributed over $\mathbb{Z}_p$, where p is a prime. For $a \in \mathbb{Z}_p$, let $Z_a := aX + Y$. Then we claim that each Z_a is uniformly distributed over $\mathbb{Z}_p$, and that the collection of random variables $\{Z_a : a \in \mathbb{Z}_p\}$ is pairwise independent.

To prove this claim, let $a, b \in \mathbb{Z}_p$ with $a \neq b$, and consider the map $f_{a,b} : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_p \times \mathbb{Z}_p$ that sends (x, y) to $(ax + y, bx + y)$. It is easy to see that $f_{a,b}$ is injective; indeed, if $ax + y = ax' + y'$ and $bx + y = bx' + y'$, then subtracting these two equations, we obtain $(a - b)x = (a - b)x'$, and since $a - b \neq [0 \bmod p]$, it follows that $x = x'$, which also implies $y = y'$. Since $f_{a,b}$ is injective, it must be a bijection from $\mathbb{Z}_p \times \mathbb{Z}_p$ onto itself. Thus, since (X, Y) is uniformly distributed over $\mathbb{Z}_p \times \mathbb{Z}_p$, so is $(Z_a, Z_b) = (aX + Y, bX + Y)$. So for all $z, z' \in \mathbb{Z}_p$, we have

$$\mathsf{P}[Z_a = z \wedge Z_b = z'] = 1/p^2,$$

and so the claim follows from Theorem 6.18.

Note that the Z_a's are not 3-wise independent, since the value of any two determines the value of all the rest (verify). □

Example 6.24 We can generalize the previous example as follows. Let $X_1, \ldots, X_t, Y$ be mutually independent and uniformly distributed over $\mathbb{Z}_p$, where p is prime, and for $a_1, \ldots, a_t \in \mathbb{Z}_p$, let $Z_{a_1,\ldots,a_t} := a_1X_1 + \cdots + a_tX_t + Y$. We leave it to the reader to verify that each $Z_{a_1,\ldots,a_t}$ is uniformly distributed over $\mathbb{Z}_p$, and that the collection of all such $Z_{a_1,\ldots,a_t}$ is pairwise independent. □

Example 6.25 Let W, X, Y be mutually independent and uniformly distributed over $\mathbb{Z}_p$, where p is prime. For any $a \in \mathbb{Z}_p$, let $Z_a = a^2W + aX + Y$. We leave it to the reader to verify that each Z_a is uniformly distributed over $\mathbb{Z}_p$, and that the collection of all Z_a's is 3-wise independent. □

Using other algebraic techniques, there are many ways to construct families of pairwise and k-wise independent random variables. Such families play an important role in many areas of computer science.

6.4 Expectation and Variance

If X is a real random variable, then its **expected value** or **mean** is

$$\mathsf{E}[X] := \sum_{u \in \mathcal{U}} X(u) \cdot \mathsf{P}[u],$$

or equivalently,

$$\mathsf{E}[X] = \sum_{x \in \mathrm{im}(X)} \sum_{u \in X^{-1}(x)} x\mathsf{P}[u] = \sum_{x \in \mathrm{im}(X)} x \cdot \mathsf{P}[X = x]. \tag{6.10}$$

By a similar calculation, one sees that if X is a random variable, and f is a real-valued function on $\mathrm{im}(X)$, then

$$\mathsf{E}[f(X)] = \sum_{x \in \mathrm{im}(X)} f(x)\mathsf{P}[X = x]. \tag{6.11}$$

Theorem 6.26 *For real random variables X, Y, and real numbers a, b, we have $\mathsf{E}[aX + bY] = a\mathsf{E}[X] + b\mathsf{E}[Y]$.*

Proof. Exercise. □

So we see that expectation is linear; however, expectation is not in general multiplicative, except in the case of independent random variables:

Theorem 6.27 *If X and Y are independent real random variables, then $\mathsf{E}[XY] = \mathsf{E}[X]\mathsf{E}[Y]$.*

Proof. We have

$$\begin{aligned}
\mathsf{E}[XY] &= \sum_{x \in \mathrm{im}(X)} \sum_{y \in \mathrm{im}(Y)} xy\mathsf{P}[X = x \wedge Y = y] \\
&= \sum_{x \in \mathrm{im}(X)} \sum_{y \in \mathrm{im}(Y)} xy\mathsf{P}[X = x]\mathsf{P}[Y = y] \\
&= \Big(\sum_{x \in \mathrm{im}(X)} x\mathsf{P}[X = x] \Big) \Big(\sum_{y \in \mathrm{im}(Y)} y\mathsf{P}[Y = y] \Big) \\
&= \mathsf{E}[X] \cdot \mathsf{E}[Y].
\end{aligned}$$

□

More generally, the above theorem implies (using a simple induction argument) that if $X_1, \ldots, X_n$ are mutually independent, then $\mathsf{E}[X] = \mathsf{E}[X_1] \cdots \mathsf{E}[X_n]$.

Exercise 6.28 A casino offers you the following four dice games. In each game, you pay 15 dollars to play, and two dice are rolled. In the first game, the house pays out four times the value of the first die (in dollars). In the second, the house pays out twice the sum of the two die. In the third, the house pays the square of the first. In the fourth, the house pays the product of the two dice. Which game should you play? That is, which game maximizes your expected winnings? □

The following fact is sometimes quite useful:

Theorem 6.29 *If X is a random variable that takes values in a set $\{0, 1, \ldots, n\}$, then*

$$\mathsf{E}[X] = \sum_{i=1}^{n} \mathsf{P}[X \geq i].$$

Proof. For $1 \leq i \leq n$, set define the random variable X_i so that $X_i = 1$ if $X \geq i$ and $X_i = 0$ if $X < i$. Observe that $\mathsf{E}[X_i] = 1 \cdot \mathsf{P}[X \geq i] + 0 \cdot \mathsf{P}[X < i] = \mathsf{P}[X \geq i]$. Moreover, $X = X_1 + \cdots + X_n$, and hence

$$\mathsf{E}[X] = \sum_{i=1}^{n} \mathsf{E}[X_i] = \sum_{i=1}^{n} \mathsf{P}[X \geq i].$$

□

The **variance** of a real random variable X is $\mathsf{Var}[X] := \mathsf{E}[(X - \mathsf{E}[X])^2]$. The variance provides a measure of the spread or dispersion of the distribution of X around its mean $\mathsf{E}[X]$. Note that since $(X - \mathsf{E}[X])^2$ is always non-negative, variance is always non-negative.

Theorem 6.30 *We have* $\mathsf{Var}[X] = \mathsf{E}[X^2] - (\mathsf{E}[X])^2$, *and for any real numbers* a *and* b, $\mathsf{Var}[aX + b] = a^2\mathsf{Var}[X]$.

Proof. Exercise. □

Example 6.31 If X denotes the value of a die toss, then $\mathsf{Var}[X] = 91/6 - 3.5^2 \approx 2.9167$. □

Theorem 6.32 *If* $X_1, \ldots, X_n$ *is a collection of* pairwise independent *random variables, then*

$$\mathsf{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathsf{Var}[X_i].$$

Proof. We have

$$\begin{aligned}
\mathsf{Var}\left[\sum_i X_i\right] &= E\left[(\sum_i X_i)^2\right] - \left(\mathsf{E}[\sum_i X_i]\right)^2 \\
&= \sum_i \mathsf{E}[X_i^2] + 2\sum_i \sum_{j<i} (\mathsf{E}[X_i X_j] - \mathsf{E}[X_i]\mathsf{E}[X_j]) - \sum_i \mathsf{E}[X_i]^2 \\
&\qquad \text{(by Theorem 6.26 and rearranging terms)} \\
&= \sum_i \mathsf{E}[X_i^2] - \sum_i \mathsf{E}[X_i]^2 \\
&\qquad \text{(by pairwise independence and Theorem 6.27)} \\
&= \sum_i \mathsf{Var}[X_i].
\end{aligned}$$

□

For any random variable X and event $\mathcal{B}$, with $\mathsf{P}[\mathcal{B}] \neq 0$, we can define the **conditional expectation** $\mathsf{E}[X \mid \mathcal{B}]$ to be the expected value of X in the conditional probability distribution given by $\mathcal{B}$. We have

$$\mathsf{E}[X \mid \mathcal{B}] = \sum_{u \in \mathcal{U}} X(u) \cdot \mathsf{P}[u \mid \mathcal{B}] = \sum_{x \in \mathrm{im}(X)} x\mathsf{P}[X = x \mid \mathcal{B}]. \tag{6.12}$$

If $\mathcal{B}_1, \ldots, \mathcal{B}_n$ is a collection of events that partitions $\mathcal{U}$, then it follows from the definitions that

$$\mathsf{E}[X] = \sum_{i=1}^{n} \mathsf{E}[X \mid \mathcal{B}_i]\mathsf{P}[\mathcal{B}_i]. \tag{6.13}$$

6.5 Some Useful Bounds

In this section, we present several theorems that can be used to bound the probability that a random variable deviates from its mean by some specified amount.

Theorem 6.33 (Markov's Inequality) *Let X be a random variable that takes only non-negative real values. Then for any $t > 0$, we have*

$$\mathsf{P}[X \geq t] \leq \mathsf{E}[X]/t.$$

Proof. We have

$$\mathsf{E}[X] = \sum_{x} x\mathsf{P}[X = x] \;=\; \sum_{x<t} x\mathsf{P}[X = x] + \sum_{x \geq t} \mathsf{P}[X = x].$$

Since X takes only non-negative values, all of the terms in the summation are non-negative. Therefore,

$$\mathsf{E}[X] \geq \sum_{x \geq t} x\mathsf{P}[X = x] \;\geq\; \sum_{x \geq t} t\mathsf{P}[X = x] \;=\; t\mathsf{P}[X \geq t].$$

□

Markov's Inequality may be the only game in town when nothing more about the distribution of X is known besides its mean. However, if the variance of X is also known, then one can get a better bound.

Theorem 6.34 (Chebyshev's Inequality) *Let X be a real random variable. Then for any $t > 0$, we have*

$$\mathsf{P}[|X - \mathsf{E}[X]| \geq t] \leq \mathsf{Var}[X]/t^2.$$

Proof. Let $Y = (X - \mathsf{E}[X])^2$. Then Y is always non-negative, and $\mathsf{E}[Y] = \mathsf{Var}[X]$. Applying Markov's Inequality to Y, we have

$$Pr[|X - \mathsf{E}[X]| \geq t] = \mathsf{P}[Y \geq t^2] \leq \mathsf{Var}[X]/t^2.$$

□

An important special case is the following.

Suppose that $X_1, \ldots, X_n$ are random variables, such that X_i is 1 with probability p_i, and 0 with probability $q_i = 1 - p_i$. Further, consider the sum $X = \sum_{i=1}^{n} X_i$. Thus, X represents the number of successes among n (not necessarily independent) Bernoulli trials.

For each i, we have

$$\mathsf{E}[X_i] = \mathsf{E}[X_i^2] = 1 \cdot p_i + 0 \cdot q_i = p_i,$$

and

$$\mathsf{Var}[X_i] = \mathsf{E}[X_i^2] - (\mathsf{E}[X_i])^2 = p_i - p_i^2 = p_i q_i.$$

By the linearity of expectation, we have

$$\mathsf{E}[X] = \sum_{i=1}^{n} p_i.$$

If the collection of X_i's is *pairwise* independent, then by Theorem 6.32, we have

$$\mathsf{Var}[X] = \sum_{i=1}^{n} p_i q_i.$$

Applying Chebyshev's inequality, we obtain the following:

Theorem 6.35 *Let $X_1, \ldots, X_n$ be pairwise independent random variables, such that X_i is 1 with probability p_i and 0 with probability $q_i = 1 - p_i$, and let $\mu := \sum_{i=1}^n p_i$ and $\nu := \sum_{i=1}^n p_i q_i$. Then for any $t > 0$, we have*

$$\mathsf{P}[|X - \mu| \geq t] \leq \frac{\nu}{t^2}.$$

If the X_i's are *mutually* independent, then stronger bounds can be obtained. Note that if the probabilities p_i are all equal, the variable X has a binomial distribution.

Theorem 6.36 (Chernoff Bound) *Let $X_1, \ldots, X_n$ be mutually independent random variables, such that X_i is 1 with probability p_i and 0 with probability $q_i = 1 - p_i$, and let $\mu := \sum_{i=1}^n p_i$ and $\nu := \sum_{i=1}^n p_i q_i$. Then for any $t > 0$, we have*

$$\mathsf{P}[X - \mu \geq t] \leq e^{-t^2/4\nu}.$$

Proof. Let $\alpha > 0$ be a parameter whose value will be fixed below. Define the random variable $Z = e^{\alpha(X-\mu)}$. Since the function $x \mapsto e^{\alpha x}$ is strictly increasing, and by Markov's Inequality, we have

$$\mathsf{P}[X - \mu \geq t] = \mathsf{P}[Z \geq e^{\alpha t}] \leq \mathsf{E}[Z] e^{-\alpha t}.$$

So we wish to bound $E[Z]$ from above.

Using parts (1) and (2) of §A.6, we derive the following, for $1 \leq i \leq n$:

$$\begin{aligned} p_i e^{\alpha q_i} + q_i e^{-\alpha p_i} &\leq p_i(1 + \alpha q_i + (\alpha q_i)^2) + q_i(1 - \alpha p_i + (\alpha p_i)^2) \\ &= 1 + \alpha^2 p_i q_i \\ &\leq e^{\alpha^2 p_i q_i}. \end{aligned}$$

For $1 \leq i \leq n$, define the random variable $Z_i = e^{\alpha X_i - p_i}$. Note that $Z = \prod_{i=1}^{n} Z_i$, that the Z_i's are mutually independent random variables, and that

$$\mathsf{E}[Z_i] = e^{\alpha(1-p_i)}p_i + e^{\alpha(0-p_i)}q_i = p_i e^{\alpha q_i} + q_i e^{-\alpha p_i} \leq e^{\alpha^2 p_i q_i}.$$

It follows that

$$\mathsf{E}[Z] = \mathsf{E}[\prod_i Z_i] = \prod_i \mathsf{E}[Z_i] \leq e^{\alpha^2 \nu}.$$

Thus we have

$$\mathsf{P}[X - \mu \geq t] \leq e^{\alpha^2 \nu - \alpha t}.$$

It is a simple matter to show that for fixed $a, b > 0$, the function $f(s) = as^2 - bs$ is minimized at $s = b/2a$. So we set $\alpha = t/2\nu$, and calculate

$$\mathsf{P}[X - \mu \geq t] \leq e^{-t^2/4\nu}.$$

□

One can also obtain the "mirror image" bound:

Theorem 6.37 *Let $X_1, \ldots, X_n$ be mutually independent random variables, such that X_i is 1 with probability p_i and 0 with probability $q_i = 1 - p_i$, and let $\mu := \sum_{i=1}^{n} p_i$ and $\nu := \sum_{i=1}^{n} p_i q_i$. Then for any $t > 0$, we have*

$$\mathsf{P}[\mu - X \geq t] \leq e^{-t^2/4\nu}.$$

Proof. Let $Y_i = 1 - X_i$ and $Y = \sum_{i=1}^{n} Y_i = n - X$. Then

$$\mu - X = \mu - n + n - X = Y - \mathsf{E}[Y],$$

and so result follows from the previous theorem, applied to the Y_i's. □

Example 6.38 If we toss n coins, and model the outcomes as mutually independent Bernoulli trials, so that each coin toss is "heads" with probability 1/2, then the probability that we obtain at least $n/2 + \epsilon n$ heads is at most $e^{-\epsilon^2 n}$. □

6.6 The Birthday Paradox

This section discusses a number of problems related to the following question: how many people must be in a room before there is a good chance that two of them were born on the same day of the year? The answer is surprisingly few. The "paradox" is that it is in fact far fewer than the number of days in the year, as we shall see.

To answer this question, we index the people in the room with integers $1, \ldots, k$, where k is the number of people in the room. We abstract the problem a bit, and assume that all years have the same number of days, say n — setting $n = 365$ corresponds to the original

problem, except that leap years are not handled correctly, but we shall ignore this detail. For $1 \leq i \leq k$, let X_i denote the day of the year on which i's birthday falls. Let us assume that birthdays are uniformly distributed over $\{0, \ldots, n-1\}$; this assumption is actually not entirely realistic, as it is well known that people are somewhat more likely to be born in some months than in others.

So for any $1 \leq i \leq k$ and $0 \leq x \leq n-1$, we have $\mathsf{P}[X_i = x] = 1/n$.

Let α be the probability that no two persons share the same birthday, so that $1 - \alpha$ is the probability that there is a pair of matching birthdays. We would like to know, how big k must be relative to n so that α is not too large, say, at most $1/2$.

We can compute α as follows, assuming the X_i's are *mutually independent*.

There are a total of n^k sequences of integers $(x_1, \ldots, x_k)$, where each $x_i \in \{0, \ldots, n-1\}$. Among these, there are a total of $n(n-1)\cdots(n-k+1)$ that contain no repetitions: there are n choices for x_1, and for any fixed value of x_1, there are $n-1$ choices for x_2, etc. Therefore

$$\alpha = n(n-1)\cdots(n-k+1)/n^n = \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\cdots\left(1 - \frac{k-1}{n}\right). \tag{6.14}$$

Using the part (1) of §A.6, we obtain

$$\alpha \leq e^{-\sum_{i=1}^{k-1} i/n} = e^{-k(k-1)/2n}.$$

So if $k(k-1) \geq (2\log 2)n$, we have $\alpha \leq 1/2$. Thus, when k is at least a small constant times $n^{1/2}$, we have $\alpha \leq 1/2$, so the probability that two people share the same birthday is at least $1/2$. For $n = 365$, $k \geq 23$ suffices. Indeed, one can simply calculate α in this case numerically from equation (6.14), obtaining $\alpha \approx 0.493$. Thus, if there are 23 people in the room, there is about a 50-50 chance that two people have the same birthday.

The above analysis assumed the X_i's are mutually independent. However, we can still obtain useful upper bounds for α under much weaker independence assumptions.

For $1 \leq i < j \leq k$, let us define the random variable $W_{ij} = 1$ if $X_i = X_j$, and $W_{ij} = 0$ if $X_i \neq X_j$. If we assume that the X_i's are pairwise independent, then

$$\begin{aligned} \mathsf{P}[W_{ij}] &= \mathsf{P}[X_i = X_j] = \sum_{x=0}^{n-1} \mathsf{P}[X_i = x \wedge X_j = x] \\ &= \sum_{x=0}^{n-1} \mathsf{P}[X_i = x]\mathsf{P}[X_j = x] = \sum_{x=0}^{n-1} 1/n^2 = 1/n. \end{aligned}$$

We can compute the expectation and variance:

$$\mathsf{E}[W_{ij}] = \frac{1}{n}, \qquad \mathsf{Var}[W_{ij}] = \frac{1}{n}(1 - \frac{1}{n}).$$

Now consider the random variable

$$W = \sum_{i=1}^{k} \sum_{j=i+1}^{k} W_{ij},$$

which represents the number of distinct pairs of people with the same birthday. There are $k(k-1)/2$ terms in this sum, so by the linearity of expectation, we have

$$\mathsf{E}[W] = \frac{k(k-1)}{2n}.$$

Thus, for $k(k-1) \geq 2n$, we "expect" there to be at least one pair of matching birthdays. However, this does not guarantee that the probability of a matching pair of birthdays is very high, assuming just pairwise independence of the X_i's. For example, suppose that n is prime and the X_i's are a subset of the family of pairwise independent random variables defined in Example 6.23. That is, each X_i is of the form $a_i X + Y$, where X and Y are uniformly and independently distributed modulo n. Then in fact, either all the X_i's are distinct, or they are all equal, where the latter event occurs exactly when $X = [0 \bmod n]$, and so with probability $1/n$ — "'when it rains, it pours."

To get a useful upper bound on α that there are no matching birthdays, it suffices to assume that the X_i's are *4-wise independent*. In this case, it is easy to verify that the variables W_{ij} are *pairwise* independent, since any two of the W_{ij}'s are determined by at most 4 of the X_i's. Therefore, in this case, the variance of the sum is equal to the sum of the variances, and so

$$\mathsf{Var}[W] = \frac{k(k-1)}{2n}(1 - \frac{1}{n}) \leq \mathsf{E}[W].$$

Furthermore, by Chebyshev's Inequality,

$$\alpha = \mathsf{P}[W = 0] \leq \mathsf{P}[|W - \mathsf{E}[W]| \geq \mathsf{E}[W]] \leq \mathsf{Var}[W]/\mathsf{E}[W]^2 \leq 1/\mathsf{E}[W] = \frac{2n}{k(k-1)}.$$

Thus, if $k(k-1) \geq 4n$, then $\alpha \leq 1/2$.

In many practical applications, it is more important to bound α from *below*, rather than from above; that is, to bound from above the probability $1-\alpha$ that there are any collisions. For this, pairwise independence of the X_i's suffices, since Markov's inequality implies that

$$1 - \alpha = \mathsf{P}[W \geq 1] \leq \mathsf{E}[W] = \frac{k(k-1)}{2n},$$

which is at most $1/2$ provided $k(k-1) \leq n$.

Hash functions

The above considerations have numerous applications in computer science. One particularly important application is to the theory and practice of hashing.

The scenario is as follows. We have finite sets $\mathcal{A}$ and $\mathcal{Z}$, with $|\mathcal{A}| = k$ and $|\mathcal{Z}| = n$, and a finite set $\mathcal{H}$ of "hash functions" which map elements of $\mathcal{A}$ into $\mathcal{Z}$. More precisely, each $h \in \mathcal{H}$ defines a function that maps $a \in \mathcal{A}$ to an element $z \in \mathcal{Z}$, and we write $z = h(a)$. Note that two distinct elements of $\mathcal{H}$ may happen to define the same function. Let H be

a random variable whose distribution is uniform on $\mathcal{H}$. For any $a \in \mathcal{A}$, $H(a)$ denotes the random variable whose value is $z = h(a)$ when $H = h$.

For any $1 \leq \ell \leq k$, we say that $\mathcal{H}$ is a ℓ-wise independent family of hash functions (from $\mathcal{A}$ to $\mathcal{Z}$) if each $H(a)$ is uniformly distributed over $\mathcal{Z}$, and the collection of all $H(a)$ is ℓ-wise independent; in case $\ell = 2$, we say that $\mathcal{H}$ is a **pairwise independent family of hash functions**. Pairwise independence is equivalent to saying that for all $a, a' \in \mathcal{A}$, with $a \neq a'$, and all $z, z' \in \mathcal{Z}$,

$$\mathsf{P}[H(a) = z \wedge H(a') = z'] = 1/n^2.$$

Example 6.39 Examples 6.23 and 6.24 provide explicit constructions for pairwise independent families of hash functions. In particular, from the discussion in Example 6.23, if n is prime, and we take $\mathcal{A} = \mathcal{Z} = \mathbb{Z}_n$, and $\mathcal{H} = \mathbb{Z}_n \times \mathbb{Z}_n$, and for $h = (x, y) \in \mathcal{H}$ and $a \in \mathcal{A}$ we define $h(a) = ax + y$, then $\mathcal{H}$ is a pairwise independent family of hash functions from $\mathbb{Z}_n$ to $\mathbb{Z}_n$. Similarly, Example 6.24 yields a pairwise independent family of hash functions from $\mathbb{Z}_n^{\times t}$ to $\mathbb{Z}_n$, with $\mathcal{H} = \mathbb{Z}_n^{\times(t+1)}$. In practice, the inputs to such a hash function may be long bit strings, which we chop into small pieces so that each piece can be viewed as an element of $\mathbb{Z}_n$. $\square$

Families of hash functions such as this may be used to implement "hash tables," which are a data structure used to implement "dictionaries." A random hash function is chosen, and elements of $\mathcal{A}$ are stored in a "bin" indexed by its hash value; likewise, to see if a particular value is stored in the table, one must search in the corresponding bin.

We do not discuss any more detailed implementation issues here. However, one typically wants the number of bins (namely, n) to not be excessively large, while at the same time, one wants the number of elements stored in any bin to not be too large either.

If $\mathcal{H}$ is a pairwise independent family, then one can easily derive some useful results from the above discussion of birthdays.

- For example, if the hash table actually stores some number $k' \leq k$ of values, then for any $a \in \mathcal{A}$, the expected number of values that are in the bin indexed by a's hash value is $1 + (k' - 1)/n$ if a is already in the hash table, and k'/n if it is not in the table. This result bounds the expected amount of "work" we have to do to search for a value in its corresponding bin. In particular, if $k' = O(n)$, then the expected amount of work is constant.

- If $k'(k' - 1) \leq n$, then with probability at least $1/2$, a randomly chosen hash function assigns each of k' distinct values to distinct bins. This result is useful if one wants to find a "perfect" hash function that hashes k' fixed values to distinct bins: if n is sufficiently large, we can just choose hash functions at random until we find one that works.

We leave it as an exercise for the reader to verify the above claims.

There are numerous other interesting questions regarding pairwise independent hash functions and hash tables, but we shall not pursue this matter any further. However, results

such as the ones mentioned above, and others, can be obtained using a broader notion of hashing called **universal hashing**. We call $\mathcal{H}$ a **universal family of hash functions (from $\mathcal{A}$ to $\mathcal{Z}$)** if for all $a, a' \in \mathcal{A}$, with $a \neq a'$,

$$\mathsf{P}[H(a) = H(a')] = 1/n.$$

Note that the pairwise independence property implies the universal property. There are even weaker notions that are relevant in practice; for example, one could just require that $\mathsf{P}[H(a) = H(a')] \leq cn$ for some constant n.

Example 6.40 If we drop the y-value from the first family of hash functions discussed in Example 6.39 so that $\mathcal{H} = \mathbb{Z}_n$, and $x \in \mathbb{Z}_n$ defines the function that sends $a \in \mathbb{Z}_n$ to $ax \in \mathbb{Z}_n$, then we get a universal family of hash functions that is not pairwise independent. The second family of hash functions can be similarly modified to get a universal family of hash functions from $\mathbb{Z}_n^{\times t}$ to $\mathbb{Z}_n$ that is not pairwise independent. □

6.7 Statistical Distance

Let X and Y be random variables which both take values on a (finite) set $\mathcal{V}$. We define the **statistical distance between X and Y** as

$$\Delta[X; Y] := \frac{1}{2} \sum_{v \in \mathcal{V}} |\mathsf{P}[X = v] - \mathsf{P}[Y = v]|.$$

The statistical distance is a useful measure of how similar or dissimilar the distributions of X and Y are.

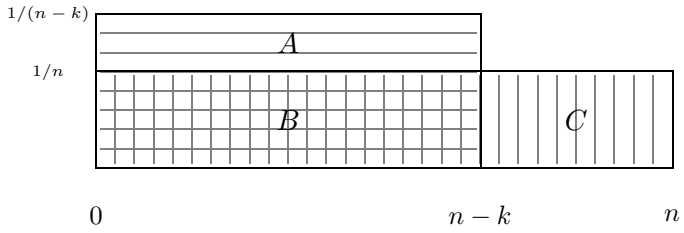
Theorem 6.41 *For random variables X, Y, Z, we have*

1. $0 \leq \Delta[X; Y] \leq 1$,
2. $\Delta[X; X] = 0$,
3. $\Delta[X; Y] = \Delta[Y; X]$, *and*
4. $\Delta[X; Z] \leq \Delta[X; Y] + \Delta[Y; Z]$.

Proof. Exercise. □

Note that $\Delta[X; Y]$ depends only on the individual distributions of X and Y, and not on the joint distribution of X and Y. As such, one may speak of the statistical distance between two distributions, rather than between two random variables.

Example 6.42 Suppose X has the uniform distribution on $\{1, \ldots, n\}$, and Y has the uniform distribution on $\{1, \ldots, n-k\}$, where $0 \leq k \leq n-1$. Let us compute $\Delta[X;Y]$. We could apply the definition directly; however, consider the following graph of the distributions of X and Y:



The statistical distance between X and Y is just $1/2$ times the area of regions A and C in the diagram. Moreover, because probability distributions sum to 1, it must be the case the areas of region A and region C are the same. Therefore,

$$\Delta[X;Y] = \text{area of } A = \text{area of } C = k/n$$

□

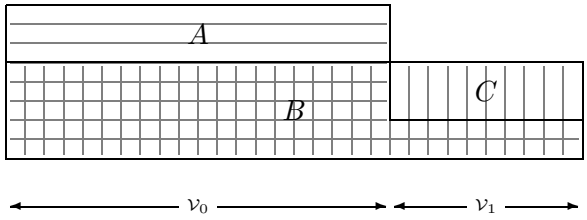
The following characterization of statistical distance is quite useful:

Theorem 6.43 *Let X and Y be random variables taking values on a set $\mathcal{V}$. For any $\mathcal{W} \subset \mathcal{V}$, we have*

$$\Delta[X;Y] \geq |\mathsf{P}[X \in \mathcal{W}] - \mathsf{P}[Y \in \mathcal{W}]|,$$

and equality holds if $\mathcal{W}$ is either the set of all $v \in \mathcal{V}$ such that $\mathsf{P}[X = v] < \mathsf{P}[Y = v]$, or the compliment of this set.

Proof. Suppose we partition the set $\mathcal{V}$ into two sets: the set $\mathcal{V}_0$ consisting of those $v \in \mathcal{V}$ such that $\mathsf{P}[X = v] < \mathsf{P}[Y = v]$, and the set $\mathcal{V}_1$ consisting of those $v \in \mathcal{V}$ such that $\mathsf{P}[X = v] \geq \mathsf{P}[Y = v]$. Consider the following rough graph of the distributions of X and Y, where X is shaded with vertical lines, Y is shaded with horizontal lines, and the elements of $\mathcal{V}_0$ are placed to the left of the elements of $\mathcal{V}_1$:



Now, as in Example 6.42,

$$\Delta[X;Y] = \text{area of } A = \text{area of } C.$$

Further, consider any subset $\mathcal{W}$ of $\mathcal{V}$. The quantity $|\mathsf{P}[X \in \mathcal{W}] - \mathsf{P}[Y \in \mathcal{W}]|$ is equal to the absolute value of the difference of the area of the sub-region of A that lies above $\mathcal{W}$ and the are of the sub-region of C that lies above $\mathcal{W}$. This quantity is maximized when $\mathcal{W} = \mathcal{V}_0$ or $\mathcal{W} = \mathcal{V}_1$, in which case it is equal to $\Delta[X;Y]$. □

This theorem says that when $\Delta[X;Y]$ is very small, for any predicate ϕ, the events $\phi(X)$ and $\phi(Y)$ occur with almost the same probability. Put another way, there is no "statistical test" that can effectively distinguish between the distributions of X and Y. For many applications, this means that the distribution of X is "for all practical purposes" equivalent to that of Y, and hence in analyzing the behavior of X, we can instead analyze the behavior of Y, if that is more convenient.

Theorem 6.44 *Let X, Y be random variables taking values on a set $\mathcal{V}$, and let f be a function from $\mathcal{V}$ into a set $\mathcal{W}$. Then $\Delta[f(X); f(Y)] \leq \Delta[X;Y]$.*

Proof. By Theorem 6.43, for any subset $\mathcal{W}'$ of $\mathcal{W}$, we have

$$|\mathsf{P}[f(X) \in \mathcal{W}'] - \mathsf{P}[f(Y) \in \mathcal{W}']| = |\mathsf{P}[X \in f^{-1}(\mathcal{W}')] - \mathsf{P}[Y \in f^{-1}(\mathcal{W}')]| \leq \Delta[X;Y].$$

In particular, again by Theorem 6.43,

$$\Delta[f(X); f(Y)] = |\mathsf{P}[f(X) \in \mathcal{W}'] - \mathsf{P}[f(Y) \in \mathcal{W}']|$$

for some $\mathcal{W}'$. □

Example 6.45 Let X be uniformly distributed on the set $\{0, \ldots, n-1\}$, and let Y be uniformly distributed on the set $\{0, \ldots, m-1\}$, for $m \geq n$. Let $f(y) = y \bmod n$. We want to compute an upper bound on the statistical distance between X and $f(Y)$. We can do this as follows. Let $m = qn - r$, where $0 \leq r < n$, so that $q = \lceil m/n \rceil$. Also, let Z be uniformly distributed over $\{0, \ldots, qn-1\}$. Then $f(Z)$ is uniformly distributed over $\{0, \ldots, n-1\}$,

since every element of $\{0, \ldots, n-1\}$ has the same number (namely, q) of pre-images under f which lie in the set $\{0, \ldots, qn-1\}$. Therefore, by the previous theorem,

$$\Delta[X; f(Y)] = \Delta[f(Z); f(Y)] \leq \Delta[Z; Y],$$

and as we saw in Example 6.42,

$$\Delta[Z; Y] = r/qn < 1/q \leq n/m.$$

Therefore,

$$\Delta[X; f(Y)] < n/m.$$

□

Another useful fact is the following:

Theorem 6.46 *Let X, Y be random variables taking values on a set $\mathcal{V}$, and let W be a random variable taking values on a set $\mathcal{W}$. Further, suppose that X, W are independent and Y, W are independent. Then*

$$\Delta[X, W; Y, W] = \Delta[X, Y].$$

Proof. From the definition of statistical distance,

$$\begin{aligned}
2\Delta[X, W; Y, W] &= \sum_{v,w} |\mathsf{P}[X = v \wedge W = w] - \mathsf{P}[Y = v \wedge W = w]| \\
&= \sum_{v,w} |\mathsf{P}[X = v]\mathsf{P}[W = w] - \mathsf{P}[Y = v]\mathsf{P}[W = w]| \quad \text{(by independence)} \\
&= \sum_{v,w} \mathsf{P}[W = w]|\mathsf{P}[X = v] - \mathsf{P}[Y = v]| \\
&= (\sum_{w} \mathsf{P}[W = w])(\sum_{v} |\mathsf{P}[X = v] - \mathsf{P}[Y = v]|) \\
&= 1 \cdot 2\Delta[X; Y].
\end{aligned}$$

□

Exercise 6.47 Let X, Y, and Z be uniformly and independently distributed over $\mathbb{Z}_p$, where p is prime. Calculate $\Delta[X, Z; X, XY]$. □

Exercise 6.48 Let X, Y be random variables on a probability distribution, and let $\mathcal{B}_1, \ldots, \mathcal{B}_n$ be events that partition of the underlying sample space. For $1 \leq i \leq n$, let X_i, Y_i denote the random variables X, Y in the conditional probability distribution given by $\mathcal{B}_i$; that is, $\mathsf{P}[X_i = v] = \mathsf{P}[X = v \mid \mathcal{B}_i]$, and similarly for Y_i. Show that

$$\Delta[X; Y] \leq \sum_{i=1}^{n} \Delta[X_i; Y_i]\mathsf{P}[\mathcal{B}_i].$$

□

Exercise 6.49 Consider two random experiments. In the first, we generate a random integer n between 3 and M, and then a random integer w between 1 and n. In the second, we generate a random integer n between 2 and M, and then a random integer w between 2 and $n-1$. Let X denote the outcome (n, w) of the first experiment, and Y the outcome of the second. Show that $\Delta[X;Y] = O(\log M/M)$. □

6.8 ♣ Measures of Randomness and the Leftover Hash Lemma

In this section, we discuss different ways to measure "how random" a probability distribution is, and relations among them. Consider a distribution defined on a finite sample space $\mathcal{V}$. In some sense, the "most random" distribution on $\mathcal{V}$ is the uniform distribution, while the least random would be a "point mass" distribution, i.e., a distribution where one point $v \in \mathcal{V}$ in the sample space has probability 1, and all other points have probability 0.

We define three measures of randomness. Let X be a random variable taking values on a set $\mathcal{V}$ of size N.

1. We say X is **δ-uniform on $\mathcal{V}$** if the statistical distance between X and the uniform distribution on $\mathcal{V}$ is equal to δ, i.e.,
$$\delta = \frac{1}{2}\sum_{v\in\mathcal{V}} |\mathsf{P}[X = v] - 1/N|.$$
2. The **guessing probability** $\gamma(X)$ of X is defined to be
$$\gamma(X) := \max\{\mathsf{P}[X = v] : v \in \mathcal{V}\}.$$
3. The **collision probability** $\kappa(X)$ of X is defined to be
$$\kappa(X) := \sum_{v\in\mathcal{V}} \mathsf{P}[X = v]^2.$$

Observe that if X is uniformly distributed on $\mathcal{V}$, then it is 0-uniform on $\mathcal{V}$, and $\gamma(X) = \kappa(X) = 1/N$. Also, if X has a point mass distribution, then it is $(1-1/N)$-uniform on $\mathcal{V}$, and $\gamma(X) = \kappa(X) = 1$. The quantity $\log_2(1/\gamma(X))$ is sometimes called the **min entropy** of X, and the quantity $\log_2(1/\kappa(X))$ is sometimes called the **Renyi entropy** of X. The collision probability $\kappa(X)$ has the following interpretation: if X and X' are identically distributed independent random variables, then $\kappa(X) = \mathsf{P}[X = X']$.

Before going further, we need the following technical fact:

Theorem 6.50 *If $x_1, \ldots, x_N$ are real numbers with $\sum_{i=1}^N x_i = 1$, then*
$$0 \le \sum_{i=1}^N (x_i - 1/N)^2 = \sum_{i=1}^N x_i^2 - 1/N.$$

In particular,

$$\sum_{i=1}^{N} x_i^2 \geq 1/N.$$

Proof. This follows from a simple calculation:

$$\begin{aligned} 0 &\leq \sum_i (x_i - 1/N)^2 = \sum_i (x_i^2 - 2x_i/N + 1/N^2) = \sum_i x_i^2 - (2/N)(\sum_i x_i) + \sum_i 1/N^2 \\ &= \sum_i x_i^2 - 2/N + 1/N = \sum_i x_i^2 - 1/N. \end{aligned}$$

□

We now state some easy inequalities:

Theorem 6.51 *Let X be a random variable taking values on a set $\mathcal{V}$ of size N, such that X is δ-uniform on $\mathcal{V}$, $\gamma = \gamma(X)$, and $\kappa = \kappa(X)$. Then we have*

1. $\kappa \geq 1/N$,
2. $\gamma^2 \leq \kappa \leq \gamma \leq 1/N + \delta$.

Proof. Part (1) is immediate from Theorem 6.50. The proof of part (2) is left as an easy exercise. □

This theorem implies that the collision and guessing probabilities are minimal for the uniform distribution, which perhaps agrees with ones intuition.

While the above theorem implies that γ and κ are close to $1/N$ when δ is small, the following theorem provides a converse of sorts:

Theorem 6.52 *If X is δ-uniform on $\mathcal{V}$, where $|\mathcal{V}| = N$, and if $\kappa = \kappa(X)$, then*

$$\kappa \geq \frac{1 + 4\delta^2}{N}.$$

Proof. We may assume that $\delta > 0$, since otherwise the theorem is already true, simply from the fact that $\kappa \geq 1/N$.

For $v \in \mathcal{V}$, let $p_v := \mathsf{P}[X = v]$. We have $\delta = \frac{1}{2}\sum_v |p_v - 1/N|$, and hence $1 = \sum_v q_v$, where $q_v := |p_v - 1/N|/(2\delta)$. So we have

$$\begin{aligned} \frac{1}{N} &\leq \sum_v q_v^2 \quad \text{(by Theorem 6.50)} \\ &= \frac{1}{4\delta^2} \sum_v (p_v - 1/N)^2 \\ &= \frac{1}{4\delta^2} (\sum_v p_v^2 - 1/N) \quad \text{(again by Theorem 6.50)} \\ &= \frac{1}{4\delta^2} (\kappa - 1/N), \end{aligned}$$

from which the theorem follows immediately. □

Theorem 6.53 (Leftover Hash lemma) *Let $\mathcal{H}$ be a universal family of hash functions from $\mathcal{A}$ to $\mathcal{Z}$, where $\mathcal{Z}$ is of size n. Let H denote a random variable with the uniform distribution of $\mathcal{H}$, and let A denote a random variable taking values in $\mathcal{A}$, with $\kappa = \kappa(A)$, and with H, A independent. Then $(H, H(A))$ is δ-uniform on $\mathcal{H} \times \mathcal{Z}$, where*

$$\delta \leq \sqrt{n\kappa}/2.$$

In the statement of this theorem, $H(A)$ denotes the random variable whose value is $h(a)$ when $H = h$ and $A = a$.

Proof. Let Z denote a random variable uniformly distributed on $\mathcal{Z}$, with H, A, Z mutually independent. Let $m = |\mathcal{H}|$ and $\delta = \Delta[H, H(A); H, Z]$.

Let us compute the collision probability $\kappa(H, H(A))$. Let H' have the same distribution as H and A' have the same distribution as A, with H, H', A, A' mutually independent. Then

$$\begin{aligned}
\kappa(H, H(A)) &= \mathsf{P}[H = H' \wedge H(A) = H'(A')] \\
&= \mathsf{P}[H = H']\mathsf{P}[H(A) = H(A')] \\
&= \frac{1}{m}\Big(\mathsf{P}[H(A) = H(A') \mid A = A']\mathsf{P}[A = A'] \; + \\
&\qquad \mathsf{P}[H(A) = H(A') \mid A \neq A']\mathsf{P}[A \neq A']\Big) \\
&\leq \frac{1}{m}(\mathsf{P}[A = A'] + \mathsf{P}[H(A) = H(A') \mid A \neq A']) \\
&= \frac{1}{m}(\kappa + 1/n) \\
&= \frac{1}{mn}(n\kappa + 1).
\end{aligned}$$

Applying Theorem 6.52 to the random variable $(H, H(A))$, which takes values on the set $\mathcal{H} \times \mathcal{Z}$ of size $N = mn$, we see that $4\delta^2 \leq n\kappa$. □

Example 6.54 The Leftover Hash Lemma allows one to convert "low quality" sources of randomness into "high quality" sources of randomness. Suppose that to conduct an experiment, we need to sample a random variable Z whose distribution is uniform on a set $\mathcal{Z}$ of size n, or at least δ-uniform for a small value of δ. However, we may not have direct access to a source of "real" randomness whose distribution looks anything like that of the desired uniform distribution, but rather, only to a "low quality" source of randomness. For example, one could model various characteristics of a person's typing at the keyboard, or perhaps various characteristics of the internal state of a computer (both its software and hardware) as a random process. We cannot say very much about the probability distributions associated with such processes, but perhaps we can conservatively estimate the collision or guessing probability associated with these distributions. Using the Leftover

Hash Lemma, we can hash the output of this random process, using a suitably generated random hash function. The hash function acts like a "magnifying glass": it "focuses" the randomness inherent in the "low quality" source distribution onto the set $\mathcal{Z}$, obtaining a "high quality," nearly uniform, distribution on $\mathcal{Z}$.

Of course, this approach requires a random hash function, which may be just as difficult to generate as a random element of $\mathcal{Z}$. The following theorem shows, however, that we can at least use the same "magnifying glass" many times over, with the statistical distance from uniform of the output distribution increasing linearly in the number of applications of the hash function. □

Theorem 6.55 *Let $\mathcal{H}$ be a universal family of hash functions from $\mathcal{A}$ to $\mathcal{Z}$, where $\mathcal{Z}$ is of size n. Let H denote a random variable with the uniform distribution of $\mathcal{H}$, and let $A_1, \ldots, A_\ell$ denote random variables taking values in $\mathcal{A}$, with $\kappa = \kappa(A_i)$ for $1 \leq i \leq \ell$, and with $H, A_1, \ldots, A_\ell$ mutually independent. Then $(H, H(A_1), \ldots, H(A_\ell))$ is $\tilde{\delta}$-uniform on $\mathcal{H} \times \mathcal{Z}^{\times \ell}$, where*

$$\tilde{\delta} \leq \ell\sqrt{n\kappa}/2.$$

Proof. Let $Z_1, \ldots, Z_\ell$ denote random variables with the uniform distribution on $\mathcal{Z}$, with $H, A_1, \ldots, A_\ell, Z_1, \ldots, Z_\ell$ mutually independent. We define random variables $W_0, W_1, \ldots, W_\ell$ as follows: We let

$$\begin{aligned} W_0 &:= (H, H(A_1), \ldots, H(A_\ell)), \\ W_i &:= (H, Z_1, \ldots, Z_i, H(A_{i+1}), \ldots, H(A_\ell)) \quad \text{for } 0 < i < \ell, \text{ and} \\ W_\ell &:= (H, Z_1, \ldots, Z_\ell). \end{aligned}$$

We have

$$\begin{aligned} \tilde{\delta} &= \Delta[W_0; W_\ell] \\ &\leq \sum_{i=1}^{\ell} \Delta[W_{i-1}; W_i] \quad \text{(by part 4 of Theorem 6.41)} \\ &\leq \sum_{i=1}^{\ell} \Delta[H, Z_1, \ldots, Z_{i-1}, H(A_i), A_{i+1}, \ldots, A_\ell; H, Z_1, \ldots, Z_{i-1}, Z_i, A_{i+1}, \ldots, A_\ell] \\ &\qquad \text{(by Theorem 6.44)} \\ &= \sum_{i=1}^{\ell} \Delta[H, H(A_i); H, Z_i] \quad \text{(by Theorem 6.46)} \\ &\leq \ell\sqrt{n\kappa}/2 \quad \text{(by Theorem 6.53).} \end{aligned}$$

□

The above proof is sometimes called a "hybrid argument," as we consider the sequence of "hybrid" variables $W_0, W_1, \ldots, W_\ell$, and show that the distance between each consecutive pair of variables is small.

Example 6.56 Another source of "low quality" randomness arises in certain cryptographic applications, where we have a "secret" random variable A that is distributed uniformly over a large subset of $\mathbb{Z}_p$, but we want to extract from A a "secret key" whose distribution is close to that of the uniform distribution on a specified "key space" $\mathcal{Z}$. The Leftover Hash Lemma allows us to do this, and in fact, it allows us to use a "public" hash function H — generated at random once and for all, and published for all to see. □

Exercise 6.57 Consider again the situation in Theorem 6.53. Suppose that $\mathcal{Z} = \{0, \ldots, n-1\}$, but that we would rather have an almost-uniform distribution over $\mathcal{Z}' = \{0, \ldots, t-1\}$, for some $t < n$. For example, the construction of the universal family of hash functions may require than n is prime, but we we would rather have t be, say, a power of 2, or some other value. While it may be possible to work with a different family of hash functions, we do not have to if n is large enough with respect to t, in which case we can just use the value $H(A) \bmod t$. If Z' is uniformly distributed over $\mathcal{Z}'$, show that

$$\Delta[H, H(A) \bmod t; H, Z'] \leq \sqrt{n\kappa}/2 + t/n.$$

Hint: use a hybrid argument. □

6.9 Discrete Probability Distributions

In addition to working with probability distributions over finite sample spaces, one can also work with distributions over infinite sample spaces. If the sample space is countable, i.e., either finite or *countably* infinite, then the distribution is called a **discrete probability distribution**. We shall not consider any other types of probability distributions in this text. The theory developed in §6.1 through §6.7 extends fairly easily to the countably infinite setting, and in this section, we discuss how this is done.

6.9.1 Basic definitions

To say that the sample space $\mathcal{U}$ is countably infinite simply means that there is a bijection f from the set of positive integers onto $\mathcal{U}$; thus, we can enumerate the elements of $\mathcal{U}$ as u_1, u_2, u_3, etc., where $u_i = f(i)$.

As in the finite case, the probability function assigns to each $u \in \mathcal{U}$ a value $0 \leq \mathsf{P}[u] \leq 1$. The basic requirement that the probabilities sum to one (equation (6.1)) is the requirement that the infinite series $\sum_{i=1}^{\infty} \mathsf{P}[u_i]$ converges to one. Luckily, the convergence properties of an infinite series whose terms are all non-negative is invariant under a re-ordering of terms (see §A.9), so it does not matter how we enumerate the elements of $\mathcal{U}$.

Example 6.58 Suppose we flip a fair coin repeatedly until it comes up "heads," and let the outcome u of the experiment denote the number of coins flipped. We can model this experiment as a discrete probability distribution $\mathbf{D} = (\mathcal{U}, \mathsf{P})$, where $\mathcal{U}$ consists of the set of all positive integers, and where for $u \in \mathcal{U}$, we set $\mathsf{P}[u] = 2^{-u}$. We can check that indeed $\sum_{u=1}^{\infty} 2^{-u} = 1$, as required.

One may be tempted to model this experiment by setting up a probability distribution on the sample space of infinite sequences of coin tosses; however, this sample space is not countably infinite, and so we cannot construct a discrete probability distribution on this space. While it is possible to extend the notion a probability distribution to such spaces, this would take us too far afield. □

Example 6.59 More generally, suppose we repeatedly execute a Bernoulli trial until it succeeds, where each execution succeeds with probability p independently of the previous trials, and let the outcome u of the experiment denote the number of trials executed. Then we associate the probability $\mathsf{P}[u] = q^{u-1}p$ with each positive integer u, where $q = p - 1$, since we have $u-1$ failures before the one success. Such a distribution is called a **geometric distribution**. □

Example 6.60 The series $\sum_{i=1}^{\infty} 1/i^3$ converges to some positive number c. Therefore, we can define a probability distribution on the set of positive integers, where we associate with each $i \geq 1$ the probability $1/ci^3$. □

Example 6.61 More generally, if $x_i, i = 1, 2, \ldots$, are non-negative numbers, and $0 < c = \sum_{i=1}^{\infty} x_i < \infty$, then we can define a probability distribution on the set of positive integers, assigning the probability x_i/c to i. □

As in the finite case, an event is an arbitrary subset $\mathcal{A}$ of $\mathcal{U}$. The probability $\mathsf{P}[\mathcal{A}]$ of $\mathcal{A}$ is defined as the sum of the probabilities associated with the elements of $\mathcal{A}$ — in the definition (6.2), the sum is treated as an infinite series when $\mathcal{A}$ is infinite. This series is guaranteed to converge, and its value does not depend on the particular enumeration of the elements of $\mathcal{A}$.

Example 6.62 Consider the geometric distribution discussed in Example 6.59, where p is the success probability of each Bernoulli trial, and $q = 1 - p$. For $j \geq 1$, consider the event $\mathcal{A}$ that the number of trials executed is at least j. Formally, $\mathcal{A}$ is the set of all integers greater than or equal to j. Intuitively, $\mathsf{P}[\mathcal{A}]$ should be q^{j-1}, since we perform at least j trials if and only if the first $j - 1$ trials fail. Just to be sure, we can compute

$$\mathsf{P}[\mathcal{A}] = \sum_{u \geq j} \mathsf{P}[u] = \sum_{u \geq j} q^{u-1}p = q^{j-1}p \sum_{u \geq 0} q^u = q^{j-1}p \cdot \frac{1}{1-q} = q^{j-1}.$$

□

It is an easy matter to check that all the statements made in §6.1 carry over *verbatim* to the case of countably infinite sample spaces. Moreover, it also makes sense in the countably infinite case to consider events that are a union or intersection of a countably infinite number of events:

Theorem 6.63 *Let $\mathcal{A}_1, \mathcal{A}_2, \ldots$ be an infinite sequence of events.*

1. *If* $\mathcal{A}_i \subset \mathcal{A}_{i+1}$ *for all* $i \geq 1$, *then* $\mathsf{P}[\bigcup_{i\geq 1} \mathcal{A}_i] = \lim_{i\to\infty} \mathsf{P}[\mathcal{A}_i]$.
2. *In general, we have* $\mathsf{P}[\bigcup_{i\geq 1} \mathcal{A}_i] \leq \sum_{i\geq 1} \mathsf{P}[\mathcal{A}_i]$.
3. *If the* $\mathcal{A}_i$*'s are pairwise disjoint, then* $\mathsf{P}[\bigcup_{i\geq 1} \mathcal{A}_i] = \sum_{i\geq 1} \mathsf{P}[\mathcal{A}_i]$.
4. *If* $\mathcal{A}_i \supset \mathcal{A}_{i+1}$ *for all* $i \geq 1$, *then* $\mathsf{P}[\bigcap_{i\geq 1} \mathcal{A}_i] = \lim_{i\to\infty} \mathsf{P}[\mathcal{A}_i]$.

Proof. For (1), let $\mathcal{A} = \cup_{i\geq 1}\mathcal{A}_i$, and let $a_1, a_2, \ldots$ be an enumeration of the elements of $\mathcal{A}$. For any $\epsilon > 0$, there exists a value k_0 such that $\sum_{i=1}^{k_0} a_i > \mathsf{P}[\mathcal{A}] - \epsilon$. Also, there is some k_1 such that $\{a_1, \ldots, a_{k_0}\} \subset \mathcal{A}_{k_1}$. Therefore, for any $k \geq k_1$, we have $\mathsf{P}[\mathcal{A}] - \epsilon < \mathsf{P}[\mathcal{A}_k] \leq \mathsf{P}[\mathcal{A}]$.

(2) and (3) follow by applying (1) to the sequence $\{\bigcup_{j=1}^{i} \mathcal{A}_j\}_i$, and making use of (6.5) and (6.6), respectively.

(4) follows by applying (1) to the sequence $\{\overline{\mathcal{A}}_i\}$, using de Morgen's law. □

6.9.2 Conditional Probability and Independence

All of the definitions and results in §6.2 carry over *verbatim* to the countably infinite case. Equation (6.7) as well as Bayes' Theorem (equation 6.8) and equation (6.9) extend *mutatis mutandus* to the case of an infinite partition $\mathcal{B}_1, \mathcal{B}_2, \ldots$.

6.9.3 Random variables

All of the definitions and results in §6.3 carry over *verbatim* to the countably infinite case (except Theorem 6.18, which of course only makes sense in the finite setting).

6.9.4 Expectation and variance

We define the expected value of a real random variable X exactly as before:

$$\mathsf{E}[X] := \sum_{u\in\mathcal{U}} X(u) \cdot \mathsf{P}[u],$$

where, of course, the sum is an infinite series. However, if X may take negative values, then we require that the series converges *absolutely*; i.e., we require that $\sum_{u\in\mathcal{U}} |X(u)| \cdot \mathsf{P}[u] < \infty$ (see §A.9). Otherwise, we say the the expected value of X **does not exist**. Recall from calculus that a series that converges absolutely will itself converge, and will converge to the same value under a re-ordering of terms. Thus, if the expectation exists at all, its value is independent of the ordering on $\mathcal{U}$. For a non-negative random variable X, if its expectation does not exist, one may express this as "$\mathsf{E}[X] = \infty$."

All of the results in §6.4 carry over essentially unchanged, except that one must pay some attention to "convergence issues."

Equations (6.10) and (6.11) hold, but with the following *caveats* (verify):

- If X is a real random variable, then its expected value $\mathsf{E}[X]$ exists if and only if the series $\sum_{x\in\mathrm{im}(X)} x\mathsf{P}[X = x]$ converges absolutely, in which case $\mathsf{E}[X]$ is equal to the value of the latter series.

- If X is a random variable and f a real-valued function on $\mathrm{im}(X)$, then $\mathsf{E}[f(X)]$ exists if and only if the series $\sum_{x \in \mathrm{im}(X)} f(x)\mathsf{P}[X = x]$ converges absolutely, in which case $\mathsf{E}[f(X)]$ is equal to the value of the latter series.

Example 6.64 Let X be a random variable whose distribution is as in Example 6.60. Since the series $\sum 1/n^2$ converges and the series $\sum 1/n$ diverges, the expectation $\mathsf{E}[X]$ exists, while $\mathsf{E}[X^2]$ does not. □

Theorems 6.26 and 6.27 hold under the additional hypothesis that $\mathsf{E}[X]$ and $\mathsf{E}[Y]$ exist.

If $X_1, X_2, \ldots$ is an infinite sequence of random variables, then the random variable $X = \sum_{i=1}^{\infty} X_i$ is well defined provided the series $\sum_{i=1}^{\infty} X_i(u)$ converges for all $u \in \mathcal{U}$. One might hope that $\mathsf{E}[X] = \sum_{i=1}^{\infty} \mathsf{E}[X_i]$; however, this is not in general true, even if the the individual expectations $\mathsf{E}[X_i]$ are non-negative, and even if the series defining X converges absolutely for all u; nevertheless, it is true when the X_i are non-negative:

Theorem 6.65 *Let $X = \sum_{i \geq 1} X_i$, where and each X_i takes non-negative values only. Then,*

$$\mathsf{E}[X] = \sum_{i \geq 1} \mathsf{E}[X_i].$$

Proof. We have

$$\sum_{i \geq 1} \mathsf{E}[X_i] = \sum_{i \geq 1} \sum_{u \in \mathcal{U}} X_i(u)\mathsf{P}[u] = \sum_{u \in \mathcal{U}} \sum_{i \geq 1} X_i(u)\mathsf{P}[u] = \sum_{u \in \mathcal{U}} \mathsf{P}[u] \sum_{i \geq 1} X_i(u) = \mathsf{E}[X],$$

where we use the fact that we may reverse the order of summation in an infinite double summation of non-negative terms (see §A.10). □

Using this theorem, one can prove the analog of Theorem 6.29 for countably infinite sample spaces, using exactly the same argument.

Theorem 6.66 *If X is a random variable that takes non-negative integer values, then*

$$\mathsf{E}[X] = \sum_{i=1}^{\infty} \mathsf{P}[X \geq i].$$

Example 6.67 To illustrate that Theorem 6.65 does not hold in general, consider the geometric distribution on the positive integers, where $\mathsf{P}[j] = 2^{-j}$ for $j \geq 1$. For $i \geq 1$, define the random variable X_i so that $X_i(i) = 2^i$, $X_i(i+1) = -2^{i+1}$, and $X_i(j) = 0$ for all $j \notin \{i, i+1\}$. Then $\mathsf{E}[X_i] = 0$ for all $i \geq 1$, and so $\sum_{i \geq 1} \mathsf{E}[X_i] = 0$. Now define $X = \sum_{i \geq 1} X_i$. This is well defined, and in fact $X(1) = 2$, while $X(j) = 0$ for all $j > 1$. Hence $\mathsf{E}[X] = 1$. □

The variance $\mathsf{Var}[X]$ of X exists if and only if $\mathsf{E}[X]$ and $\mathsf{E}[(X - \mathsf{E}[X])^2]$ exist, which holds if and only if $\mathsf{E}[X]$ and $\mathsf{E}[X^2]$ exist.

Theorem 6.30 holds under the additional hypothesis that $\mathsf{E}[X]$ and $\mathsf{E}[X^2]$ exist. Similarly, Theorem 6.32 holds under the additional hypothesis that $\mathsf{E}[X_i]$ and $\mathsf{E}[X_i^2]$ exist for each i.

The definition of conditional expectation carries over *verbatim*, as do equations (6.12) and (6.13). The analog of (6.13) for infinite partitions $\mathcal{B}_1, \mathcal{B}_2, \ldots$ does not hold in general, but does hold if X is always non-negative.

6.9.5 Some useful bounds

Both Theorems 6.33 and 6.34 (Markov's and Chebyshev's Inequalities) hold, under the additional hypothesis that the relevant expectations and variances exists.

6.9.6 Statistical Distance

The definitions and results in §6.7 carry over *verbatim*. The notions and results discussed in §6.8 do not have meaningful analogs in the infinite setting.

6.10 Notes

Our proof of Theorem 6.53 (Leftover Hash Lemma), is loosely based on Impagliazzo and Zuckermann [36]. That paper also presents further applications of the leftover Hash Lemma. A very important application of the Leftover Hash Lemma to cryptographic theory may be found in Impagliazzo, Levin, and Luby [35].

Chapter 7

Probabilistic Algorithms

It is sometimes useful to endow our algorithms with the ability to generate random numbers. To simplify matters, we only consider algorithms that generate random bits. Where such random bits actually come from will not be of great concern to us here. In a practical implementation, one would use a pseudo-random bit generator, which should produce bits that "for all practical purposes" are "as good as random." While there is a well-developed theory of pseudo-random bit generation (some of which builds on the ideas in §6.8), we will not delve into this here. Moreover, the pseudo-random bit generators used in practice are not based on this general theory, and are much more *ad hoc* in design. So, although we will present a rigorous formal theory of probabilistic algorithms, the application of this theory to practice is ultimately a bit heuristic.

7.1 Basic Definitions

Formally speaking, we will add a new type of instruction to our Random Access Machine described in §3.2:

random bit This type of instruction is of the form $\alpha \leftarrow \text{RANDOM}$, where α takes the same form as in arithmetic instructions. Execution of this type of instruction assigns to α a value sampled from the uniform distribution on $\{0,1\}$, and independently from the execution of all other random-bit instructions.

In describing algorithms at a high level, we shall write "$b \leftarrow_R \{0,1\}$" to denote the assignment of a random bit to the variable b, and "$s \leftarrow_R \{0,1\}^{\times \ell}$" to denote the assignment of a random bit string of length ℓ to the variable s.

In describing the behavior of such a **probabilistic** or **randomized algorithm** A, for any input x, we view its running time and output as random variables, denoted $T_A(x)$ and $A(x)$, respectively.

Defining the distributions of $T_A(x)$ and $A(x)$ is a bit tricky. If A on input x *always* halts after a finite number of steps, regardless of the outcomes of its random choices, then we can naturally view $T_A(x)$ and $A(x)$ as random variables on a uniform distribution over bit

strings of some particular length. However, there may be no *a priori* bound on the number of steps: think of an algorithm that generates random bits until it generates, say, a 0-bit — just as in Example 6.58, we do not attempt to model this as a probability distribution on the uncountable set of infinite bit strings, but rather, we directly define an appropriate discrete probability distribution that models the execution of A on input x.

To motivate our definition, which may at first seem a bit strange, consider again Example 6.58. We could view the sample space in that example to be the set of all bit strings consisting of zero or more "zero" bits, followed by a single "one" bit, and to each such bit string σ of this special form, we assign the probability $2^{-|\sigma|}$. The "experiment" we have in mind is to generate random bits until one of these special "halting" strings is generated. In developing the definition of the probability distribution for a probabilistic algorithm, we simply consider more general sets of "halting" strings, defined by the algorithm and its input.

To simplify matters just a bit, we assume that the machine produces a stream of random bits, one with every instruction executed, and if the instruction happens to be a random-bit instruction, then this is the bit used by that instruction. For any bit string σ, we can run A on input x for up to $|\sigma|$ steps (where $|\sigma|$ denotes the length of σ), using σ for the stream of random bits, and observe the behavior of the algorithm. In this context, we call σ an **execution path**. Some further terminology will be helpful:

- If A halts within $|\sigma|$ steps, then we call σ a **complete execution path**;
- if A halts in exactly $|\sigma|$ steps, then we call σ an **exact execution path**;
- if A does not halt within $|\sigma|$ steps, then we call σ an **incomplete execution path**;
- if σ is an exact or incomplete execution path, then we may also call it a **partial execution path**.

The sample space $\mathcal{S}$ of the probability distribution associated with A on input x consists of all *exact* execution paths. Clearly, $\mathcal{S}$ is **prefix free**, i.e., no string in $\mathcal{S}$ is a proper prefix of another.

Theorem 7.1 *If $\mathcal{S}$ be a prefix-free set of bit strings, then $\sum_{\sigma\in\mathcal{S}} 2^{-|\sigma|} \leq 1$.*

Proof. We first claim that the theorem holds for any finite prefix-free set $\mathcal{S}$. We may assume that $\mathcal{S}$ is non-empty, since otherwise, the claim is trivial. We prove the claim by induction on the sum of the lengths of the elements of $\mathcal{S}$. The base case is when $\mathcal{S}$ contains just the empty string, in which case the claim is clear. If $\mathcal{S}$ contains non-empty strings, let τ be a string in $\mathcal{S}$ of maximal length, and let τ' be the prefix of length $|\tau| - 1$ of τ. Now remove from $\mathcal{S}$ all strings which have τ' as a prefix (there are either one or two such strings), and add to $\mathcal{S}$ the string τ'. It is easy to see (verify) that the resulting set $\mathcal{S}'$ is also prefix-free, and that

$$\sum_{\sigma\in\mathcal{S}} 2^{-|\sigma|} \leq \sum_{\sigma\in\mathcal{S}'} 2^{-|\sigma|}.$$

The claim now follows by induction.

For the general case, let $\sigma_1, \sigma_2, \ldots$ be a particular enumeration of $\mathcal{S}$, and consider the partial sums $S_i = \sum_{j=1}^{i} 2^{-|\sigma_j|}$ for $i = 1, 2, \ldots$. From the above claim, each of these partial sums is at most 1, from which it follows that $\lim_{i \to \infty} S_i \leq 1$. $\Box$

From the above theorem, if $\mathcal{S}$ is the sample space associated with algorithm A on input x, we have

$$S := \sum_{\sigma \in \mathcal{S}} 2^{-|\sigma|} \leq 1.$$

If $S = 1$, the we say that A **halts with probability** 1 **on input** x, and we define the distribution $\mathbf{D}_{A,x}$ associated with A on input x to be the distribution on $\mathcal{S}$ that assigns the probability $2^{-|\sigma|}$ to the bit string $\sigma \in \mathcal{S}$.

We shall confine ourselves to algorithms that halt with probability 1 on all inputs. However, to analyze a given algorithm, we still have to prove that it halts with probability 1 on all inputs before we can bring to bear all the tools of discrete probability theory.

A simple necessary condition for halting with probability 1 on a given input is that for all incomplete execution paths, there exists some extension that is a complete execution path; indeed, if this does not hold, then with some non-zero probability, the algorithm falls into an infinite loop. This is not, however, a sufficient condition for halting with probability 1. A simple sufficient condition is the following: there exists a bound ℓ (possibly depending on the input) such that for every partial execution path σ, there exists a complete execution path of length at most $|\sigma| + \ell$ that has σ as a prefix. It is usually fairly straightforward to verify this property for a particular algorithm "by inspection."

Example 7.2 Consider the following algorithm:

```
repeat
    b ←_R {0,1}
until b = 1
```

Since every loop is only a constant number of instructions, and since there is one chance to terminate with every loop iteration, the algorithm halts with probability 1. $\Box$

Example 7.3 Consider the following algorithm:

```
i ← 0
repeat
    i ← i + 1
    s ←_R {0,1}^{×i}
until s = 0^{×i}
```

The probability of executing at least n loop iterations is

$$\prod_{i=1}^{n-1}(1-2^{-i}) \geq \prod_{i=1}^{n-1} e^{-2^{-i+1}} = e^{-\sum_{i=0}^{n-2} 2^{-i}} \geq e^{-2},$$

where we have made use of the estimate (3) in §A.6. As this probability does not tend to zero, the algorithm does not halt with probability 1.

Note that every incomplete execution path can be extended to a complete execution path, but the length of the extension is not *a priori* bounded. □

Having defined a probability distribution, we can define $T_A(x)$ and $A(x)$ as random variables on the distribution in the obvious way.

We say that a probabilistic algorithm A runs in **expected polynomial time** if there exist constants c, d such that for all $n \geq 0$ and all inputs x of length n, we have $\mathsf{E}[T_A(x)] \leq n^c + d$. We say that A runs in **strict polynomial time** if there exist constants c, d such that for all n and all inputs x of length n, $\mathsf{P}[T_A(x) \leq n^c + d] = 1$, i.e., it *always* halts in a polynomial number of steps, regardless of its random choices.

Note that in defining expected polynomial time, we are not considering the *input* to be drawn from some probability distribution. One could, of course, define such a notion; however, it is not always easy to come up with a distribution on the input space that reasonably models a particular real-world situation. We do not pursue this issue any more here.

Exercise 7.4 Let $\mathcal{S}$ be a prefix-free set of bit strings with $\sum_{\sigma\in\mathcal{S}} 2^{-|\sigma|} = 1$, and let τ be a bit string that is a prefix of some $\sigma \in \mathcal{S}$. Then if $\mathcal{S}'$ is the subset of of strings in $\mathcal{S}$ which have τ as a prefix, then $\sum_{\sigma\in\mathcal{S}'} 2^{-|\sigma|} = 2^{-|\tau|}$. □

Exercise 7.5 Suppose algorithm A calls algorithm B as a subroutine. In the probability distribution $\mathbf{D}_{A,x}$, consider a particular partial execution path τ that drives A to a point where A invokes algorithm B with a particular input y (determined by x and τ). Consider the conditional probability distribution given by the event that τ is a prefix of A's actual execution path. We can define a random variable X on this conditional distribution whose value is the sub-path traced out by the invocation of subroutine B. Show that the distribution of X is the same as $\mathbf{D}_{B,y}$. Hint: use the previous exercise. □

Exercise 7.6 Let A be a probabilistic algorithm, and for an input x and integer $k \geq 1$, consider the experiment in which we choose a random execution path of length k, and run A on input x for up to k steps using the selected execution path. If A halts within k steps, we define $A_k(x)$ to be the output produced by A, and $T_{A_k}(x)$ to be the actual number of steps executed by A; otherwise, we define $A_k(x)$ to be the distinguished value "$\perp$" and $T_{A_k}(x)$ to be k.

(a) Show that A halts with probability 1 on input x if and only if

$$\lim_{k\to\infty} \mathsf{P}[A_k(x) = \perp] = 0.$$

(b) Show that if A halts with probability 1 on input x, then for all possible outputs y,

$$\mathsf{P}[A(x) = y] = \lim_{k \to \infty} \mathsf{P}[A_k(x) = y].$$

(c) Show that if A halts with probability 1 on input x, then

$$\mathsf{E}[T_A(x)] = \lim_{k \to \infty} \mathsf{E}[T_{A_k}(x)].$$

□

Note that one could simply *define* the output distribution and expected running time of a probabilistic algorithm using the identities of parts (b) and (c) of the above exercise, and thus avoid the construction of an underlying probability distribution. However, without such a probability distribution, we would have very few tools at our disposal to analyze the output distribution and running time of particular algorithms.

To rigorously analyze a the running time and output distributions (or other characteristics) of a probabilistic algorithm in complete detail, one can typically reduce the analysis of some particular infinite event $\mathcal{A} \subset \mathcal{S}$ to the analysis of a countably infinite number of events $\mathcal{B}$, each of which is either finite, or more generally, **finitely determined**, meaning that there exists a $k \geq 0$ such that for any two strings $\tau, \tau' \in \mathcal{S}$ that agree in the first k bit positions, either both are in $\mathcal{B}$ or neither or in $\mathcal{B}$. This means that the event $\mathcal{B}$ is completely determined by the first k bits of the execution path. The reader may easily verify (using Exercise 7.4) the following: the probability of $\mathcal{B}$ is equal to the probability that a randomly selected k-bit string τ agrees with the first $\min\{k, |\sigma|\}$ bits of σ for some $\sigma \in \mathcal{B}$. Thus, the probability of such a finitely determined event may be determined by analyzing a certain event in a finite probability distribution, i.e., by observing the behavior of the algorithm on a random k-bit execution path. Another tool that we can use to analyze probabilistic algorithms is the result of Exercise 7.5, which allows us to analyze the behavior of a algorithm in terms of the behavior of its subroutines.

Exercise 7.7 One can generalize the notion of a discrete, probabilistic process, as follows. Let Γ be a finite or countably infinite set. Let f be a function mapping sequences of one or more elements of Γ to $[0, 1]$, such that the following property holds:

> for all finite sequences $(\gamma_1, \ldots, \gamma_{i-1})$, where $i \geq 1$, $f(\gamma_1, \ldots, \gamma_{i-1}, \gamma)$ is non-zero for at most a finite number of $\gamma \in \Gamma$, and
>
> $$\sum_{\gamma \in \Gamma} f(\gamma_1, \ldots, \gamma_{i-1}, \gamma) = 1.$$

Now consider any prefix-free set $\mathcal{S}$ of finite sequences of elements of Γ. For $\sigma = (\gamma_1, \ldots, \gamma_n) \in \mathcal{S}$, define

$$\mathsf{P}[\sigma] := \prod_{i=1}^{n} f(\gamma_1, \ldots, \gamma_i).$$

Show that $\sum_{\sigma \in \mathcal{S}} \mathsf{P}[\sigma] \leq 1$, and hence we may define a probability distribution on $\mathcal{S}$ using the probability function $\mathsf{P}[\cdot]$ if this sum is 1.

The intuition is that we are modeling a process in which we start out in the "empty" configuration; at each step, if we are in configuration $(\gamma_1, \ldots, \gamma_{i-1})$, we halt if this is a "halting" configuration, i.e., an element of $\mathcal{S}$, and otherwise, we move to configuration $(\gamma_1, \ldots, \gamma_{i-1}, \gamma)$ with probability $f(\gamma_1, \ldots, \gamma_{i-1}, \gamma)$. □

7.2 Approximation of Functions

Suppose f is a function mapping bit strings to bit strings. We may have an algorithm that **approximately computes** f in the following sense: there exists a constant $0 \leq \epsilon < 1/2$, such that for all inputs x, $\mathsf{P}[A(x) = f(x)] \geq 1 - \epsilon$. The value ϵ is a bound on the **error probability**, which is defined as $\mathsf{P}[A(x) \neq f(x)]$. There is a standard "trick" by which one can make the error probability very small; namely, run A on input x some number, say t, times, and take the majority output as the answer. Using Theorem 6.36 (Chernoff Bound), the error probability for the iterated version of A is bounded by

$$e^{-(1/2-\epsilon)^2 t/4\epsilon(1-\epsilon)} \leq e^{-(1/2-\epsilon)^2 t},$$

and so the error probability decreases exponentially with the number of iterations.

If we have an algorithm that runs in expected polynomial time, and which approximately computes a function f, then we can easily turn it into an algorithm that runs in *strict* polynomial time, and also approximates f, as follows. Suppose that $\epsilon < 1/2$ is a bound on the error probability, and $T(n)$ is a polynomial bound on the expected running time for inputs of length n. The new algorithm simply runs the original algorithm for at most $tT(n)$ steps, where t is any constant chosen so that $\epsilon + 1/t < 1/2$ — if the original algorithm does not halt within this time bound, the new algorithm simply halts with an arbitrary output. The probability that the new algorithm errs is at most the probability that the original algorithm errs plus the probability that the original algorithm runs for more than $tT(n)$ steps. By Theorem 6.33 (Markov's inequality), the latter probability is at most $1/t$, and hence the new algorithm approximates f as well.

An important special case of the above is when the output of the function f is either 0 or 1 (or equivalently, *false* or *true*). In this case, f may be viewed as the characteristic function of the language $L := \{x : f(x) = 1\}$. There are several "flavors" of probabilistic algorithms for computing f that are traditionally considered:

- We call a probabilistic, expected polynomial time algorithm an **Atlantic City** algorithm for recognizing L if it approximately computes f with error probability bounded by a constant $\epsilon < 1/2$.
- We call a probabilistic, *strictly* polynomial time algorithm A a **Monte Carlo** algorithm for recognizing L if for some constant $\epsilon > 0$, we have:

  - for any $x \in L$, we have $\mathsf{P}[A(x) = 1] \geq \epsilon$, and
  - for any $x \notin L$, we have $\mathsf{P}[A(x) = 1] = 0$.

- We call a probabilistic, expected polynomial time algorithm a **Las Vegas** algorithm for recognizing L if it computes f correctly on all inputs x.

One also says an Atlantic City algorithm has "two sided" error, a Monte Carlo algorithm has "one sided" error, and a "Las Vegas" algorithm has "zero sided" error.

Exercise 7.8 Show that any language recognized by a Las Vegas algorithm is also recognized by a Monte Carlo algorithm, and that any any language recognized by a Monte Carlo algorithm is also recognized by an Atlantic City algorithm. □

Exercise 7.9 Show that if L has a Monte Carlo algorithm, then it also has a Monte Carlo algorithm A such that $\mathsf{P}[A(x) = 1] \geq 1 - 2^{-n}$ for all inputs $x \in L$ of length n. □

Exercise 7.10 Show that a language is recognized by a Las Vegas algorithm iff the language and its compliment are recognized by Monte Carlo algorithms. □

7.3 Flipping a Coin until a Head Appears

In this and subsequent sections of this chapter, we discuss a number of specific probabilistic algorithms, starting with the algorithm in Example 7.2 (which takes no input). We have already established that it halts with probability 1.

Let X be a random variable that represents the number of loop iterations made by the algorithm. Further, define random variables $B_1, B_2, \ldots$, where B_i represents the value of the bit assigned to b in the ith loop iteration, if $X \geq i$, and 0 otherwise. Clearly, exactly one B_i will take the value 1, and all others the value 0, in which case X takes the value i.

It need not be the case that the values of the B_i's are located at pre-determined positions of the execution path. Perhaps for this particular algorithm, one could carefully program the algorithm so that this were the case, but we do not want to make such assumptions in general. Nevertheless, for any $i \geq 1$, if we condition on an any particular partial execution path τ that drives the algorithm to the point where it is just about to sample the bit B_i, then in this conditional probability distribution, B_i is uniformly distributed over $\{0, 1\}$. To prove this rigorously in our formal framework, define the event $\mathcal{A}_\tau$ to be the event that τ is a prefix of the execution path. If $|\tau| = \ell$, then the events $\mathcal{A}_\tau$, $\mathcal{A}_\tau \wedge (B_i = 0)$, and $\mathcal{A}_\tau \wedge (B_i = 1)$ are all finitely determined, and in particular, are determined by the first $\ell + 1$ bits of the execution path. We can then consider corresponding events in a probabilistic experiment wherein we observe the behavior of the algorithm on a random $(\ell + 1)$-bit execution path. In the latter experiment, it is clear that the conditional probability distribution of B_i, given that the first ℓ bits of the actual execution path σ agree with τ, is uniform over $\{0, 1\}$, and thus, the same holds in the original probability distribution. Since this holds for all relevant τ, it follows that it holds conditioned on $X \geq i$.

From the above discussion, it follows that $\mathsf{P}[B_1 = 1] = 1/2$, $\mathsf{P}[B_2 = 1] = \mathsf{P}[B_2 = 1 \mid B_1 = 0]\mathsf{P}[B_1 = 0] = 1/4$, and in general, $\mathsf{P}[B_i = 1] = 2^{-i}$, for $i = 1, 2, \ldots$. Thus, X has a geometric distribution, with $\mathsf{P}[X = i] = 2^{-i}$ for $i = 1, 2, \ldots$.

Let Y denote the total running time of the algorithm. Then $Y \leq cX + d$ for some constants c and d, and hence

$$\mathsf{E}[Y] \leq c\mathsf{E}[X] + d = 2c + d,$$

and we conclude that the expected running time of the algorithm is a constant, the exact value of which depends on the details of the implementation.

All of these conclusions were perhaps obvious, but the main point was to illustrate how we can rigorously prove such statements in our formal model by reducing the analysis from the infinite setting to the finite setting.

7.4 Generating a Random Number from a Given Interval

Suppose we want to generate a number n uniformly at random from the interval $\{0, \ldots, M-1\}$, for a given value of $M \geq 1$.

If M is a power of 2, say $M = 2^k$, then we can do this directly as follows: generate a random k-bit string s, and convert s to the integer $I(s)$ whose base-2 representation is s, i.e., if $s = b_{k-1}b_{k-1}\cdots b_0$, where the b_i's are bits, then

$$I(s) := \sum_{i=0}^{k-1} b_i 2^i.$$

In the general case, we do not have a direct way to do this, since we can only directly generate random bits. However, suppose that M is a k-bit number, so that $2^{k-1} \leq M < 2^k$. Then the following algorithm does the job:

Algorithm RN:

repeat
    $s \leftarrow_R \{0,1\}^{\times k}$
    $n \leftarrow I(s)$
until $n < M$
output n

In every loop iteration, n is uniformly distributed over $\{0, \ldots, 2^k - 1\}$, and the event $n < M$ occurs with probability $M/2^k \geq 1/2$; moreover, conditioning on the latter event, n is uniformly distributed over $\{0, \ldots, M-1\}$. Therefore, if X denotes the number of iterations of the main loop, and if N denotes the output of the algorithm, we conclude that X has a geometric distribution with an associated success probability of $M/2^k \geq 1/2$, that N has the uniform distribution over $\{0, \ldots, M-1\}$, and that X and N are independent.

Further, if Y denotes the running time of the algorithm, then we also may conclude that $\mathsf{E}[X] = O(1)$ and $\mathsf{E}[Y] = O(k)$.

In the above analysis, we have not gone in to all the details as we did in §7.3. Similarly as to what was done there, one would define random variables N_i representing the value of n in the ith loop iteration. Then, one would consider various conditional distributions, conditioning on particular partial execution paths τ that bring the computation just to the beginning of the ith loop iteration; for any particular such τ, the ith loop iteration will terminate in at most $\ell := |\tau| + c$ steps, for some constant c (which depends on k, but not τ). Therefore, the conditional distribution of N_i, given the partial execution path τ, can be analyzed by considering the execution of the algorithm on a random ℓ-bit execution path. It is then clear that the conditional distribution of N_i given the partial execution path τ is uniform over $\{0, \ldots, 2^k - 1\}$, and since this holds for all relevant τ, it follows that the conditional distribution of N_i, given that the ith loop is entered, is uniform over $\{0, \ldots, 2^k - 1\}$. The output distribution is the same as the conditional distribution of N_i, given that $X = i$, which is precisely the uniform distribution on $\{0, \ldots, M - 1\}$.

This sketch of the details again shows how one can formally reduce questions regarding the discrete distribution to questions regarding appropriate finite distributions.

Of course, by adding an appropriate value to the output, we can generate random numbers uniformly in an interval $\{A, \ldots, B\}$, for given A and B. In the what follows, we shall denote the execution of this algorithm as

$$n \leftarrow_R \{A, \ldots, B\}.$$

We also mention the following alternative approach to generating a random number from an interval. Given a positive k-bit integer M, and a parameter $t > 0$, we do the following:

Algorithm RN′:

$s \leftarrow_R \{0,1\}^{\times(k+t)}$
$n \leftarrow I(s) \bmod M$
output n

Compared with algorithm RN, algorithm RN′ the advantage that there are no loops — it halts in an *a priori* bounded number of steps; however, it has the disadvantage that its output is *not* uniformly distributed over the interval $\{0, \ldots, M-1\}$. However, the statistical distance between its output distribution and the uniform distribution on $\{0, \ldots, M-1\}$ is at most 2^{-t} (see Example 6.45 in §6.7). Thus, by choosing t suitably large, we can make the output distribution "as good as uniform" for most practical purposes.

Exercise 7.11 Prove that no probabilistic algorithm that always halts in a bounded number of steps can have an output distribution that is uniform on $\{0, \ldots, M-1\}$, unless M is a power of 2. □

Exercise 7.12 Design and analyze an efficient probabilistic algorithm that takes as input an integer $M \geq 2$, and outputs a random element of $\mathbb{Z}_M^*$. $\square$

7.5 Generating a Random Prime

Suppose we are given an integer $M \geq 2$, and want to generate a random prime between 1 and M. One way to proceed is simply to generate random numbers until we get a prime. This idea will work, assuming the existence of an efficient algorithm $IsPrime(\cdot)$ that determines whether or not its input is prime.

Now, the most naive method of testing if a number n is prime is to see of any of the numbers between 2 and $n-1$ divide n. Of course, one can be slightly more clever, and only perform this divisibility check for prime numbers between 2 and $\sqrt{n}$. Nevertheless, such an approach does not give rise to a polynomial-time algorithm. Indeed, the design and analysis of efficient primality tests has been an active research area for many years. There is, in fact, a deterministic, polynomial-time algorithm for testing primality, which we shall discuss in a later chapter. For the moment, we shall just assume we have such an algorithm, and use it as a "black box."

Our algorithm to generate a random prime between 1 and M runs as follows:

Algorithm RP:

repeat
    $n \leftarrow_R \{1, \ldots, M\}$
until $IsPrime(n)$
output n

Consider a single loop iteration of algorithm RP. For any fixed prime p between 1 and M, the probability that the variable n takes the value p is precisely $1/M$. Thus, every prime is equally likely, and the probability that n is a prime is precisely $\pi(M)/p$. When this algorithm terminates, its output will be uniformly distributed over the set of all primes between 1 and M. If T denotes the number of loop iterations performed by the algorithm, then $\mathsf{E}[T] = M/\pi(M)$, which by Theorem 5.2 (Chebyshev's Theorem), is $\Theta(k)$, where $k := \mathrm{len}(M)$.

So we have bounded the expected number of loop iterations. We now want to bound the expected overall running time. Let μ be the expected running time of the body of the main loop of algorithm RP. Let X_i denote the running time of the ith loop iteration of algorithm RP, so that $X = \sum_{i \geq 1} X_i$ is the total running time. Then we have

$$\mathsf{E}[X] = \sum_{i \geq 1} \mathsf{E}[X_i] = \sum_{i \geq 1} \mathsf{E}[X_i \mid T \geq i]\mathsf{P}[T \geq i] = \mu \sum_{i \geq 1} \mathsf{P}[T \geq i] = \mu \mathsf{E}[T].$$

So it suffices to estimate μ.

To this end, define W_n to be the running time of algorithm *IsPrime* on input n. Also, define

$$W'_M := \frac{1}{M}\sum_{n=1}^{M} W_n.$$

That is, W'_M is the average value of W_n, for a random choice of $n \in \{1, \ldots, M\}$. Thus, μ is equal to W'_M plus the expected running time of algorithm RN, which is $O(k)$, plus any other small overhead, which is also $O(k)$. So we have $\mu \leq W'_M + O(k)$, and assuming that $W'_M = \Omega(k)$, which is perfectly reasonable, we have $\mu = O(W'_M)$. We conclude that the expected running time of algorithm RP is

$$\mathsf{E}[X] = \mu\mathsf{E}[T] = O(kW'_M).$$

Instead of generating a random prime between 1 and M, we may instead want to generate a random k-bit prime, i.e., a prime between 2^{k-1} and $2^k - 1$. Theorem 5.11 (Bertrand's Postulate) tells us that there exist such primes for every k, and that in fact, there are $\Omega(2^k/k)$ such primes. Because of this, it is trivial to modify algorithm RP to generate a random k-bit prime. We leave the details of this to the reader.

7.5.1 Using a probabilistic primality test

In the above analysis, we assumed that *IsPrime* was a deterministic, polynomial time algorithm. While such an algorithm exists, there are in fact simpler and more efficient algorithms that are probabilistic. We shall discuss such algorithms in greater depth later. A number of these algorithms have "one sided" error in the following sense: if the input n is a prime, the output is certainly "true"; however, if the input n is composite, the output will be "false" with high probability, but may be "true" with some small error probability bounded by ϵ. The value of ϵ may be easily "tuned" by adjusting a parameter of the algorithm — indeed, it will turn out that we can make ϵ essentially as small as we like, without too much extra computational effort.

Let us analyze the behavior of algorithm RP under the assumption that *IsPrime* is implemented by a probabilistic algorithm as described in the previous paragraph, with an error probability for composite inputs bounded by ϵ. Let us define W_n to be the expected running tine of *IsPrime* on input n, and as before, we define

$$W'_M := \frac{1}{M}\sum_{n=1}^{M} W_n.$$

Thus, W'_M is the expected running time of algorithm *IsPrime*, where the average is taken with respect to randomly chosen n *and* the random choices of the algorithm itself.

Consider a single loop iteration of algorithm RP. For any fixed prime p between 1 and M, the probability that n takes the value p is $1/M$. Thus, if the algorithm halts with a prime, every prime is equally likely, and the probability that it halts at all is at least $\pi(M)/M$ — the algorithm may also halt with a composite value of n if the primality test

makes a mistake. So we see that the expected number of loop iterations should be no more than in the case where we use a deterministic primality test. Using the same argument as was used before to estimate the expected total running time of algorithm RP, we find that this is $O(kW'_M)$, where $k := \mathrm{len}(M)$. As for the probability that algorithm RP mistakenly outputs a composite, one might be tempted to say that this probability is at most ϵ, the probability that *IsPrime* makes a mistake. However, drawing such a conclusion, we would be committing the fallacy of Example 6.12.

Let us be a bit more precise. Again, consider the probability distribution defined by a single loop iteration, and let $\mathcal{A}$ be the event that *IsPrime* outputs *true*, and $\mathcal{B}$ the event that n is composite. Let $\beta := \mathsf{P}[\mathcal{B}]$ and $\alpha := \mathsf{P}[\mathcal{A} \mid \mathcal{B}]$. First, observe that $\alpha \leq \epsilon$. Now, the probability δ that the algorithm halts and outputs a composite in this loop iteration is

$$\delta = \mathsf{P}[\mathcal{A} \wedge \mathcal{B}] = \alpha\beta.$$

The probability δ' that the algorithm halts and outputs either a prime or composite is

$$\delta' = \mathsf{P}[\mathcal{A}] = \mathsf{P}[\mathcal{A} \wedge \mathcal{B}] + \mathsf{P}[\mathcal{A} \wedge \overline{\mathcal{B}}] = \mathsf{P}[\mathcal{A} \wedge \mathcal{B}] + \mathsf{P}[\overline{\mathcal{B}}] = \alpha\beta + (1 - \beta).$$

It follows that, with T being the number of loop iterations as before,

$$\mathsf{E}[T] = \frac{1}{\delta'} = \frac{1}{\alpha\beta + (1 - \beta)}, \tag{7.1}$$

and hence

$$\mathsf{E}[T] \leq \frac{1}{(1 - \beta)} = \frac{M}{\pi(M)} = O(k).$$

Let us now consider the probability γ that the output of algorithm RP is composite. We have

$$\gamma = \sum_{i \geq 1} \delta \mathsf{P}[T \geq i] = \delta \mathsf{E}[T] = \frac{\alpha\beta}{\alpha\beta + (1 - \beta)}, \tag{7.2}$$

and hence

$$\gamma \leq \frac{\alpha}{(1 - \beta)} \leq \frac{\epsilon}{(1 - \beta)} = \epsilon \frac{M}{\pi(M)} = O(k\epsilon).$$

Another way of analyzing the output distribution of algorithm RP is to consider its statistical distance Δ from the uniform distribution on the set of primes between 1 and M. As we have already argued, every prime between 1 and M is equally likely to be output, and in particular, any fixed prime p is output with probability at most $1/\pi(M)$. It follows from Theorem 6.43 that $\Delta = \gamma$.

Exercise 7.13 Analyze algorithm RP assuming that the primality test is implemented by an "Atlantic City" algorithm with error probability at most ϵ. □

Exercise 7.14 Consider the following probabilistic algorithm that takes as input a positive integer M:

$S \leftarrow \{\ \}$
repeat
    $n \leftarrow_R \{1, \ldots, M\}$
    $S \leftarrow S \cup \{n\}$
until $|S| = M$

Show that the expected number of iterations of the main loop is $\sim M \log M$. □

7.6 Generating a Random Non-Increasing Sequence

The following algorithm will be used below as a fundamental subroutine in a beautiful algorithm that generates random numbers in *factored form*. The algorithm takes as input an integer $M \geq 2$, and runs as follows:

Algorithm RS:

$n_0 \leftarrow M$
$i \leftarrow 0$
repeat
    $i \leftarrow i + 1$
    $n_i \leftarrow_R \{1, \ldots, n_{i-1}\}$
until $n_i = 1$
$t \leftarrow i$
Output $(n_1, \ldots n_t)$

That the algorithm halts with probability 1 is clear, since in every loop iteration, the algorithm picks $n_i = 1$ with probability at least $1/M$, and immediately terminates.

We analyze first the output distribution, and then the running time.

7.6.1 Analysis of the output distribution

Let $N_1, N_2, \ldots$ be random variables denoting the choices of n_i's (for completeness, define $N_i := 1$ if loop i is never entered).

A particular output of the algorithm is a non-increasing chain $(n_1, \ldots, n_t)$, where $n_1 \geq n_2 \geq \cdots \geq n_{t-1} > n_t = 1$. For any such chain, we have

$$\begin{aligned} \mathsf{P}[N_1 = n_1 \wedge \cdots \wedge N_t = n_t] &= \mathsf{P}[N_1 = n_1]\mathsf{P}[N_2 = n_2 \mid N_1 = n_1] \cdots \\ &\quad \mathsf{P}[N_t = n_t \mid N_1 = n_1 \wedge \cdots \wedge N_{t-1} = n_{t-1}] \\ &= \frac{1}{M} \cdot \frac{1}{n_1} \cdot \cdots \cdot \frac{1}{n_{t-1}}. \end{aligned} \tag{7.3}$$

This completely describes the output distribution, in the sense that we have determined the probability with which each non-increasing chain appears as an output. However, there is another way to characterize the output distribution that is significantly more useful. For $2 \leq j \leq M$, define the random variable E_j to be the number of occurrences of j among the N_i's. The E_j's determine the N_i's, and *vice versa*. Indeed, $E_M = e_M, \ldots, E_2 = e_2$ iff the output of the algorithm is the non-increasing chain

$$(\underbrace{M, \ldots, M}_{e_M \text{ times}}, \underbrace{M-1, \ldots, M-1}_{e_{M-1} \text{ times}}, \ldots, \underbrace{2, \ldots, 2}_{e_2 \text{ times}}, 1).$$

From (7.3), we can therefore directly compute

$$\mathsf{P}[E_M = e_M \wedge \ldots \wedge E_2 = e_2] = \frac{1}{M} \prod_{j=2}^{M} \frac{1}{j^{e_j}}. \tag{7.4}$$

Notice that we can write $1/M$ as a telescoping product:

$$\frac{1}{M} = \frac{M-1}{M} \cdot \frac{M-1}{M-2} \cdot \cdots \cdot \frac{2}{3} \cdot \frac{1}{2} = \prod_{j=2}^{M} (1 - 1/j),$$

so we can re-write (7.4) as

$$\mathsf{P}[E_M = e_M \wedge \ldots \wedge E_2 = e_2] = \prod_{j=2}^{M} j^{-e_j}(1 - 1/j). \tag{7.5}$$

Notice that for $2 \leq j \leq M$,

$$\sum_{e_j \geq 0} j^{-e_j}(1 - 1/j) = 1,$$

and so by the discrete version of Theorem 6.17, the E_j's are mutually independent, and for any $2 \leq j \leq M$ and $e_j \geq 0$, we have

$$\mathsf{P}[E_j = e_j] = j^{-e_j}(1 - 1/j), \tag{7.6}$$

In summary, we have shown that the E_j's are mutually independent variables, where for $2 \leq j \leq M$, the variable $E_j + 1$ has a geometric distribution with an associated success probability of $1 - 1/j$.

Another, perhaps more intuitive, analysis of the joint distribution of the E_j's runs as follows. Conditioning on the event $E_M = e_M, \ldots, E_{j+1} = e_{j+1}$, one sees that the value of E_j is the number of times the value j appears in the sequence $N_i, N_{i+1}, \ldots$, where $i = e_M + \cdots + e_{j+1} + 1$; moreover, in this conditional probability distribution, it is not too hard to convince oneself that N_i is uniformly distributed over $\{1, \ldots, j\}$. Hence the probability that $E_j = e_j$ in this conditional probability distribution is the probability of getting a run of exactly e_j copies of the value j in an experiment in which we successively choose numbers between 1 and j at random, and this latter probability is clearly $j^{-e_j}(1 - 1/j)$.

7.6.2 Analysis of the running time

Let T be the random value that takes the value t when the output is $(n_1, \ldots, n_t)$. Clearly, it is the value of T that essentially determines the running time of the algorithm.

With the random variables E_j defined as above, we see that $T = 1 + \sum_{j=2}^{M} E_j$. Moreover, for each j, $E_j + 1$ has a geometric distribution with associated success probability $1 - 1/j$, and hence

$$\mathsf{E}[E_j] = \frac{1}{1 - 1/j} - 1 = \frac{1}{j-1}.$$

Thus,

$$\mathsf{E}[T] = 1 + \sum_{j=2}^{M} \mathsf{E}[E_j] = 1 + \sum_{j=1}^{M-1} \frac{1}{j} = \int_1^M \frac{dy}{y} + O(1) \sim \log M.$$

Intuitively, this is roughly as we would expect, since with probability $1/2$ each successive n_i is at most one half as large as its predecessor, and so after $O(\mathrm{len}(M))$ steps, we expect to reach 1.

To complete the running time analysis, let us consider the total number of times X that the main loop of algorithm RN in §7.4 is executed. For $i = 1, 2, \ldots$, let X_i denote the number of times that loop is executed in the ith loop of algorithm RS, defining this to be zero if the ith loop is never reached. So $X = \sum_{i=1}^{\infty} X_i$. Arguing just as in §7.5, we have

$$\mathsf{E}[X] = \sum_{i \geq 1} \mathsf{E}[X_i] \leq 2 \sum_{i \geq 1} \mathsf{P}[T \geq i] = 2\mathsf{E}[T] \sim 2 \log M.$$

To finish, if Y denotes the running time of algorithm RS on input M, then we have $Y \leq c\,\mathrm{len}(M)(X+1)$ for some constant c, and hence $\mathsf{E}[Y] = O(\mathrm{len}(M)^2)$.

Exercise 7.15 Show that when algorithm RS runs on input M, the expected number of (not necessarily distinct) primes in the output sequence is $\sim \log\log M$. □

Exercise 7.16 For $2 \leq j \leq M$, let $F_j = 1$ if the j appears in the output of algorithm RS on input M, and 0 otherwise. Determine the joint distribution of the F_j's. Using this, show that the expected number of distinct primes appearing in the output sequence is $\sim \log\log M$. □

Exercise 7.17 Design and analyze a simple probabilistic algorithm that runs in expected constant time, and whose output distribution is a random variable X taking integer values $M = 1, 2, \ldots$ such that $\mathsf{P}[X = M] = \Theta(1/M^3)$. □

7.7 Generating a Random Factored Number

We now present an efficient algorithm that generates a random factored number. That is, on input $M \geq 2$, the algorithm generates a number r uniformly distributed over the interval

$\{1, \ldots, M\}$, but instead of the usual output format for such a number r, the output consists of the prime factorization of r.

As far as anyone knows, there are no efficient algorithms for factoring large numbers, despite years of active research in search of such an algorithm. So our algorithm to generate a random factored number will *not* work by generating a random number and then factoring it.

Our algorithm will use algorithm RS in §7.6 as a subroutine. In addition, as we did in §7.5, we shall assume the existence of an deterministic, polynomial-time primality test *IsPrime*$(\cdot)$. We denote its running time on input n by W_n, and set $W_M^* := \max\{W_n : 2 \leq n \leq M\}$.

In the analysis of the algorithm, we shall make use of Mertens' Theorem, which we proved in §5 (Theorem 5.17).

On input $M \geq 2$, the algorithm to generate a random factored number $r \in \{1, \ldots, M\}$ runs as follows:

Algorithm RFN:

repeat
    Run algorithm RS on input M, obtaining $(n_1, \ldots, n_t)$
(*) Let $n_{i_1}, \ldots, n_{i_\ell}$ be the primes among $n_1, \ldots, n_t$, including duplicates
(**) Set $r \leftarrow \prod_{i=1}^{\ell} n_{i_j}$
    If $r \leq M$ then
        $s \leftarrow_R \{1, \ldots, M\}$
        if $s \leq r$ then output $n_{i_1}, \ldots, n_{i_\ell}$ and halt
forever

(*) Each n_i is tested for primality in turn using algorithm *IsPrime*$(\cdot)$.

(**) We assume that the product is computed by a simple iterative procedure that halts as soon as the partial product exceeds M. This ensures that the time spent forming the product is always $O(\text{len}(M)^2)$, which simplifies the analysis.

Now, let $1 \leq n \leq M$ be a fixed integer, and let us calculate the probability that the variable r takes the particular value n in any one loop iteration. Let $n = \prod_{p \leq M} p^{e_p}$ be the prime factorization of n. Then r takes the value n iff $E_p = e_p$ for all primes $p \leq M$, which by the analysis in §7.6, happens with probability precisely

$$\prod_{p \leq M} p^{-e_p}(1 - 1/p) = \frac{U(M)}{n},$$

where

$$U(M) := \prod_{p \leq M} (1 - 1/p).$$

Now, the probability that any one loop iteration produces n as output is equal to the probability that r takes the value n *and* $s \leq n$, which is

$$\frac{U(M)}{n} \cdot \frac{n}{M} = \frac{U(M)}{M}.$$

Thus, every n is equally likely, and summing over all $n \in \{1, \ldots, M\}$, we see that the probability that any one loop iteration succeeds in producing some output is $U(M)$.

It follows from the above that the output distribution is as required, and if H denotes the number of loop iterations of the algorithm, then $\mathsf{E}[H] = U(M)^{-1}$, which by Theorem 5.17 is $O(k)$, where $k = \mathrm{len}(M)$.

To finish the running time analysis, consider the expected running time of the loop body. From the analysis in §7.6, it is easy to see that this is $O(kW_M^*)$. It follows that the expected total running time is $O(k^2 W_M^*)$.

7.7.1 Using a probabilistic primality test

Analogous to the discussion in §7.5.1, we can analyze the behavior of algorithm RFN under the assumption that *IsPrime* is a probabilistic algorithm which may erroneously indicate that a composite number is prime with probability bounded by ϵ. Here, we assume that W_n denotes the expected running time of the primality test on input n, and set $W_M^* := \max\{W_n : 2 \leq n \leq M\}$.

The situation here is a bit more complicated than in the case of algorithm RP, since an erroneous output of the primality test in algorithm RFN could lead either to the algorithm halting prematurely (with a wrong output), or to the algorithm being delayed (because an opportunity to halt may be missed).

Let us first analyze in detail the behavior of a single loop iteration of algorithm RFN. Let $\mathcal{A}$ denote the event that the primality test makes a mistake in this loop iteration, and let $\delta := \mathsf{P}[\mathcal{A}]$. If T is the number of loop iterations on a given run of algorithm RS, it is easy to see that

$$\delta \leq \epsilon \, \mathsf{E}[T] = \epsilon \, \ell(M),$$

where

$$\ell(M) := 1 + \sum_{j=1}^{M-1} \frac{1}{j} \leq 2 + \log M.$$

Now, let $1 \leq n \leq M$ be a fixed integer, and let us calculate the probability α_n that the correct prime factorization of n is output in this loop iteration. Let $\mathcal{B}_n$ be the event that the primes among the output of algorithm RS multiply out to n. Then $\alpha_n = \mathsf{P}[\mathcal{B}_n \wedge \overline{\mathcal{A}}](n/M)$. Moreover, because of the mutual independence of the E_j's, not only does it follow that $\mathsf{P}[\mathcal{B}_n] = U(M)/n$, but it also follows that $\mathcal{B}_n$ and $\mathcal{A}$ are independent events: to see this, note that $\mathcal{B}_n$ is determined by the variables $\{E_j : j \text{ prime}\}$, and $\mathcal{A}$ is determined by the variables $\{E_j : j \text{ composite}\}$ and the random choices of primality test. Hence,

$$\alpha_n = \frac{U(M)}{M}(1 - \delta).$$

Thus, every n is equally likely to be output. If $\mathcal{C}$ is the event that the algorithm halts with *some* output (correct or not) in this loop iteration, then

$$\mathsf{P}[\mathcal{C}] \geq U(M)(1-\delta), \tag{7.7}$$

and

$$\mathsf{P}[\mathcal{C} \vee \mathcal{A}] = U(M)(1-\delta) + \delta = U(M) - \delta U(M) + \delta \geq U(M). \tag{7.8}$$

The expected running time of a single loop iteration of algorithm RFN is also easily seen to be $O(kW_M^*)$. That completes the analysis of a single loop iteration.

We next analyze the total running time of algorithm RFN. If H is the number of loop iterations of algorithm RFN, it follows from (7.7) that

$$\mathsf{E}[H] \leq \frac{1}{U(M)(1-\delta)},$$

and assuming that $\epsilon\ell(M) \leq 1/2$, it follows that the expected running time of algorithm RFN is $O(k^2 W_M^*)$.

Finally, we analyze the statistical distance Δ between the output distribution of algorithm RFN and the uniform distribution on the numbers 1 to M, in correct factored form. Let H' denote the first loop iteration i for which the event $\mathcal{C} \vee \mathcal{A}$ occurs, i.e., the algorithm either halts or the primality test makes a mistake. Then, by (7.8), H' has a geometric distribution with an associated success probability of at least $U(M)$. If $\mathcal{A}^*$ is the event that the primality test makes a mistake in any loop iteration, then

$$\mathsf{P}[\mathcal{A}^*] = \sum_{i \geq 1} \delta \mathsf{P}[H' \geq i] = \delta \mathsf{E}[H'] \leq \delta U(M).$$

Now, if γ be the probability that the output of algorithm RFN is not in correct factored form, then

$$\gamma \leq \mathsf{P}[\mathcal{A}^*] = \delta U(M) = O(k^2 \epsilon).$$

We have already argued that each value n between 1 and M, in correct factored form, is equally likely to be output, and in particular, each such value occurs with probability at most $1/M$. It follows from Theorem 6.43 that $\Delta = \gamma$ (verify).

Exercise 7.18 To simplify the analysis, we analyzed algorithm RFN using the worst-case estimate W_M^* on the expected running time of the primality test. Define

$$W_M^+ := \sum_{j=2}^{M} \frac{W_j}{j-1},$$

where W_n denotes the expected running time of a probabilistic implementation of *IsPrime* on input n. Show that the expected running time of algorithm RFN is $O(kW_M^+)$, assuming $\epsilon\ell(M) \leq 1/2$. □

Exercise 7.19 Analyze algorithm RFN assuming that the primality test is implemented by an "Atlantic City" algorithm with error probability at most ϵ. □

7.8 Notes

See Luby [45] for an exposition of the theory of pseudo-random bit generation.

The algorithm presented here for generating a random factored number is due to Kalai [37]. Kailai's algorithm is significantly simpler, though less efficient than, an earlier algorithm due to Bach [8], which uses an expected number of $O(k)$ primality tests.

Chapter 8

Abelian Groups

This chapter introduces the notion of an abelian group. This is an abstraction that models many different algebraic structures, and yet despite the level of generality, a number of very useful results can be easily obtained.

8.1 Definitions, Basic Properties, and Some Examples

A **binary operation** $\star$ on a set S is a function mapping pairs of elements of S into S; the value of the function on the pair (a, b) is denoted $a \star b$.

Definition 8.1 *An **abelian group** is a set G together with a binary operation $\star$ on G such that*

1. *for all $a, b \in G$, $a \star b = b \star a$ (commutivity property),*
2. *for all $a, b, c \in G$, $a \star (b \star c) = (a \star b) \star c$ (associativity property),*
3. *there exists $e \in G$ (called the **identity element**) such that for all $a \in G$, $a \star e = a$ (identity property),*
4. *for all $a \in G$ there exists $a' \in G$ such that $a \star a' = e$ (inverse property).*

Before looking at examples, let us state some very basic properties of abelian groups that follow directly from the definition.

Theorem 8.2 *Let G be an abelian group with operator $\star$. Then we have*

1. *the identity element is unique, i.e., there is only one element $e \in G$ such that $a \star e = a$ for all $a \in G$;*
2. *inverses are unique, i.e., for all $a \in G$, there is only one element $a' \in G$ such that $a \star a'$ is the identity.*

Proof. Suppose e, e' are both identities. Then we have

$$e = e \star e' = e' \star e = e',$$

where we have used part (3) of the definition with e' as the identity, then part (1), and then part (3) with e as the identity. So we see that there is only one identity.

Now let $a \in G$, and suppose that $a \star a' = e$ and $a \star a'' = e$. Then we have

$$a' = a' \star e = a' \star (a \star a'') = (a' \star a) \star a'' = (a \star a') \star a'' = e \star a'' = a'' \star e = a'',$$

where we have used part (3) of the definition, the identity $a \star a'' = e$, parts (2) and (1) of the definition, the identity $a \star a' = e$, and parts (1) and (3) of the definition. That proves a has only one inverse. □

Abelian groups are lurking everywhere, as the following examples illustrate.

Example 8.3 The set of integers $\mathbb{Z}$ under addition forms an abelian group, with 0 being the identity, and $-a$ being the inverse of $a \in \mathbb{Z}$. □

Example 8.4 For integer n, the set $n\mathbb{Z} := \{nz : z \in \mathbb{Z}\}$ under addition forms as abelian group, again, with 0 being the identity, and $n(-z)$ being the inverse of nz. □

Example 8.5 The set of non-negative integers under addition does not form an abelian group, since inverses do not exist for integers other than 0. □

Example 8.6 The set of integers under multiplication does not form an abelian group, since inverses do not exist for integers other than ± 1. □

Example 8.7 The set of integers $\{\pm 1\}$ under multiplication forms an abelian group, with 1 being the identity, and -1 its own inverse. □

Example 8.8 The set of rational numbers $\mathbb{Q} = \{a/b : a, b \in \mathbb{Z},\ b \neq 0\}$ under addition forms an abelian group, with 0 being the identity, and $(-a)/b$ being the inverse of a/b. □

Example 8.9 The set of non-zero rational numbers $\mathbb{Q}^*$ under multiplication forms an abelian group, with 1 being the identity, and b/a being the inverse of a/b. □

Example 8.10 The set $\mathbb{Z}_n$ under addition forms an abelian group, where $[0 \bmod n]$ is the identity, and where $[-a \bmod n]$ is the inverse of $[a \bmod n]$. □

Example 8.11 The set $\mathbb{Z}_n^*$ of residue classes $[a \bmod n]$ with $\gcd(a, n) = 1$ under multiplication forms an abelian group, where $[1 \bmod n]$ is the identity, and if $as + nt = 1$, then $[s \bmod n]$ is the inverse of $[a \bmod n]$. $\mathbb{Z}_n^*$ is called the **multiplicative group of units modulo** n. □

Example 8.12 Continuing the previous example, let us set $n = 15$, and enumerate the elements of $\mathbb{Z}_{15}^*$. They are

$$[1], [2], [4], [7], [8], [11], [13], [14].$$

An alternative enumeration is

$$[\pm 1], [\pm 2], [\pm 4], [\pm 7].$$

□

Example 8.13 As another special case, consider $\mathbb{Z}_5^*$. We can enumerate the elements of this groups as

$$[1], [2], [3], [4]$$

or alternatively as

$$[\pm 1], [\pm 2].$$

□

Example 8.14 For any positive integer n, the set of n-bit strings under the "exclusive or" operator forms an abelian group, where the "all zero" bit string is the identity, and every bit string is its own inverse. □

From the above examples, one can see that a group may be infinite or finite. If the group is finite, we define its **order** to be the number of elements in the underlying set G; otherwise, we say that the group has **infinite order**.

Example 8.15 The order of $\mathbb{Z}_n$ is n. □

Example 8.16 The order of $\mathbb{Z}_n^*$ is $\phi(n)$, where ϕ is Euler's function, defined in §2.4. □

Note that in specifying a group, one must specify both the underlying set G as well as the binary operation; however, in practice, the binary operation is often implicit from context, and by abuse of notation, one often refers to G itself as the group.

Usually, instead of using a special symbol like $\star$ for an abelian group operator, one instead uses the usual addition ("+") or multiplication ("$\cdot$") operators.

If an abelian group G is written additively, then the identity element is denoted by 0_G (or just 0 if G is clear from context), and the inverse of an element $a \in G$ is denoted by $-a$. For $a, b \in G$, $a - b$ denotes $a + (-b)$. If n is a positive integer, then $n \cdot a$ denotes $a + a + \cdots + a$, where there are n terms in the sum. Moreover, if $n = 0$, then $n \cdot a$ denotes 0, and if n is a negative integer then $n \cdot a$ denotes $-((-n) \cdot a)$.

If an abelian group G is written multiplicatively, then the identity element is denoted by 1_G (or just 1 if G is clear from context), and the inverse of an element $a \in G$ is denoted by a^{-1} or $1/a$. As usual, one may write ab in place of $a \cdot b$. For $a, b \in G$, a/b denotes $a \cdot b^{-1}$. If n is a positive integer, then a^n denotes $a \cdot a \cdot \cdots \cdot a$, where there are n terms in

the product. Moreover, if $n = 0$, then a^n denotes 1, and if n is a negative integer, then a^n denotes $(a^{-n})^{-1}$.

For any particular, concrete abelian group, the most natural choice of notation is clear; however, for a "generic" group, the choice is largely a matter of taste. By convention, whenever we consider a "generic" abelian group, we shall use *additive* notation for the group operation, unless otherwise specified.

We now record a few simple but useful properties of abelian groups.

Theorem 8.17 *Let G be an abelian group. Then for all $a, b, c \in G$ and $n, m \in \mathbb{Z}$, we have:*

1. *if $a + b = a + c$, then $b = c$;*
2. *the equation $a + x = b$ in x has a unique solution in G;*
3. *$-(a + b) = (-a) + (-b)$;*
4. *$-(-a) = a$;*
5. *$(-n)a = -(na) = n(-a)$;*
6. *$(n + m)a = na + ma$;*
7. *$n(ma) = (nm)a = m(na)$;*
8. *$n(a + b) = na + nb$.*

Proof. Exercise. □

If $G_1, \ldots, G_k$ are abelian groups, we can form the **direct product** $G_1 \times \cdots \times G_k$, which consists of all k-tuples $(a_1, \ldots, a_k)$ with $a_1 \in G_1, \ldots, a_k \in G_k$. We can view $G_1 \times \cdots \times G_k$ in a natural way as an abelian group if we define the group operation "component wise":

$$(a_1, \ldots, a_k) + (b_1, \ldots, b_k) := (a_1 + b_1, \ldots, a_k + b_k).$$

Of course, the groups $G_1, \ldots, G_k$ may be different, and the group operation applied in the ith component corresponds to the group operation associated with G_i. We leave it to the reader to verify that $G_1 \times \cdots \times G_k$ is in fact an abelian group.

In this text, we have chosen only to discuss the notion of an abelian group. There is a more general notion of a **group**, which may be defined simply by dropping the commutivity property in Definition 8.1, but we shall not need this notion in this text, and restricting to abelian groups helps to simplify the discussion significantly. Nevertheless, many of the notions and results we discuss here regarding abelian groups extend (sometimes with slight modification) to general groups.

Example 8.18 The set of 2×2 integer matrices with determinant ± 1 with respect to matrix multiplication forms a group, but not an abelian group. □

8.2 Subgroups

We next introduce the notion of a subgroup.

Definition 8.19 *Let G be an abelian group, and let H be a non-empty subset of G such that*

- *for all $a, b \in H$, $a + b \in H$, and*
- *for all $a \in H$, $-a \in H$.*

*Then H is called a **subgroup of** G.*

Theorem 8.20 *If G is an abelian group, and H is a subgroup, then the binary operation of G defines a binary operation on H, and with respect to this binary operation, H forms an abelian group whose identity is the same as that of G.*

Proof. Exercise. □

Clearly, for an abelian group G, the subsets G and $\{0_G\}$ are subgroups. These are not very interesting subgroups. An easy way to sometimes find other, more interesting, subgroups within an abelian group is by using the following two theorems:

Theorem 8.21 *Let G be an abelian group, and let m be an integer. Then $mG := \{ma : a \in G\}$ is a subgroup of G.*

Proof. For $ma, mb \in mG$, we have $ma+mb = m(a+b) \in mG$, and $-(ma) = m(-a) \in mG$. □

Theorem 8.22 *Let G be an abelian group, and let m be an integer. Then $G\{m\} := \{a \in G : ma = 0_G\}$ is a subgroup of G.*

Proof. If $ma = 0_G$ and $mb = 0_G$, then $m(a + b) = ma + mb = 0_G + 0_G = 0_G$ and $m(-a) = -(ma) = -0_G = 0_G$. □

Multiplicative notation: if the abelian group G in the above two theorems is written using multiplicative notation, then we write the subgroup of the first theorem as $G^m := \{a^m : a \in G\}$. The subgroup in the second theorem is denoted in the same way: $G\{m\} := \{a \in G : a^m = 1_G\}$.

Example 8.23 For every integer m, the set $m\mathbb{Z}$ is the subgroup of $\mathbb{Z}$ consisting of all integer multiples of m. Two such subgroups $m\mathbb{Z}$ and $m'\mathbb{Z}$ are equal if and only if $m = \pm m'$. The subgroup $\mathbb{Z}\{m\}$ is equal to $\mathbb{Z}$ if $m = 0$, and is equal to $\{0\}$ otherwise. □

Example 8.24 Let n be a positive integer, and let $m \in \mathbb{Z}$, and consider the subgroup $m\mathbb{Z}_n$ of $\mathbb{Z}_n$. Now, $[b \bmod n] \in m\mathbb{Z}_n$ if and only if there exists $x \in \mathbb{Z}$ such that $mx \equiv b \pmod{n}$. By Theorem 2.5, such an x exists if and only if $d \mid b$, where $d = \gcd(m, n)$. Thus, $m\mathbb{Z}_n$ consists precisely of the n/d distinct residue classes

$$[i \cdot d \bmod n] \quad (i = 0, \ldots, n/d - 1),$$

and in particular, $m\mathbb{Z}_n = d\mathbb{Z}_n$.

Now consider the subgroup $\mathbb{Z}_n\{m\}$. The residue class $[x \bmod n]$ is in $\mathbb{Z}_n\{m\}$ if and only if $mx \equiv 0 \pmod{n}$. By Theorem 2.5, this happens if and only if $x \equiv 0 \pmod{n/d}$, where $d = \gcd(m, n)$ as above. Thus, $\mathbb{Z}_n\{m\}$ consists precisely of the d residue classes

$$[i \cdot n/d \bmod n] \quad (i = 0, \ldots, d - 1),$$

and in particular, $\mathbb{Z}_n\{m\} = \mathbb{Z}_n\{d\} = (n/d)\mathbb{Z}_n$. □

Example 8.25 For $n = 15$, consider again the table in Example 2.6. For $m = 1, 2, 3, 4, 5, 6$, the elements appearing in the mth row of that table form the subgroup $m\mathbb{Z}_n$, and also the subgroup $\mathbb{Z}_n\{n/d\}$, where $d = \gcd(m, n)$. □

Because the abelian groups $\mathbb{Z}$ and $\mathbb{Z}_n$ are of such importance, it is a good idea to completely characterize all subgroups of these abelian groups. As the following two theorems show, the subgroups in the above examples are the *only* subgroups of these groups.

Theorem 8.26 *If G is a subgroup of $\mathbb{Z}$, then there exists a unique non-negative integer m such that $G = m\mathbb{Z}$.*

Proof. Actually, we have already proven this. One only needs to observe that a subset G is a subgroup if and only if it is an ideal (as defined in §1.2), and then apply Theorem 1.10. □

Theorem 8.27 *If G is a subgroup of $\mathbb{Z}_n$, then there exists a unique positive integer m dividing n such that $G = m\mathbb{Z}_n$.*

Proof. Let G be a subgroup of $\mathbb{Z}_n$. Define $G' := \{a \in \mathbb{Z} : [a] \in G\}$. It is easy to see that $G = \{[a] : a \in G'\}$.

First, we claim that G' is a subgroup of $\mathbb{Z}$. Suppose that $a, b \in G'$. This means that $[a] \in G$ and $[b] \in G$, which implies that $[a + b] = [a] + [b] \in G$, and hence $a + b \in G'$. Similarly, if $[a] \in G$, then $[-a] = -[a] \in G$, and hence $-a \in G'$.

By the previous theorem, it follows that G' is of the form $m\mathbb{Z}$ for some non-negative integer m. Moreover, note that $n \in G'$, since $[n] = [0]$ is the identity element of $\mathbb{Z}_n$, and hence belongs to G. Therefore, $m \mid n$.

So we have $G = \{[a] : a \in m\mathbb{Z}\} = m\mathbb{Z}_n$.

From the observations in Example 8.24, the uniqueness of m is clear. □

Of course, not all abelian groups have such a simple subgroup structure.

Example 8.28 Consider the group $G = \mathbb{Z}_2 \times \mathbb{Z}_2$. For any non-zero $\alpha \in G$, $\alpha + \alpha = 0_G$. From this, it is easy to see that the set $H = \{0_G, \alpha\}$ is a subgroup of G. However, for any integer m, $mG = G$ if m is odd, and $mG = \{0_G\}$ if m is even. Thus, the subgroup H is not of the form mG for any m. □

Example 8.29 Consider the group $\mathbb{Z}_n^*$ discussed in Example 8.11. The subgroup $(\mathbb{Z}_n^*)^2$ plays an important role in some situations. Integers a such that $[a] \in (\mathbb{Z}_n^*)^2$ are called **quadratic residues modulo** n. □

Example 8.30 Consider again the group $\mathbb{Z}_n^*$, for $n = 15$, discussed in Example 8.12. As discussed there, we have $\mathbb{Z}_{15}^* = \{[\pm 1], [\pm 2], [\pm 4], [\pm 7]\}$. Therefore, the elements of $(\mathbb{Z}_{15}^*)^2$ are

$$[1]^2 = [1],\ [2]^2 = [4],\ [4]^2 = [16] = [1],\ [7]^2 = [49] = [4];$$

thus, $(\mathbb{Z}_{15}^*)^2$ has order 2, consisting as it does of the two distinct elements [1] and [4].

Going further, one sees that $(\mathbb{Z}_{15}^*)^4 = \{[1]\}$. Thus, $\alpha^4 = [1]$ for all $\alpha \in \mathbb{Z}_{15}^*$.

By direct calculation, one can determine that $(\mathbb{Z}_{15}^*)^3 = \mathbb{Z}_{15}^*$; that is, cubing simply permutes $\mathbb{Z}_{15}^*$.

For any integer m, write $m = 4q + r$, where $0 \leq r < 4$. Then for any $\alpha \in \mathbb{Z}_{15}^*$, we have $\alpha^m = \alpha^{4q+r} = \alpha^{4q}\alpha^r = \alpha^r$. Thus, $(\mathbb{Z}_{15}^*)^m$ is either $\mathbb{Z}_{15}^*$, $(\mathbb{Z}_{15}^*)^2$, or $\{[1]\}$.

However, there are certainly other subgroups of $\mathbb{Z}_{15}^*$ — for example, the subgroup $\{[\pm 1]\}$. □

Example 8.31 Consider again the group $\mathbb{Z}_5^*$ from Example 8.13. As discussed there, $\mathbb{Z}_5^* = \{[\pm 1], [\pm 2]\}$. Therefore, the elements of $(\mathbb{Z}_5^*)^2$ are

$$[1]^2 = [1],\ [2]^2 = [4] = [-1];$$

thus, $(\mathbb{Z}_5^*)^2 = \{[\pm 1]\}$ and has order 2.

There are in fact no other subgroups of $\mathbb{Z}_5^*$ besides $\mathbb{Z}_5^*$, $\{[\pm 1]]\}$, and $\{[1]\}$. Indeed, if H is a subgroup containing [2], then we must have $H = \mathbb{Z}_5^*$: $[2] \in H$ implies $[2]^2 = [4] = [-1] \in H$, which implies $[-2] \in H$ as well. The same holds if H is a subgroup containing $[-2]$. □

If G is an abelian group, and H_1 and H_2 are subgroups, we define $H_1 + H_2 := \{h_1 + h_2 : h_1 \in H_1,\ h_2 \in H_2\}$. Note that $H_1 + H_2$ contains $H_1 \cup H_2$.

Multiplicative notation: if G is written multiplicatively, then we write $H_1 \cdot H_2 := \{h_1 h_2 : h_1 \in H_1,\ h_2 \in H_2\}$.

We close this section with a simple theorem that makes it easier to verify that something is a subgroup when it is is finite.

Theorem 8.32 *If G is an abelian group, and H is a non-empty, finite subset of G such that $a + b \in H$ for all $a, b \in H$, then H is a subgroup of G.*

Proof. We only need to show that $-a \in H$ for all $a \in H$. Let $a \in H$ be given. If $a = 0$, then clearly $-a = 0 \in H$, so assume that $a \neq 0$, and consider the set S of all elements of G of the form ma, for $m = 1, 2, \ldots$. Since H is closed under addition, it follows that $S \subset H$. Moreover, since H is finite, S must be finite, and hence there must exist positive integers m_1, m_2, such that $m_1 a = m_2 a$, but $m_1 \neq m_2$. We may assume that $m_1 > m_2$. We may further assume that $m_1 - m_2 > 1$, since otherwise $a = (m_1 - m_2)a = 0$, and we are assuming that $a \neq 0$. It follows that $-a = (m_1 - m_2 - 1)a \in S$. □

Exercise 8.33 Show that if H_1 and H_2 are subgroups of an abelian group G, then so is $H_1 + H_2$. Moreover, show that any subgroup H of G that contains $H_1 \cup H_2$ contains $H_1 + H_2$, and $H_1 \subset H_2$ if and only if $H_1 + H_2 = H_2$. □

Exercise 8.34 Show that if H_1 and H_2 are subgroups of an abelian group G, then so is $H_1 \cap H_2$. □

Exercise 8.35 Show that if H' is a subgroup of an abelian group G, then a set $H \subset H'$ is a subgroup of G if and only if H is a subgroup of H'. □

8.3 Cosets and Quotient Groups

We now generalize the notion of a congruence relation.

Let G be an abelian group, and let H be a subgroup. For $a, b \in G$, we write $a \equiv b \pmod{H}$ if $a - b \in H$.

It is easy to verify that the relation $\cdot \equiv \cdot \pmod{H}$ is an equivalence relation (see §A.5). Therefore, this relation partitions G into equivalence classes. It is easy to see that for any $a \in G$, the equivalence class containing a is precisely $a + H := \{a + h : h \in H\}$; indeed, $a \equiv b \pmod{H} \iff b - a = h$ for some $h \in H \iff b = a + h$ for some $h \in H \iff b \in a + H$. The equivalence class $a + H$ is called the **coset of H in G containing a**, and an element of such a coset is called a **representative** of the coset.

Multiplicative notation: if G is written multiplicatively, then $a \equiv b \pmod{H}$ means $a/b \in H$, and the coset of H in G containing a is $aH := \{ah : h \in H\}$.

Example 8.36 Let $G = \mathbb{Z}$ and $H = n\mathbb{Z}$ for some positive integer n. Then $a \equiv b \pmod{H}$ if and only if $a \equiv b \pmod{n}$. The coset $a + H$ is exactly the same thing as the residue class $[a \bmod n]$. □

Example 8.37 Let $G = \mathbb{Z}_4$ and let H be the subgroup $2\mathbb{Z}_4 = \{[0], [2]\}$. The coset of H containing $[1]$ is $\{[1], [3]\}$. These are all the cosets of H in G. □

Theorem 8.38 *Any two cosets of a subgroup H in an abelian group G have equal cardinality; i.e., there is a bijective map from one coset to the other.*

Proof. Let $a + H$ and $b + H$ be two cosets, and consider the map $f : G \to G$ that sends $x \in G$ to $x - a + b \in G$. The reader may verify that f is injective and carries $a + H$ onto $b + H$. □

An incredibly useful consequence of the above theorem is:

Theorem 8.39 *If G is a finite abelian group, and H is a subgroup of G, then the order of H divides the order of G.*

Proof. This is an immediate consequence of the previous theorem, and the fact that the cosets of H in G partition G. □

Analogous to Theorem 2.1, we have:

Theorem 8.40 *Let G be an abelian group and H a subgroup. For $a, a', b, b' \in G$, if $a \equiv a' \pmod{H}$ and $b \equiv b' \pmod{H}$, then $a + b \equiv a' + b' \pmod{H}$.*

Proof. Now, $a \equiv a' \pmod{H}$ and $b \equiv b' \pmod{H}$ means that $a' = a + h_1$ and $b' = b + h_2$ for $h_1, h_2 \in H$. Therefore, $a' + b' = (a + h_1) + (b + h_2) = (a + b) + (h_1 + h_2)$, and since $h_1 + h_2 \in H$, this means that $a + b \equiv a' + b' \pmod{H}$. □

Let G be an abelian group and H a subgroup. Theorem 8.40 allows us to define a group operation on the collection of cosets of H in G in the following natural way: for $a, b \in G$, define

$$(a + H) + (b + H) := (a + b) + H.$$

The fact that this definition is unambiguous follows immediately from Theorem 8.40. Also, one can easily verify that this operation defines an abelian group. The resulting group is called the **quotient group of G modulo H**, and is denoted G/H.

The order of the group G/H is sometimes denoted $[G : H]$ and is called the **index of H in G**. If G is of finite order, then by Theorem 8.38, $[G : H] = |G|/|H|$. Moreover, if H and H' are subgroups of G with $H \subset H'$, then H is a subgroup of H', and we have then we have

$$[G : H'] = \frac{|G|}{|H'|} = \frac{|G|/|H|}{|H'|/|H|} = \frac{[G : H]}{[H' : H]},$$

and we conclude that

$$[G : H] = [G : H'][H' : H].$$

Multiplicative notation: if G is written multiplicatively, then the definition of the group operation of G/H is expressed

$$(aH) \cdot (bH) := (ab)H.$$

Example 8.41 For the additive group of integers $\mathbb{Z}$ and the subgroup $n\mathbb{Z}$ for $n > 0$, the quotient group $\mathbb{Z}/n\mathbb{Z}$ is precisely the same as the additive group $\mathbb{Z}_n$ that we have already defined. For $n = 0$, $\mathbb{Z}/n\mathbb{Z}$ is essentially just a "renaming" of $\mathbb{Z}$. □

Example 8.42 Let $G := \mathbb{Z}_6$ and $H = 3G$ be the subgroup of G consisting of the two elements $\{[0], [3]\}$. The cosets of H in G are $\alpha := H = \{[0], [3]\}$, $\beta := [1] + H = \{[1], [4]\}$, and $\gamma := [2] + H = \{[2], [5]\}$. If we write out an addition table for G, grouping together elements in cosets of H in G, then we also get an addition table for the quotient group G/H:

	0	3	1	4	2	5
0	0	3	1	4	2	5
3	3	0	4	1	5	2
1	1	4	2	5	3	0
4	4	1	5	2	0	3
2	2	5	3	0	4	1
5	5	2	0	3	1	4

This table illustrates quite graphically the point of Theorem 8.40, that if we take any element of one coset and add it to any element of another, we always end up in the same coset.

We can also write down just the addition table for G/H:

	α	β	γ
α	α	β	γ
β	β	γ	α
γ	γ	α	β

□

Example 8.43 Let us return to Example 8.30. The multiplicative group $\mathbb{Z}_{15}^*$, as we saw, is of order 8. The subgroup $(\mathbb{Z}_{15}^*)^2$ has order 2. Therefore, the quotient group has order 4. Indeed, the cosets are $\alpha_{00} = \{[1], [4]\}$, $\alpha_{01} = \{[-1], [-4]\}$, $\alpha_{10} = \{[2], [-7]\}$, and $\alpha_{11} = \{[7], [-2]\}$. In the group $\mathbb{Z}_{15}^*/(\mathbb{Z}_{15}^*)^2$, α_{00} is the identity; moreover, we have

$$\alpha_{01}^2 = \alpha_{10}^2 = \alpha_{11}^2 = \alpha_{00}$$

and

$$\alpha_{01}\alpha_{10} = \alpha_{11},\ \alpha_{10}\alpha_{11} = \alpha_{01},\ \alpha_{10}\alpha_{11} = \alpha_{01}.$$

This completely describes the behavior of the group operation of the quotient group. Note that this group is essentially just a "renaming" of the group $\mathbb{Z}_2 \times \mathbb{Z}_2$. □

Example 8.44 As we saw in Example 8.31, $(\mathbb{Z}_5^*)^2 = \{[\pm 1]\}$. Therefore, the quotient group $\mathbb{Z}_5^*/(\mathbb{Z}_5^*)^2$ has order 2. The cosets of $(\mathbb{Z}_5^*)^2$ in $\mathbb{Z}_5^*$ are $\alpha_0 = \{[\pm 1]\}$ and $\alpha_1 = \{[\pm 2\}$. In the group $\mathbb{Z}_5^*/(\mathbb{Z}_5^*)^2$, α_0 is the identity, and α_1 is its own inverse, and we see that this group is essentially just a "renaming" of $\mathbb{Z}_2$. □

8.4 Group Homomorphisms and Isomorphisms

Definition 8.45 *A **group homomorphism** is a function from an abelian group G to an abelian group G' such that $\rho(a+b) = \rho(a) + \rho(b)$ for all $a, b \in G$.*

*The set $\rho^{-1}(0_{G'})$ is called the **kernel** of ρ, and is denoted $\ker(\rho)$. The set $\rho(G)$ is called the **image** of ρ, and may be denoted $\mathrm{im}(\rho)$.*

*If ρ is bijective, then ρ is called a **group isomorphism** of G with G', and moreover, if $G = G'$, then ρ is called a **group automorphism** on G.*

It is easy to see that if $\rho : G \to G'$ and $\rho' : G' \to G''$ are group homomorphisms, then so is their composition $\rho' \circ \rho : G \to G''$; indeed, for $a, b \in G$, we have $\rho'(\rho(a+b)) = \rho'(\rho(a) + \rho(b)) = \rho'(\rho(a)) + \rho'(\rho(b))$.

It is also easy to see that if ρ is an isomorphism of G with G', then the inverse function ρ^{-1} is an isomorphism of G' with G, since

$$\rho(\rho^{-1}(a') + \rho^{-1}(b')) = \rho(\rho^{-1}(a')) + \rho(\rho^{-1}(b')) = a' + b',$$

and hence $\rho^{-1}(a') + \rho^{-1}(b') = \rho^{-1}(a' + b')$. If such a group isomorphism exists, we say that G and G' are **isomorphic**, and write $G \cong G'$. we stress that an isomorphism of G with G' is essentially just a "renaming" of the group elements — all structural properties of the group are preserved.

Example 8.46 For any abelian group G and any integer m, the map that sends $a \in G$ to $ma \in G$ is clearly a group homomorphism from G into G. The image of this homomorphism is mG and the kernel is $G\{m\}$. We call this map the **m-multiplication map on G**. If G is written multiplicatively, we call this the **m-power map on G**, and its image is G^m. □

Example 8.47 For any abelian groups G, H, the function ρ that sends $(g, h) \in G \times H$ to $g \in G$ is a group homomorphism from $G \times H$ into G. The image of ρ is G, and the kernel of ρ is $\{0_G\} \times H$. □

Example 8.48 Consider the m-multiplication map on $\mathbb{Z}_n$. The image of this map is $m\mathbb{Z}_n$, which as we saw above in Example 8.24 is a subgroup of $\mathbb{Z}_n$ of order n/d, where $d = \gcd(n, m)$. Thus, this map is bijective if and only if $d = 1$, in which case it is a group automorphism on $\mathbb{Z}_n$. □

Example 8.49 For $n > 0$, we have defined $\mathbb{Z}_n$ so that it is literally the same as $\mathbb{Z}/n\mathbb{Z}$. A more "low tech" approach is to define the group C_n which consists of the set of integers $\{0, 1, \ldots, n-1\}$, with the group operation that sends i, j to $(i + j) \bmod n$. It is easy to verify that $\mathbb{Z}_n$ is isomorphic to C_n. For $n = 0$, as we said in Example 8.41, the group $\mathbb{Z}/n\mathbb{Z}$ is isomorphic to $\mathbb{Z}$. □

Example 8.50 As was shown in Example 8.43, the quotient group $\mathbb{Z}_{15}^*/(\mathbb{Z}_{15}^*)^2$ is isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2$, and as was shown in Example 8.44, the quotient group $\mathbb{Z}_5^*/(\mathbb{Z}_5^*)^2$ is isomorphic to $\mathbb{Z}_2$. □

The following theorem summarizes some of the most important properties of group homomorphisms.

Theorem 8.51 *Let ρ be a group homomorphism from G to G'.*

1. $\rho(0_G) = 0_{G'}$.
2. $\rho(-a) = -\rho(a)$ *for all* $a \in G$.
3. $\rho(na) = n\rho(a)$ *for all* $n \in \mathbb{Z}$ *and* $a \in G$.
4. *For any subgroup H of G, $\rho(H)$ is a subgroup of G'.*
5. $\ker(\rho)$ *is a subgroup of* G.
6. *For all* $a, b \in G$, $\rho(a) = \rho(b)$ *if and only if* $a \equiv b \pmod{\ker(\rho)}$.
7. ρ *is injective if and only if* $\ker(\rho) = \{0_G\}$.
8. *For any subgroup H' of G', $\rho^{-1}(H')$ is a subgroup of G containing* $\ker(\rho)$.

Proof.

1. We have
$$0_{G'} + \rho(0_G) = \rho(0_G) = \rho(0_G + 0_G) = \rho(0_G) + \rho(0_G).$$
Now cancel $\rho(0_G)$ from both sides (using part (1) of Theorem 8.17).

2. We have
$$0_{G'} = \rho(0_G) = \rho(a + (-a)) = \rho(a) + \rho(-a),$$
and hence $\rho(-a)$ is the inverse of $\rho(a)$.

3. For non-negative n, this follows by induction from the definitions, and for negative n, this follows from the positive case and part (5) of Theorem 8.17.

4. For any $a, b \in H$, we have $a + b \in H$ and $-a \in H$; hence, $\rho(H)$ contains $\rho(a + b) = \rho(a) + \rho(b)$ and and $\rho(-a) = -\rho(a)$.

5. If $\rho(a) = 0_{G'}$ and $\rho(b) = 0_{G'}$, then $\rho(a + b) = \rho(a) + \rho(b) = 0_{G'} + 0_{G'} = 0_{G'}$, and $\rho(-a) = -\rho(a) = -0_{G'} = 0_{G'}$.

6. $\rho(a) = \rho(b)$ iff $\rho(a) - \rho(b) = 0_{G'}$ iff $\rho(a - b) = 0_{G'}$ iff $a - b \in \ker(\rho)$ iff $a \equiv b \pmod{\ker(\rho)}$.

7. If ρ is injective, then in particular, $\rho^{-1}(0_{G'})$ cannot contain any other element besides 0_G. If ρ is not injective, then there exist two distinct elements $a, b \in G$ with $\rho(a) = \rho(b)$, and by part part (6), $\ker(\rho)$ contains the element $a - b$, which in non-zero.

8. This is very similar to part (5). If $\rho(a) \in H'$ and $\rho(b) \in H'$, then $\rho(a+b) = \rho(a)+\rho(b) \in H'$, and $\rho(-a) = -\rho(a) \in H'$. Moreover, since H' contains $0_{G'}$, we must have $\rho^{-1}(H') \supset \rho^{-1}(0_{G'}) = \ker(\rho)$.

□

Part (7) of the above theorem is particular useful: to check that a group homomorphism is injective, it suffices to determine if $\ker(\rho) = \{0_G\}$.

Theorem 8.52 *If H is a subgroup of an abelian group G, then the map $\rho : G \to G/H$ given by $\rho(a) = a + H$ is a surjective group homomorphism whose kernel is H. This is sometimes called the "natural" map from G to G/H.*

Proof. This really just follows from the definition of the quotient group. To verify that ρ is a group homomorphism, note that

$$\rho(a+b) = (a+b)+H = (a+H)+(b+H) = \rho(a)+\rho(b).$$

Surjectivity follows from the fact that every coset is of the form $a+H$ for some $a \in G$. □

Theorem 8.53 *Let ρ be a group homomorphism from G into G'. Then the map $\bar{\rho} : G/\ker(\rho) \to \mathrm{im}(\rho)$ that sends the coset $a + \ker(\rho)$ for $a \in G$ to $\rho(a)$ is unambiguously defined and is a group isomorphism of $G/\ker(\rho)$ with $\mathrm{im}(\rho)$.*

Proof. To see that the definition $\bar{\rho}$ is unambiguous, note that if $a \equiv a' \bmod \ker(\rho)$, then by part (6) of Theorem 8.51, $\rho(a) = \rho(a')$. To see that $\bar{\rho}$ is a group homomorphism, note that

$$\begin{aligned} \bar{\rho}((a+\ker(\rho))+(b+\ker(\rho)) &= \bar{\rho}((a+b)+\ker(\rho)) = \rho(a+b) = \rho(a)+\rho(b) \\ &= \bar{\rho}(a+\ker(\rho))+\bar{\rho}(b+\ker(\rho)). \end{aligned}$$

It is clear that $\bar{\rho}$ maps onto $\mathrm{im}(\rho)$, since any element of $\mathrm{im}(\rho)$ is of the form $\rho(a)$ for some $a \in G$, and the map $\bar{\rho}$ sends $a+\ker(\rho)$ to $\rho(a)$. Finally, to see that $\bar{\rho}$ is injective, note that $\bar{\rho}(a+\ker(\rho)) = 0_{G'}$ implies that $\rho(a) = 0_{G'}$, which implies that $a \in \ker(\rho)$, which implies that the coset $a+\ker(\rho)$ is equal to $\ker(\rho)$, which is the zero element of $G/\ker(\rho)$. Injectivity follows from part (7) of Theorem 8.51. □

The following theorem is an easy generalization of the previous one.

Theorem 8.54 *Let ρ be a group homomorphism from G into G'. Then for any subgroup H contained in $\ker(\rho)$, the map $\bar{\rho} : G/H \to \mathrm{im}(\rho)$ that sends the coset $a+H$ for $a \in G$ to $\rho(a)$ is unambiguously defined and is a group homomorphism from G/H onto $\mathrm{im}(\rho)$ with kernel $\ker(\rho)/H$.*

Proof. Exercise — just mimic the proof of the previous theorem. □

Theorem 8.55 *Let G be an abelian group with subgroups H_1, H_2 such that $H_1 \cap H_2 = \{0_G\}$. Then the map that sends $(h_1, h_2) \in H_1 \times H_2$ to $h_1 + h_2 \in H_1 + H_2$ is a group isomorphism of $H_1 \times H_2$ with $H_1 + H_2$.*

Proof. Let ρ be the map defined above. To see that ρ is a group homomorphism, note that for $h_1, h_1' \in H_1$ and $h_2, h_2' \in H_2$, we have

$$\rho(h_1 + h_1', h_2 + h_2') = (h_1 + h_1') + (h_2 + h_2') = (h_1 + h_2) + (h_1' + h_2') = \rho(h_1, h_2) + \rho(h_1', \rho_2').$$

That ρ is surjective is clear from the definitions. To see that ρ is injective, it suffices to show that $\ker(\rho)$ is trivial, i.e., that for all $h_1 \in H_1$ and $h_2 \in H_2$, $h_1 + h_2 = 0$ implies $h_1 = 0$ and $h_2 = 0$. But $h_1 + h_2 = 0$ implies $h_1 = -h_2 \in H_2$, and hence $h_1 \in H_1 \cap H_2 = \{0\}$, and so $h_1 = 0$. Similarly, one shows that $h_2 = 0$, and that finishes the proof. □

The last theorem says that when $H_1 \cap H_2 = \{0\}$, every element of $H_1 + H_2$ can be expressed *uniquely* as $h_1 + h_2$, with $h_1 \in H_1$ and $h_2 \in H_2$. In this situation, one calls $H_1 + H_2$ the **internal direct sum** of H_1 and H_2 (or the **internal direct product** if the group is written multiplicatively). More generally, if $H_1, \ldots, H_n$ are subgroups of G such that every element of $H_1 + \cdots + H_n$ can be expressed uniquely as $h_1 + \cdots + h_n$ for $h_1 \in H_1, \ldots, h_n \in H_n$, then $H_1 + \cdots + H_n$ is called the **internal direct sum** of $H_1, \ldots, H_n$, and is isomorphic to the direct product $H_1 \times \cdots \times H_n$.

Example 8.56 For $n \geq 1$, the natural map ρ from $\mathbb{Z}$ to $\mathbb{Z}_n$ sends $a \in \mathbb{Z}$ to the residue class $[a \bmod n]$. This map is a surjective group homomorphism with kernel $n\mathbb{Z}$. □

Example 8.57 We may restate Theorem 2.7 (Chinese Remainder Theorem) in more algebraic terms. Let $n_1, \ldots, n_k$ be integers, all greater than 1, such that $\gcd(n_i, n_j) = 1$ for all $1 \leq i < j \leq k$. Consider the group homomorphism from the group $\mathbb{Z}$ to the group $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$ that sends $x \in \mathbb{Z}$ to $([x \bmod n_1], \ldots, [x \bmod n_k])$. In our new language, Theorem 2.7 says that this group homomorphism is surjective and the kernel is $n\mathbb{Z}$, where $n = \prod_{i=1}^{k} n_i$. Therefore, by Theorem 8.53, the map that sends $[x \bmod n] \in \mathbb{Z}_n$ to $([x \bmod n_1], \ldots, [x \bmod n_k])$ is a group isomorphism of the group $\mathbb{Z}_n$ with the group $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$. □

Example 8.58 Let n_1, n_2 be positive integers with $n_1 > 1$ and $n_1 \mid n_2$. Then the map $\bar{\rho} : \mathbb{Z}_{n_2} \to \mathbb{Z}_{n_1}$ that sends $[a \bmod n_2]$ to $[a \bmod n_1]$ is a surjective group homomorphism, and $[a \bmod n_2] \in \ker(\rho)$ if and only if $n_1 \mid a$, i.e., $\ker(\rho) = n_1\mathbb{Z}_{n_2}$. The map $\bar{\rho}$ can also be viewed as the map obtained from Theorem 8.54 applied to the natural map ρ from $\mathbb{Z}$ to $\mathbb{Z}_{n_1}$ and the subgroup $n_2\mathbb{Z}$ of $\mathbb{Z}$, which is contained in $\ker(\rho) = n_1\mathbb{Z}$. □

Exercise 8.59 Let ρ be a group homomorphism from G into G'. Show that for any subgroup H of G, we have $\rho^{-1}(\rho(H)) = H + \ker(\rho)$. □

Exercise 8.60 Let ρ be a group homomorphism from G into G'. Show that the subgroups of G containing $\ker(\rho)$ are in one-to-one correspondence with the subgroups of $\mathrm{im}(\rho)$, where the the subgroup H in G containing $\ker(\rho)$ corresponds to the subgroup $\rho(H)$ in $\mathrm{im}(\rho)$. □

Exercise 8.61 Show that if $H \subset H'$ are subgroups of an abelian group G, then we have a group isomorphism

$$G/H' \cong \frac{G/H}{H'/H}.$$

In particular, show that if $[G : H]$ is finite, then $[G : H] = [G : H'] \cdot [H' : H]$. □

Exercise 8.62 Sow that if $G = G_1 \times G_2$ for abelian groups G_1 and G_2, and H_1 is a subgroup of G_1 and H_2 is a subgroup of G_2, then $H := H_1 \times H_2$ is a subgroup of G, and $G/H \cong G_1/H_1 \times G_2/H_2$. □

Exercise 8.63 Let ρ_1 and ρ_2 be group homomorphisms from G into G'. Show that the map $\rho : G \to G'$ that sends $a \in G$ to $\rho_1(a) + \rho_2(a) \in G'$ is also a group homomorphism. □

Exercise 8.64 Let $\rho_i : G \to G_i$, for $i = 1, \ldots, n$, be group homomorphisms. Show that the map $\rho : G \to G_1 \times \cdots \times G_n$ that sends $a \in G$ to $(\rho_1(a), \ldots, \rho_n(a))$ is also a group homomorphism. □

Exercise 8.65 This exercise develops some simple — bet extremely useful — connections between group theory and probability theory. Let $\rho : G \to G'$ be a group homomorphism, where G is a finite abelian group.

(a) Show that if g is a random variable with the uniform distribution on G, then $\rho(g)$ is a random variable with the uniform distribution on $\mathrm{im}(\rho)$.

(b) Show that if g is a random variable with the uniform distribution on G, and g' is a fixed element in $\mathrm{im}(\rho)$, then the conditional distribution of g, given by the event $\rho(g) = g'$, is the uniform distribution on $\rho^{-1}(g')$.

(c) Show that if g'_1 is a fixed element of G', g_1 is uniformly distributed over $\rho^{-1}(g'_1)$, g'_2 is a fixed element of G', and g_2 is a fixed element of $\rho^{-1}(g'_2)$, then $g_1 + g_2$ is uniformly distributed over $\rho^{-1}(g'_1 + g'_2)$.

(d) Show that if g'_1 is a fixed element of G', g_1 is uniformly distributed over $\rho^{-1}(g'_1)$, g'_2 is a fixed element of G', g_2 is uniformly distributed over $\rho^{-1}(g'_2)$, and g_1 and g_2 are independent, then $g_1 + g_2$ is uniformly distributed over $\rho^{-1}(g'_1 + g'_2)$.

□

8.5 Cyclic Groups

Let G be an abelian group. For $a \in G$, define $\langle a \rangle := \{za : z \in \mathbb{Z}\}$. It is clear that $\langle a \rangle$ is a subgroup of G, and moreover, that any subgroup H of G that contains a must also contain $\langle a \rangle$. The subgroup $\langle a \rangle$ is called **the subgroup generated by** a. Also, one defines the **order** of a to be the order of the subgroup $\langle a \rangle$, which is denoted $\mathrm{ord}(a)$.

More generally, for $a_1, \ldots, a_k \in G$, we define $\langle a_1, \ldots, a_k \rangle := \{z_1 a_1 + \cdots + z_k a_k : z_1, \ldots, z_k \in \mathbb{Z}\}$. One also verifies that $\langle a_1, \ldots, a_k \rangle$ is a subgroup of G, and that any subgroup H of G that contains $a_1, \ldots, a_k$ must contain $\langle a_1, \ldots, a_k \rangle$. The subgroup $\langle a_1, \ldots, a_k \rangle$ is called the **subgroup generated by** $a_1, \ldots, a_k$.

An abelian group G is called a **cyclic group** if $G = \langle a \rangle$ for some $a \in G$, in which case, a is called a **generator for** G.

Multiplicative notation: if G is written multiplicatively, then $\langle a \rangle := \{a^z : z \in \mathbb{Z}\}$, and $\langle a_1, \ldots, a_k \rangle := \{a_1^{z_1} \cdots a_k^{z_k} : z_1, \ldots, z_k \in \mathbb{Z}\}$.

Example 8.66 $\mathbb{Z}$ is a cyclic group generated by 1. The only other generator is -1. More generally, $\langle m \rangle = m\mathbb{Z}$. □

Example 8.67 $\mathbb{Z}_n$ is a cyclic group generated by $[1 \bmod n]$. More generally, $\langle [m \bmod n] \rangle = m\mathbb{Z}_n$, and so as we saw in Example 8.24, the order of $m\mathbb{Z}_n$ is n/d, where $d = \gcd(m, n)$. Therefore, the number of generators of $\mathbb{Z}_n$ is $\phi(n)$. □

Example 8.68 Consider the group $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$. For $m \in \mathbb{Z}$, then the element $m([1 \bmod n_1], [1 \bmod n_2]) = ([0 \bmod n_1], [0 \bmod n_2])$ if and only if $n_1 \mid m$ and $n_2 \mid m$. This implies that $([1 \bmod n_1], [1 \bmod n_2])$ has order $\operatorname{lcm}(n_1, n_2)$. In particular, if $\gcd(n_1, n_2) = 1$, then $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ is cyclic of order $n_1 n_2$. Moreover, if $\gcd(n_1, n_2) = d > 1$, then all elements of $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ have order dividing $n_1 n_2 / d$, and so $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ cannot be cyclic. □

Example 8.69 As we saw in Example 8.30, all elements of $\mathbb{Z}_{15}^*$ have order dividing 4, and since $\mathbb{Z}_{15}^*$ has order 8, we conclude that $\mathbb{Z}_{15}^*$ is not cyclic. □

Example 8.70 The group $\mathbb{Z}_5^*$ is cyclic, with $[2]$ being a generator:

$$[2]^2 = [4] = [-1], \quad [2]^3 = [-2], \quad [2]^4 = [1].$$

□

We can very quickly characterize all cyclic groups, up to isomorphism. Suppose that G is a cyclic group with generator a. Consider the map $\rho : \mathbb{Z} \to G$ that sends $z \in \mathbb{Z}$ to $za \in G$. This map is clearly a surjective group homomorphism. Now, $\ker(\rho)$ is a subgroup of $\mathbb{Z}$, and by Theorem 8.26, it must be of the form $n\mathbb{Z}$ for some non-negative integer n. Also, by Theorem 8.53, we have $\mathbb{Z}/n\mathbb{Z} \cong G$.

Case 1: $n = 0$. In this case, $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}$, and so we see $G \cong \mathbb{Z}$. Moreover, by Theorem 8.51, the only integer z such that $za = 0_G$ is the integer 0, and more generally, $z_1 a = z_2 a$ if and only if $z_1 = z_2$.

Case 2: $n > 0$. In this case, $\mathbb{Z}/n\mathbb{Z} = \mathbb{Z}_n$, and so we see that $G \cong \mathbb{Z}_n$. Moreover, by Theorem 8.51, $za = 0_G$ if and only if $n \mid z$, and more generally, $z_1 a = z_2 a$ if and only if $z_1 \equiv z_2 \pmod{n}$. The order of G is evidently n, and G consists of the distinct elements

$$0 \cdot a, 1 \cdot a, \ldots, (n-1) \cdot a.$$

From this characterization, we immediately have:

Theorem 8.71 *Let G be an abelian group and let $a \in G$. If there exists a positive integer m such that $ma = 0_G$, then the least such integer is the order of a. Moreover, if G of finite order n, then* $\mathrm{ord}(a) \mid n$, *and in particular $na = 0_G$.*

Proof. The first statement follows from the above characterization. For the second statement, since $\langle a \rangle$ is a subgroup of G, by Theorem 8.39, its order must divide that of G. Of course, if $ma = 0_G$, then for any multiple m' of m (in particular, $m' = n$), we also have $m'a = 0_G$. □

Based on this theorem, we can trivially derive a classical result:

Theorem 8.72 (Fermat's Little Theorem) *For any prime p, and any integer $x \not\equiv 0 \pmod{p}$, we have $x^{p-1} \equiv 1 \pmod{p}$. Moreover, for any integer x, we have $x^p \equiv x \pmod{p}$.*

Proof. The first statement follows from Theorem 8.71, and the fact that $\mathbb{Z}_p^*$ is an abelian group of order $p - 1$. The second statement is clearly true if $x \equiv 0 \pmod{p}$, and if $x \not\equiv 0 \pmod{p}$, we simply multiply both sides of the congruence $x^{p-1} \equiv 1 \pmod{p}$ by x. □

More generally, we have

Theorem 8.73 *For any positive integer n, and any integer x relatively prime to n, we have $x^{\phi(n)} \equiv 1 \pmod{n}$.*

Proof. This follows from Theorem 8.71 and the fact that $\mathbb{Z}_p^*$ is an abelian group of order $p - 1$. □

The following two theorems completely characterize the subgroup structure of cyclic groups:

Theorem 8.74 *Let G be a cyclic group of infinite order.*

1. *G is isomorphic to $\mathbb{Z}$.*
2. *The subgroups of G are in one-to-one correspondence with the non-negative integers m, where each such integer corresponds to the cyclic group mG.*
3. *For any two non-negative integers m, m', $mG \subset m'G$ if and only if $m' \mid m$.*

Proof. That $G \cong \mathbb{Z}$ was established in the above characterization of cyclic groups, and so it suffices to prove the other statements of the theorem for $G = \mathbb{Z}$. The second statement was already established in Theorem 8.26. For the third statement, if $m\mathbb{Z} \subset m'\mathbb{Z}$, the in particular, $m \in m'\mathbb{Z}$, which means that $m' \mid m$; conversely, if $m' \mid m$, so that $m = m'd$, then for any $mz \in m\mathbb{Z}$, we have $mz = m'(dz) \in m'\mathbb{Z}$. □

Theorem 8.75 *Let G be a cyclic group of finite order n.*

1. *G is isomorphic to $\mathbb{Z}_n$.*
2. *The subgroups of G are in one-to-one correspondence with the positive divisors of n, where each such divisor d corresponds to the subgroup $G\{d\}$, which is a cyclic subgroup of order d and contains precisely those elements of G whose order divides d.*
3. *For each $d \mid n$, $G\{d\} = (n/d)G$; that is, $G\{d\}$ is the image of the (n/d)-multiplication map.*
4. *For any two divisors d, d' of n, $G\{d\} \supset G\{d'\}$ if and only if $d' \mid d$.*
5. *For any $d \mid n$, the number of elements of order d in G is precisely $\phi(d)$.*

Proof. That $G \cong \mathbb{Z}_n$ was established in the above characterization of cyclic groups, and so it suffices to prove the other statements of the theorem for $G = \mathbb{Z}_n$. By Theorem 8.27, any subgroup of $\mathbb{Z}_n$ is of the form $d\mathbb{Z}_n$ for a uniquely determined divisor d of n, and as we saw in Example 8.24, $d\mathbb{Z}_n = \mathbb{Z}_n\{d\}$, where $d = n/d$, and the order of this group is d. That proves statements (2) and (3) of the theorem.

For the third statement, if $\mathbb{Z}_n\{d\} \supset \mathbb{Z}_n\{d'\}$, then $\mathbb{Z}_n\{d'\}$ is a subgroup of $\mathbb{Z}_n\{d\}$, and so the order d' of $\mathbb{Z}_n\{d'\}$ must divide the order d of $\mathbb{Z}_n\{d\}$. Conversely, suppose $d' \mid d$. From statement (2), we know that $\mathbb{Z}_n\{d\}$ contains all those elements in $\mathbb{Z}_n$ whose order divides d, and so in particular, all those elements whose order divides d', and so contains $\mathbb{Z}_n\{d'\}$.

For the fifth statement, the elements of order d in $\mathbb{Z}_n$ are all contained in $\mathbb{Z}_n\{d\}$, and so the number of such elements is equal to the number of generators of $\mathbb{Z}_n\{d\}$. The group $\mathbb{Z}_n\{d\}$ is cyclic of order d, and so is isomorphic to $\mathbb{Z}_d$, and as we saw in Example 8.67, this group has $\phi(d)$ generators. □

We continue to develop the theory of cyclic groups in the following sequence of theorems.

Theorem 8.76 *If G is a cyclic group, and $\rho : G \to G'$ is a group homomorphism from G into G', then* $\mathrm{im}(\rho)$ *is cyclic.*

Proof. If a is a generator for G, then it is clear that $\rho(a)$ generates $\mathrm{im}(\rho)$. □

Theorem 8.77 *If G is a finite abelian group of order n, and m is an integer relatively prime to n, then $mG = G$.*

Proof. Consider the m-multiplication map on G.

We claim that the kernel of this map is $\{0_G\}$. Indeed, $ma = 0_G$, implies $\mathrm{ord}(a)$ divides m, and since $\mathrm{ord}(a)$ also divides n and $\gcd(m, n) = 1$, we must have $\mathrm{ord}(a) = 1$, i.e., $a = 0_G$. That proves the claim.

Thus, the m-multiplication map is injective, and because G is finite, it must be surjective as well. □

Theorem 8.78 *If G is an abelian group of prime order, then G is cyclic.*

Proof. Let $|G| = p$. Let $a \in G$ with $a \neq 0_G$. Since $\mathrm{ord}(a) \mid p$, we have $\mathrm{ord}(a) = 1$ or $\mathrm{ord}(a) = p$. Since $a \neq 0_G$, we must have $\mathrm{ord}(a) \neq 1$, and so $\mathrm{ord}(a) = p$, which implies a generates G. □

Theorem 8.79 *Suppose that a is an element of an abelian group, and for some prime p and integer $e \geq 1$, we have $p^e a = 0_G$ and $p^{e-1}a \neq 0_G$. Then a has order p^e.*

Proof. If m is the order of a, then since $p^e a = 0_G$, we have $m \mid p^e$. So $m = p^f$ for some $0 \leq f \leq e$. If $f < e$, then $p^{e-1}a = 0_G$, contradicting the assumption that $p^{e-1}a \neq 0_G$. □

Theorem 8.80 *Suppose G is an abelian group with $a_1, a_2 \in G$ such that a_1 is of finite order n_1, a_2 is of finite order n_2, and $\gcd(n_1, n_2) = 1$. Then the order of $a_1 + a_2$ is $n_1 n_2$.*

Proof. Consider the subgroup H_1 of G generated by a_1, and the subgroup H_2 of G generated by a_2. We claim that $H_1 \times H_2$ is generated by (a_1, a_2). The reason is essentially the same as that in Example 8.68: if m is the order of (a_1, a_2), then we must have $n_1 \mid m$ and $n_2 \mid m$, but since n_1 and n_2 are relatively prime, we must have $n_1 n_2 \mid m$. Moreover, $H_1 \cap H_2$ is a subgroup of both H_1 and of H_2, and hence the order of $H_1 \cap H_2$ must divide both n_1 and n_2; again, since n_1 and n_2 are relatively prime, we must have that $H_1 \cap H_2 = \{0_G\}$. By Theorem 8.55, the map that sends $(h_1, h_2) \in H_1 \times H_2$ to $h_1 + h_2 \in H_1 + H_2$ is an isomorphism of groups; in particular, since $(a_1, a_2) \in H_1 \times H_2$ has order $n_1 n_2$, so must its image $a_1 + a_2 \in H_1 + H_2$. □

For an abelian group G, we say that an integer k **kills** G if $kG = \{0_G\}$. Consider the set $\mathcal{K}_G$ of integers that kill G. Evidently, $\mathcal{K}_G$ is a subgroup of $\mathbb{Z}$, and hence of the form $m\mathbb{Z}$ for a uniquely determined non-negative integer m. This integer m is called the **exponent** of G. If $m \neq 0$, then we see that m is the least positive integer that kills G.

We first state some basic properties.

Theorem 8.81 *Let G be an abelian group of exponent m.*

1. *For any integer k such that $kG = \{0_G\}$, we have $m \mid k$.*
2. *If G has finite order, then m divides $|G|$.*
3. *If $m \neq 0$, for any $a \in G$, the order of a is finite, and $\mathrm{ord}(a) \mid m$.*

Proof. Exercise. □

Theorem 8.82 *For finite abelian groups G_1, G_2 whose exponents are m_1 and m_2, the exponent of $G_1 \times G_2$ is $\mathrm{lcm}(m_1, m_2)$.*

Proof. Exercise. □

Theorem 8.83 *If a finite abelian group G has exponent m, then G contains an element of order m. In particular, a finite abelian group is cyclic if and only if its order equals its exponent.*

Proof. The second statement follows immediately from the first. For the first statement, assume that $m > 1$, and let $m = \prod_{i=1}^{r} p_i^{e_i}$ be the prime factorization of m.

First, we claim that for each $1 \leq i \leq r$, there exists $a_i \in G$ such that $(m/p_i)a_i \neq 0_G$. Suppose the claim were false: then for some i, $(m/p_i)a = 0_G$ for all $a \in G$; however, this contradicts the minimality property in the definition of the exponent m. That proves the claim.

Let $a_1, \ldots, a_r$ be as in the above claim. Then by Theorem 8.79, $(m/p_i^{e_i})a_i$ has order $p_i^{e_i}$ for each $1 \leq i \leq r$. Finally, by Theorem 8.80, the group element

$$(m/p_1^{e_1})a_1 + \cdots + (m/p_r^{e_r})a_r$$

has order m. □

Theorem 8.84 *If G is a finite abelian group of order n, and p is a prime dividing n, then G contains an element of order p.*

Proof. First, note that if G contains an element whose order is divisible by p, then it contains an element of order p; indeed, if a has order mp, then ma has order p.

Let $a_1, \ldots, a_n$ be an enumeration of all the elements of G, and consider the "tower" of subgroups

$$H_0 := \{0_G\}, \quad H_i := \langle a_1, \ldots, a_i \rangle \quad (i = 1, \ldots, n).$$

We have

$$n = |H_n|/|H_0| = \prod_{i=1}^{n} |H_i|/|H_{i-1}| = \prod_{i=1}^{n} |H_i/H_{i-1}|,$$

and therefore, for some $1 \leq i \leq n$, $p \mid |H_i/H_{i-1}|$. Let $k = |H_i/H_{i-1}|$. Now, the quotient group H_i/H_{i-1} is clearly cyclic and is generated by the coset $a_i + H_{i-1}$. Let $k' = \mathrm{ord}(a_i)$. Then $k'(a_i + H_{i-1}) = k'a_i + H_{i-1} = 0_G + H_{i-1}$. Therefore, $k \mid k'$. That proves that $p \mid \mathrm{ord}(a_i)$, so we are done. □

With this last theorem, we can prove the converse of Theorem 8.77.

Theorem 8.85 *If G is a finite abelian group of order n, and $mG = G$, then m is relatively prime to n.*

Proof. To the contrary, suppose that p is a prime dividing m and n. Then G contains an element of order p by Theorem 8.84, and this element is in the kernel of the m-multiplication map. Therefore, this map is not injective, and hence not surjective since G is finite. Thus, $mG \neq G$, a contradiction. □

We also have:

Theorem 8.86 *Let G be a finite abelian group. Then the primes dividing the exponent of G are the same as the primes dividing its order.*

Proof. Since the exponent divides the order, any prime dividing the exponent must divide the order. Conversely, if a prime p divides the order, then since there is an element of order p in the group, the exponent must be divisible by p. □

8.6 ♣ The Structure of Finite Abelian Groups

We next state a theorem that characterizes all finite abelian groups up to isomorphism.

Theorem 8.87 (Fundamental Theorem of Finite Abelian Groups)
A finite abelian group (with more than one element) is isomorphic to a direct product of cyclic groups

$$\mathbb{Z}_{p_1^{e_1}} \times \cdots \times \mathbb{Z}_{p_r^{e_r}},$$

where the p_i are primes (not necessarily distinct) and the e_i are positive integers. This direct product of cyclic groups is unique up to the order of the factors.

An alternative characterization of this theorem is the following:

Theorem 8.88 *A finite abelian group (with more than one element) is isomorphic to a direct product of cyclic groups*

$$\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_t},$$

where all $m_i > 1$ and $m_1 \mid m_2 \mid \cdots \mid m_t$. Moreover, the integers $m_1, \ldots, m_t$ are unique, and m_t is the exponent of the group.

Exercise 8.89 Show that the above two theorems are equivalent, i.e., that each one implies the other. To do this, give a natural one-to-one correspondence between sequences of prime powers (as in Theorem 8.87) and sequences of integers $m_1, \ldots, m_t$ (as in Theorem 8.88), and also make use of Example 8.68. □

Exercise 8.90 Using the Fundamental Theorem of Finite Abelian Groups (either form), give short and simple proofs of Theorems 8.83 and 8.84. □

We now prove Theorem 8.88, which we break into two lemmas, the first of which proves the existence part of the theorem, and the second of which proves the uniqueness part.

Lemma 8.91 *A finite abelian group (with more than one element) is isomorphic to a direct product of cyclic groups*

$$\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_t},$$

where all $m_i > 1$ and $m_1 \mid m_2 \mid \cdots \mid m_t$, and m_t is the exponent of the group.

Proof. Let G be a finite abelian group with more than one element, and let m be the exponent of G. By Theorem 8.83, there exists an element $a \in G$ of order m. Let $A = \langle a \rangle$. Then $A \cong \mathbb{Z}_m$. Now, if $A = G$, the lemma is proved. So assume that $A \subsetneq G$.

We will show that there exists a subgroup B of G such that $G = A + B$ and $A \cap B = \{0\}$. From this, Theorem 8.55 gives us an isomorphism of G with $A \times B$. Moreover, the exponent of B is clearly a divisor of m, and so the lemma will follow by induction (on the order of the group).

So it suffices to show the existence of a subgroup B as above. We prove this by contradiction. Suppose that there is no such subgroup, and among all subgroups B such that $A \cap B = \{0\}$, assume that B is maximal, i.e., there is no subgroup B' of G such that $B \subsetneq B'$ and $A \cap B' = \{0\}$. By assumption $C := A + B \subsetneq G$.

Let p be any prime divisor of $|G/C|$. By Theorem 8.84, there exists an element $d + C$ of order p in G/C. We shall define a group element d' with slightly nicer properties than d, as follows. Since $pd \in C$, we have $pd = sa + b$ for some $s \in \mathbb{Z}$ and $b \in B$. We claim that $p \mid s$. To see this, first note that $p \mid m$. So we have $0 = md = (m/p)pd = (m/p)sa + (m/p)b$, and since $A \cap B = \{0\}$, we have $(m/p)sa = 0$, which can only happen if $p \mid s$. That proves the claim. This allows us to define $d' := d - (s/p)a$. Since $d \equiv d' \pmod{C}$, we see that $d' + C$ also has order p in G/C, but also that $pd' \in B$.

We next show that $A \cap (B + \langle d' \rangle) = \{0\}$, which will yield the contradiction we seek, and thus prove the lemma. To this end, it will suffice to show that $A \cap (B + \langle d' \rangle) \subset B$. Now, suppose we have a group element $xd' + b' \in A$, where $x \in \mathbb{Z}$ and $b' \in B$. Then in particular, $xd' \in C$, and so $p \mid x$, since $d' + C$ has order p in G/C. Further, since $pd' \in B$, we have $xd' \in B$, whence $xd' + b' \in B$. □

Lemma 8.92 *Suppose that* $G := \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_t}$ *and* $H := \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_t}$ *are isomorphic, where the* m_i*'s and* n_i*'s are positive integers (possibly* 1*) such that* $m_1 \mid \cdots \mid m_t$ *and* $n_1 \mid \cdots \mid n_t$*. Then* $m_i = n_i$ *for* $1 \leq i \leq t$*.*

Proof. Clearly, $\prod_i m_i = |G| = |H| = \prod_i n_i$. We prove the lemma by induction on the order of the group. If the group order is 1, then clearly all m_i and n_i must be 1, and we are done. Otherwise, let p be a prime dividing the group order. Now, suppose that p divides $m_r, \ldots, m_t$ (but not $m_1, \ldots, m_{r-1}$) and that p divides $n_s, \ldots, n_t$ (but not $n_1, \ldots, n_{s-1}$), where $r \leq t$ and $s \leq t$. Evidently, the groups pG and pH are isomorphic. Moreover,

$$pG \cong \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_{r-1}} \times \mathbb{Z}_{m_r/p} \times \cdots \times \mathbb{Z}_{m_t/p},$$

and

$$pH \cong \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_{s-1}} \times \mathbb{Z}_{n_s/p} \times \cdots \times \mathbb{Z}_{n_t/p}.$$

Thus, we see that $|pG| = |G|/p^{t-r+1}$ and $|pH| = |H|/p^{t-s+1}$, from which it follows that $r = s$, and the lemma then follows by induction. □

Chapter 9

Rings

This chapter reviews the notion of a ring, more specifically, a commutative ring with unity.

9.1 Definitions, Basic Properties, and Examples

Definition 9.1 *A **commutative ring with unity** is a set R together with addition and multiplication operators on R, such that*

1. *the set R under addition forms an abelian group, and we denote the additive identity by 0_R;*
2. *multiplication is commutative, i.e., for all $a, b \in R$, we have $ab = ba$;*
3. *multiplication is associative, i.e., for all $a, b, c \in R$, we have $a(bc) = (ab)c$;*
4. *multiplication distributes over addition, i.e., for all $a, b, c \in R$, $a(b + c) = ab = ac$;*
5. *there exists a multiplicative identity, i.e., there exists an element $1_R \in R$, such that $1_R \cdot a = a$ for all $a \in R$.*

There are other, more general (and less convenient) types of rings, but we will not be discussing them here. Therefore, to simplify terminology, from now on we will refer to a commutative ring with unity simply as a **ring**.

Notice that for any fixed $a \in R$, the map from R to R that sends $b \in R$ to $ab \in R$ is a group homomorphism with respect to the underlying additive group of R. We call this the **a-multiplication map**.

We first state some simple facts:

Theorem 9.2 *Let R be a ring. Then*

1. *the multiplicative identity 1_R is unique;*
2. *$0_R \cdot a = 0_R$ for all $a \in R$;*

3. $(-a)b = a(-b) = -(ab)$ *for all* $a, b \in R$;

4. $(-a)(-b) = ab$ *for all* $a, b \in R$;

5. $(na)b = a(nb) = n(ab)$ *for all* $n \in \mathbb{Z}$ *and* $a, b \in R$;

Proof. Part (1) may be proved using the same argument as was used to prove part (1) of theorem 8.2. Parts (2), (3), and (5) follow directly from parts (1), (2), and (3) of Theorem 8.51, using appropriate multiplication maps, discussed above. Part (4) follows from part (3) and part (4) of Theorem 8.17. □

Example 9.3 The set $\mathbb{Z}$ under the usual rules of multiplication and addition forms a ring. □

Example 9.4 For $n \geq 1$, the set $\mathbb{Z}_n$ under the rules of multiplication and addition defined in §2.3 forms a ring. □

Example 9.5 The set $\mathbb{Q}$ of rational numbers under the usual rules of multiplication and addition forms a ring. □

Example 9.6 The set $\mathbb{R}$ of real numbers under the usual rules of multiplication and addition forms a ring. □

Example 9.7 The set $\mathbb{C}$ of complex numbers under the usual rules of multiplication and addition forms a ring. Recall that any complex number z may be written $z = a + bi$, for $a, b \in \mathbb{R}$. For $z := a + bi \in \mathbb{C}$ and $z' := a' + b'i$, we have $z + z' := (a + a') + (b + b')i$ and $zz' := (aa' - bb') + (ab' - ab')i$. In particular, note that $i^2 = -1$. The fact that $\mathbb{C}$ is a ring can be derived, by direct calculation, from the fact that $\mathbb{R}$ is a ring, and the above definitions of addition and multiplication in $\mathbb{C}$; however, we shall see later that this follows more easily from more general considerations.

Recall the **complex conjugation** operation, that sends $z := a + bi \in \mathbb{C}$ to $\bar{z} := a - bi$. One can verify by direct calculation that complex conjugation is both additive and multiplicative; that is, for all $z, z' \in \mathbb{C}$, we have (1) $\overline{z + z'} = \bar{z} + \bar{z}'$, and (2) $\overline{z \cdot z'} = \bar{z} \cdot \bar{z}'$.

For $z \in \mathbb{C}$, the **norm** of z is $N(z) := z\bar{z}$. If $z := a + bi$, then $N(z) = a^2 + b^2$, and so we see that $N(z)$ is a non-negative real number, and is zero iff $z = 0$. Moreover, from the multiplicativity of complex conjugation, it is easy to see that the norm is multiplicative as well: $N(zz') = zz'\overline{zz'} = zz'\bar{z}\bar{z}' = N(z)N(z')$. □

Note that in a ring R, if $1_R = 0_R$, then for all $a \in R$, $a = 1_R \cdot a = 0_R \cdot a = 0_R$, and hence the ring R is **trivial**, in the sense that it consists of the single element 0_R, with $0_R + 0_R = 0_R$ and $0_R \cdot 0_R = 0_R$. If $1_R \neq 0_R$, we say that R is **non-trivial**. We shall rarely be concerned with trivial rings for their own sake; however, they do sometimes arise in certain constructions.

If $R_1, \ldots, R_k$ are rings, then the set of all k-tuples $(a_1, \ldots, a_k)$ with $a_i \in R_i$ for $1 \leq i \leq k$, with addition and multiplication defined component-wise, forms a ring. The ring is denoted $R_1 \times \cdots \times R_k$, and is called the **direct product** of $R_1, \ldots, R_k$.

The **characteristic** of a ring R is defined as the exponent of the underlying additive group (see §8.5). Equivalently, the characteristic if the least positive integer m such that $m \cdot 1_R = 0_R$, if such an m exists, and is zero otherwise.

Example 9.8 The ring $\mathbb{Z}$ has characteristic zero, $\mathbb{Z}_n$ has characteristic n, and $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ has characteristic $\mathrm{lcm}(n_1, n_2)$. □

For $a, b \in R$, we say that b **divides** a, written $b \mid a$, if there exists $c \in R$ such that $a = bc$, in which case we say that b is a **divisor** of a.

Note that parts 1-5 of Theorem 1.1 holds for an arbitrary ring.

When there is no possibility for confusion, one may write "0" instead of "0_R" and "1" instead of "1_R." Also, one may also write, e.g., 2_R to denote $2 \cdot 1_R$, 3_R to denote $3 \cdot 1_R$, etc., and where the context is clear, one may use an implicit "type cast," so that $m \in \mathbb{Z}$ really means $m \cdot 1_R$.

Exercise 9.9 Show that the familiar "binomial theorem" holds in an arbitrary ring R; i.e., for $a, b \in R$ and positive integer n, we have

$$(a + b)^n = \sum_{i=0}^{n} \binom{n}{i} a^{n-i} b^i.$$

□

9.1.1 Units and Fields

Let R be a ring. We call $u \in R$ a **unit** if it has a multiplicative inverse, i.e., if $uu' = 1_R$ for some $u' \in R$. It is easy to see that the multiplicative inverse of u, if its exists, is unique, and we denote it by u^{-1}; also, for $a \in R$, we may write a/u to denote au^{-1}. It is clear that a unit u divides every $a \in R$.

We denote the set of units R^*. It is easy to verify that the set R^* is closed under multiplication, from which it follows that R^* is an abelian group, called the **multiplicative group of units** of R.

If R is non-trivial and R^* contains all non-zero elements of R, i.e., every non-zero element of R has a multiplicative inverse, then R is called a **field**.

Example 9.10 The only units in the ring $\mathbb{Z}$ are ± 1. Hence, $\mathbb{Z}$ is not a field. □

Example 9.11 For $n > 1$, the units in $\mathbb{Z}_n$ are the residue classes $[a \bmod n]$ with $\gcd(a, n) = 1$. In particular, if n is prime, all non-zero residue classes are units, and conversely, if n is composite, some non-zero residue classes are not units. Hence, $\mathbb{Z}_n$ is a field if and only if n is prime. □

Example 9.12 Every non-zero element of $\mathbb{Q}$ is a unit. Hence, $\mathbb{Q}$ is a field. □

Example 9.13 Every non-zero element of $\mathbb{R}$ is a unit. Hence, $\mathbb{R}$ is a field. □

Example 9.14 For non-zero $z := a + bi \in \mathbb{C}$, we have $c := N(z) = a^2 + b^2 > 0$. It follows that the complex number $\bar{z}c^{-1} = (ac^{-1}) + (-bc^{-1})i$ is the multiplicative inverse of z. Hence, every non-zero element of $\mathbb{C}$ is a unit, and so, $\mathbb{C}$ is a field. □

Example 9.15 In this example, we present a specific field F of size 4. We write the elements of F as pairs of bits: 00, 01, 10, 11. Addition is bit-wise exclusive-or, so that 00 is the additive identity. Multiplication in F is defined by the following table:

	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Observe that 01 acts as the multiplicative identity. The reader may verify by inspection that this indeed defines a field. The non-zero elements F^* form a group under multiplication, and in fact it is a cyclic group; the reader may check that both 10 and 11 are generators. Thus, F^* is isomorphic to the additive group $\mathbb{Z}_3$.

As we shall see later, any finite field must be of size p^w for some prime p and positive integer w, and moreover, for every such p and w, there exists an essentially unique field of size p^w. □

Example 9.16 For two rings R_1, R_2, the group of units of $R_1 \times R_2$ is $R_1^* \times R_2^*$. In particular, a direct product of non-trivial rings cannot be a field. □

9.1.2 Zero divisors and Integral Domains

Let R be a ring. An element $a \in R$ is called a **zero divisor** if $a \neq 0_R$ and there exists non-zero $b \in R$ such that $ab = 0_R$.

If R is non-trivial and has no zero divisors, then it is called an **integral domain**. Put another way, a non-trivial ring R is an integral domain if and only if $ab = 0_R$ implies $a = 0_R$ or $b = 0_R$ for all $a, b \in R$.

Note that if u is a unit in R, it cannot be a zero divisor (if $ub = 0_R$, then multiplying both sides of this equation by u^{-1} yields $b = 0_R$). In particular, it follows that any field is an integral domain.

Example 9.17 $\mathbb{Z}$ is an integral domain. □

Example 9.18 For $n > 1$, $\mathbb{Z}_n$ is an integral domain if and only if n is prime. In particular, if n is composite, so $n = n_1 n_2$ with $1 < n_1, n_2 < n$, then $[n_1]$ and $[n_2]$ are zero divisors: $[n_1][n_2] = [0]$, but $[n_1] \neq [0]$ and $[n_2] \neq [0]$. □

Example 9.19 $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ are fields, and hence, are also integral domains. □

Example 9.20 For two rings R_1, R_2, an element $(a_1, a_2) \in R_1 \times R_2$ is a zero divisor if and only if a_1 is a zero divisor, a_2 is a zero divisor, or exactly one of a_1 or a_2 is zero. In particular, a direct product ring cannot be an integral domain. □

We have the following "cancellation law":

Theorem 9.21 *If R is a ring, and $a, b, c \in R$ such that $a \neq 0_R$ and a is not a zero divisor, then $ab = ac$ implies $b = c$.*

Proof. $ab = bc$ implies $a(b - c) = 0_R$. The fact that $a \neq 0$ and a is not a zero divisor implies that we must have $b - c = 0_R$, i.e., $b = c$. □

Theorem 9.22 *If D is an integral domain, then*

1. *for all $a, b, c \in D$, $a \neq 0_D$ and $ab = ac$ implies $b = c$;*
2. *for all $a, b \in D$, $a \mid b$ and $b \mid a$ if and only if $a = bc$ for $c \in D^*$.*
3. *for all $a, b \in D$ with $b \neq 0_D$ and $b \mid a$, then there is a unique $c \in D$ such that $a = bc$, which we may denote as a/b.*

Proof. The first statement follows immediately from the previous theorem and the definition of an integral domain.

For the second statement, if $a = bc$ for $c \in D^*$, then we also have $b = ac^{-1}$; thus, $b \mid a$ and $a \mid b$. Conversely, $a \mid b$ implies $b = ax$ for $x \in D$, and $b \mid a$ implies $a = by$ for $y \in D$, and hence $b = bxy$. If $b = 0_R$, then the equation $a = by$ implies $a = 0_R$, and so the statement holds for any c; otherwise, cancel b, we have $1_D = xy$, and so x and y are units.

For the third statement, if $a = bc$ and $a = bc'$, then $bc = ac'$, and cancel b. □

Theorem 9.23 *Any finite integral domain is a field.*

Proof. Let D be a finite integral domain, and let a be any non-zero element of D. Consider the a-multiplication map that sends $b \in D$ to ab, which is a group homomorphism on the additive group of D. Since a is not a zero-divisor, it follows that the kernel of the a-multiplication map is $\{0_D\}$, hence the map is injective, and by finiteness, it must be surjective as well. In particular, there must be an element $b \in D$ such that $ab = 1_D$. □

9.1.3 Subrings

A subset R' of a ring R is called a **subring** if

- R' is an additive subgroup of R,
- R' is closed under multiplication, and

- $1_R \in R'$.

It is clear that the operations of addition and multiplication on R make R' itself into a ring, where 0_R is the additive identity of R' and 1_R is the multiplicative identity of R'. One may also call R an **extension ring** of R'.

Some texts do not require that 1_R belongs to R', and instead require only that R' contains a multiplicative identity, which may be different than that of R. This is perfectly reasonable, but for simplicity, we restrict ourselves to the case when $1_R \in R'$.

Example 9.24 $\mathbb{Z}$ is a subring of $\mathbb{Q}$. □

Example 9.25 $\mathbb{Q}$ is a subring of $\mathbb{R}$. □

Example 9.26 $\mathbb{R}$ is a subring of $\mathbb{C}$, where we identify $a \in \mathbb{R}$ with the complex number $a + 0i$. Note that for $z := a + bi \in \mathbb{C}$, we have $\bar{z} = z$ iff $a + bi = a - bi$ iff $b = 0$. That is, $\bar{z} = z$ iff $z \in \mathbb{R}$. □

Example 9.27 The set $\mathbb{Z}[i]$ of complex numbers of the form $a + bi$, with $a, b \in \mathbb{Z}$, is a subring of $\mathbb{C}$. It is called the ring of **Gaussian integers**. Since $\mathbb{C}$ is a field, it contains no zero divisors, and hence $\mathbb{Z}[i]$ contains no zero divisors. Hence, $\mathbb{Z}[i]$ is an integral domain.

Let us determine the units of $\mathbb{Z}[i]$. If $z \in \mathbb{Z}[i]$ is a unit, then there exists $z' \in \mathbb{Z}[i]$ such that $zz' = 1$. Taking norms, we obtain

$$1 = N(1) = N(zz') = N(z)N(z').$$

Clearly, the norm of a Gaussian integer is a non-negative integer, and so $N(z)N(z') = 1$ implies $N(z) = 1$. Now, if $z := a + bi$, then $N(z) = a^2 + b^2$, and so $N(z) = 1$ implies $z = \pm 1$ or $z = \pm i$. Conversely, it is clear that ± 1 and $\pm i$ are indeed units, and so these are the only units in $\mathbb{Z}[i]$. □

Example 9.28 Consider the field F defined in Example 9.15. The subset F' *deq*$\{\texttt{00}, \texttt{01}\}$ is a subring of F, and is in fact a field. □

Example 9.29 If R and S are rings, then $R' := R \times \{0_S\}$ satisfies the first two requirements of the definition of a subring, but it does not satisfy the third. However, R' does contain an element that acts as a multiplicative identity of R', namely $(1_R, 0_S)$, and hence could be viewed as a subring of $R \times S$ under a more liberal definition. □

Theorem 9.30 *Any subring of an integral domain is also an integral domain.*

Proof. If D' is a subring of the integral domain D, then any zero divisor in D' would itself be a zero divisor in D. □

Note that it is not the case that a subring of a field is always a field: the subring $\mathbb{Z}$ of $\mathbb{Q}$ is a counter-example. If F' is a subring of a field F, and F' is itself a field, then we say that F' is a **subfield** of F, and that F is an **extension field** of F'.

Example 9.31 $\mathbb{Q}$ is a subfield of $\mathbb{R}$ is a subfield of $\mathbb{C}$. □

Exercise 9.32 Show that the set $\mathbb{Q}[i]$ of complex numbers of the form $a+bi$, with $a, b \in \mathbb{Q}$, is a subfield of $\mathbb{C}$. □

Exercise 9.33 Show that if R' and R'' are subrings of R, then so is $R' \cap R''$. □

Exercise 9.34 Show that if R is a ring, and R' is a subset of R containing 1_R, and is closed under addition and multiplication, then R' is a subring of R. □

9.2 Polynomial rings

If R is a ring, then we can form the **ring of polynomials** $R[\mathtt{X}]$, consisting of all polynomials $\sum_{i=0}^{k} a_i \mathtt{X}^i$ in the indeterminate, or "formal" variable, $\mathtt{X}$, with coefficients in R, with addition and multiplication being defined in the usual way. To state the rules precisely but simply, we temporarily consider "'polynomials" with terms involving arbitrary powers i of $\mathtt{X}$, both positive and negative, where it is understood that all coefficients are zero, except for a finite number of non-negative values of i. With this convention, if

$$a = \sum_{i=-\infty}^{\infty} a_i \mathtt{X}^i \quad \text{and} \quad b = \sum_{i=-\infty}^{\infty} b_i \mathtt{X}^i,$$

then

$$a + b := \sum_{i=-\infty}^{-\infty} (a_i + b_i)\mathtt{X}^i, \tag{9.1}$$

and

$$a \cdot b := \sum_{i=-\infty}^{-\infty} \Big(\sum_{j+k=i} a_j b_k \Big) \mathtt{X}^i, \tag{9.2}$$

where the inner sum is over all pairs of indices j, k such that $j + k = i$; note that there are only a finite number of non-zero terms in this sum.

For $a = \sum_{i=0}^{k} a_i \mathtt{X}^i \in R[\mathtt{X}]$, if $k = 0$, we call a a **constant** polynomial, and if $k > 0$ and $a_k \neq 0_R$, we call a a **non-constant** polynomial.

Clearly, R is a subring of $R[\mathtt{X}]$, and consists precisely of the constant polynomials of $R[\mathtt{X}]$. In particular, 0_R is the additive identity of $R[\mathtt{X}]$, and 1_R is the multiplicative identity of $R[\mathtt{X}]$. Note that if R is the trivial ring, then so is $R[\mathtt{X}]$.

9.2.1 Polynomials versus polynomial functions

Of course, a polynomial $a = \sum_{i=0}^{k} a_i \mathtt{X}^i$ defines a polynomial function on R that sends $x \in R$ to $\sum_{i=0}^{k} a_i x^i$, and we denote the value of this function as $a(x)$. However, it is important to to regard polynomials over R as formal expressions, and not to identify them with their

corresponding functions. In particular, a polynomial $a = \sum_{i=0}^{k} a_i \mathtt{X}^i$ is zero if and only if $a_i = 0_R$ for $0 \leq i \leq k$, and two polynomials are equal if and only if their difference is zero. This distinction is important, since there are rings R over which two different polynomials define the same function. One can of course define the ring of polynomial functions on R, but in general, that ring has a different structure from the ring of polynomials over R.

Example 9.35 In the ring $\mathbb{Z}_p$, for prime p, by Theorem 8.72 (Fermat's Little Theorem), we have $x^p - x = [0]$ for all $x \in \mathbb{Z}_p$. But consider the polynomial $a = \mathtt{X}^p - \mathtt{X} \in \mathbb{Z}_p[\mathtt{X}]$. We have $a(x) = 0_R$ for all $x \in 0_R$, and hence the function defined by a is the zero function, yet a is definitely *not* the zero polynomial. □

More generally, if R is a subring of a ring S, a polynomial $a = \sum_{i=0}^{k} a_i \mathtt{X}^i \in R[\mathtt{X}]$ defines a polynomial function from R to S that sends $x \in R$ to $\sum_{i=0}^{k} a_i x^i \in S$, and the value of this function is denoted $a(x)$.

A simple, but important, fact is the following:

Theorem 9.36 *Let R be a subring of a ring S. Then for $a, b \in R[\mathtt{X}]$ and $x \in S$, we have $(ab)(x) = a(x)b(x)$ and $(a + b)(x) = a(x) + b(x)$.*

Proof. Exercise. □

Note that the syntax for evaluating polynomial functions creates some ambiguities: for $a, b, c \in R[\mathtt{X}]$, one could interpret $a(b + c)$ as either a times $b + c$, or a evaluated at $b + c$; to avoid such ambiguities, if the intended meaning is the former, one should write this as, say, $a \cdot (b + c)$ or $(b + c)a$.

So as to keep the distinction between ring elements and indeterminates clear, we shall use the symbol "$\mathtt{X}$" only to denote the latter. Also, for a polynomial $a \in R[\mathtt{X}]$, we shall in general not write this as "$a(\mathtt{X})$," but simply as "a." Of course, the choice of the symbol "$\mathtt{X}$" is arbitrary; occasionally, we may use other symbols, such as "$\mathtt{Y}$," as alternatives.

9.2.2 Basic properties of polynomial rings

Let R be a ring.

For non-zero $a \in R[\mathtt{X}]$, if $a = \sum_{i=0}^{k} a_i \mathtt{X}^i$ with $a_k \neq 0_R$, we call k the **degree** of a, denoted $\deg(a)$, and we call a_k the **leading coefficient** of a, denoted $\mathrm{lc}(a)$, and we call a_0 the **constant term** of a. If $\mathrm{lc}(a) = 1_R$, then a is called **monic**.

Note that if $a, b \in R[\mathtt{X}]$, both non-zero, and their leading coefficients are not both zero divisors, then the product ab is non-zero and $\deg(ab) = \deg(a) + \deg(b)$. However, if the leading coefficients of a and b are both zero divisors, then we could get some "collapsing": we could have $ab = 0_R$, or $ab \neq 0_R$ but $\deg(ab) < \deg(a) + \deg(b)$.

For the zero polynomial, we establish the following conventions: its leading coefficient and constant term are defined to be 0_R, and its degree is defined to be $-\infty$ (see §A.4).

This notion of "negative infinity" should not be construed as a particularly meaningful algebraic notion — it is simply a convenience of notation; for example, it allows us to succinctly state that

for all $a, b \in R[\mathtt{X}]$, $\deg(ab) \leq \deg(a) + \deg(b)$, with equality holding unless the leading coefficients of both a and b are zero divisors.

Theorem 9.37 *Let D be an integral domain. Then*

1. *for all $a, b \in D[\mathtt{X}]$, $\deg(ab) = \deg(a) + \deg(b)$;*
2. *$D[\mathtt{X}]$ is an integral domain;*
3. *$(D[\mathtt{X}])^* = D^*$.*

Proof. Exercise. □

9.2.3 Division with remainder

An extremely important property of polynomials is a division with remainder property, analogous to that for the integers:

Theorem 9.38 (Division with Remainder Property) *Let R be a non-trivial ring. For $a, b \in R[\mathtt{X}]$ with $\mathrm{lc}(b) \in R^*$, there exist unique $q, r \in R[\mathtt{X}]$ such that $a = bq + r$ and $\deg(r) < \deg(b)$.*

Proof. Consider the set S of polynomials of the form $a - zb$ with $z \in R[\mathtt{X}]$. Let $r = a - qb$ be an element of S of minimum degree. We must have $\deg(r) < \deg(b)$, since otherwise, we would have $r' := r - (\mathrm{lc}(r)\,\mathrm{lc}(b)^{-1}\mathtt{X}^{\deg(r)-\deg(b)}) \cdot b \in S$, and $\deg(r') < \deg(r)$, contradicting the minimality of $\deg(r)$.

That proves the existence of r and q. For uniqueness, suppose that $a = bq + r$ and $a = bq' + r'$, where $\deg(r) < \deg(b)$ and $\deg(r') < \deg(b)$. This implies $r' - r = b(q - q')$. However, if $q \neq q'$, then

$$\deg(b) > \deg(r' - r) = \deg(b(q - q')) = \deg(b) + \deg(q - q') \geq \deg(b),$$

which is impossible. Therefore, we must have $q = q'$, and hence $r = r'$. □

If $a = bq + r$ as in the above theorem, we define $a \operatorname{rem} b := r$. Clearly, $b \mid a$ if and only if $a \operatorname{rem} b = 0$.

As a special case of the above theorem, we have:

Theorem 9.39 *If F is field, then for $a, b \in F[\mathtt{X}]$ with $b \neq 0_F$, there exist unique $q, r \in F[\mathtt{X}]$ such that $a = bq + r$ and $\deg(r) < \deg(b)$.*

Theorem 9.40 *For a non-trivial ring R and $a \in R[\mathtt{X}]$ and $x \in R$, $a(x) = 0_R$ if and only if $(\mathtt{X} - x)$ divides a.*

Proof. Let us write $a = (\mathtt{X} - x)q + r$, with $q, r \in R[\mathtt{X}]$ and $\deg(r) < 1$, which means that $r \in R$. Then we have $a(x) = (x - x)q(x) + r = r$. Thus, $a(x) = 0$ if and only if $\mathtt{X} - x$ divides a. □

With R, a, x as in the above theorem, we say that x is a **root** of a if $a(x) = 0_R$.

Theorem 9.41 *Let D be an integral domain, and let $a \in D[\mathtt{X}]$, with $\deg(a) = k \geq 0$. Then a has at most k roots.*

Proof. We can prove this by induction. If $k = 0$, this means that a is a non-zero element of D, and so it clearly has no roots.

Now suppose that $k > 0$. If a has no roots, we are done, so suppose that a has a root x. Then we can write $a = (\mathtt{X} - x)q$, where $\deg(q) = k - 1$. Now, for any root y of a with $y \neq x$, we have $0_D = a(y) = (y - x)q(y)$, and using the fact that D is an integral domain, we must have $q(y) = 0$. Thus, the only roots of a are x and the roots of q. By induction, q has at most $k - 1$ roots, and hence a has at most k roots. □

Theorem 9.42 *Let D be an infinite integral domain, and let $a \in D[\mathtt{X}]$. If $a(x) = 0_D$ for all $x \in D$, then $a = 0_D$.*

Proof. Exercise. □

With this last theorem, one sees that for an infinite integral domain D, there is a one-to-one correspondence between polynomials over D and polynomial functions on D.

Exercise 9.43 Let F be a field of characteristic other than 2, so that the $2_F \neq 0_F$. Show that the familiar "quadratic formula" holds for F. That is, for $a, b, c \in F$ with $a \neq 0_F$, the polynomial $f := a\mathtt{X}^2 + b\mathtt{X} + c \in F[\mathtt{X}]$ has a root if and only if there exists $z \in F$ such that $z^2 = d$, where d is the discriminant of f, i.e., $d := b^2 - 4ac$, in which case the roots of f are

$$\frac{-b \pm z}{2a}.$$

□

9.2.4 Multi-variate polynomials

Consider the ring $R[\mathtt{X}]$ of polynomials over a ring R. If $\mathtt{Y}$ is another indeterminate, we can form the ring $R[\mathtt{X}][\mathtt{Y}]$ of polynomials in $\mathtt{Y}$ whose coefficients are themselves polynomials in $\mathtt{X}$ over the ring R. We can write $R[\mathtt{X}, \mathtt{Y}]$ instead of $R[\mathtt{X}][\mathtt{Y}]$. Any element of $R[\mathtt{X}, \mathtt{Y}]$ is called a **bivariate polynomial**, and can be expressed uniquely as a sum of **monomials**, where each monomial is of the form $c\mathtt{X}^i\mathtt{Y}^j$ for non-zero $c \in R$ and non-negative integers i and j. The **degree** of such a monomial $c\mathtt{X}^i\mathtt{Y}^j$ is defined to be $i + j$, and for non-zero $a \in R[\mathtt{X}, \mathtt{Y}]$, the **degree** of a, denoted $\deg(a)$, is the maximum degree of the monomials of a. As for ordinary

(univariate) polynomials, the degree of 0 is defined to be $-\infty$. In general, for $a, b \in R[\mathtt{X}, \mathtt{Y}]$, we have $\deg(ab) \leq \deg(a) + \deg(b)$, while equality holds if R is an integral domain.

More generally, if $\mathtt{X}_1, \ldots, \mathtt{X}_n$ are indeterminates, we can form the ring $R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ of **multi-variate polynomials** in n variables over R. Formally, we can think of this ring as $R[\mathtt{X}_1][\mathtt{X}_2] \cdots [\mathtt{X}_n]$. Any multi-variate polynomial can be expressed uniquely as the sum of monomials of the form $c\mathtt{X}_1^{e_1} \cdots \mathtt{X}_n^{e_n}$ for non-zero $c \in R$ and non-negative integers $e_1, \ldots, e_n$; the degree of such a monomial is defined to be $\sum_i e_i$, and the degree of a multi-variate polynomial is defined to be the maximum degree of its monomials. As above, for $a, b \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$, we have $\deg(ab) \leq \deg(a) + \deg(b)$, while equality holds if R is an integral domain.

For $a \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ and $x = (x_1, \ldots, x_n) \in R^{\times n}$, we define $a(x)$ to be the element of R obtained by evaluating the expression obtained by substituting x_i for $\mathtt{X}_i$ in a.

Exercise 9.44 This exercise generalizes Theorem 9.41. Let D be an integral domain, and let $a \in D[\mathtt{X}_1, \ldots, \mathtt{X}_n]$, with $\deg(a) = k \geq 0$. Let S be a finite subset of D. Show that the number of elements $x \in S^{\times n}$ such that $a(x) = 0$ is at most $k|S|^{n-1}$. □

9.3 Ideals and Quotient Rings

Throughout this section, let R denote a ring.

Definition 9.45 *An **ideal** of R is a additive subgroup I of R that is closed under multiplication by elements of R, that is, for all $z \in I$ and $a \in R$, $za \in I$.*

Clearly, $\{0\}$ and R are ideals of R. From the fact that an ideal I is closed under multiplication by elements of R, it is easy to see that $I = R$ if and only if $1_R \in I$.

Example 9.46 For $m \in \mathbb{Z}$, the set $m\mathbb{Z}$ is not only an additive subgroup of $\mathbb{Z}$, it is also an ideal of the ring $\mathbb{Z}$. □

Example 9.47 For $m \in \mathbb{Z}$, the set $m\mathbb{Z}_n$ is not only an additive subgroup of $\mathbb{Z}_n$, it is also an ideal of the ring $\mathbb{Z}_n$. □

If $d_1, \ldots, d_k \in R$, then the set

$$d_1R_1 + \cdots + d_kR := \{d_1a_1 + \cdots + d_ka_k : a_1, \ldots, a_k \in R\}$$

is clearly an ideal, and contains $d_1, \ldots, d_k$. It is called the **ideal generated by** $d_1, \ldots, d_k$. Clearly, any ideal I that contains $d_1, \ldots, d_k$ must contain $d_1R_1 + \cdots + d_kR$. An alternative notation that is often used is to write $(d_1, \ldots, d_k)$ to denote the ideal generated by $d_1, \ldots, d_k$, when the ring R is clear from context. If an ideal I is equal to dR for some $d \in R$, then we say that I is a **principal ideal**.

Note that if I and J are ideals, then so are $I + J := \{x + y : x \in I, y \in J\}$ and $I \cap J$.

Throughout the rest of this section, I denotes an ideal of R.

Since I is an additive subgroup, we may adopt the congruence notation in §8.3, writing $a \equiv b \pmod{I}$ if and only if $a - b \in I$.

Note that if $I = dR$, then $a \equiv b \pmod{I}$ if and only if $d \mid (a - b)$, and as a matter of notation, one may simply write this congruence as $a \equiv b \pmod{d}$. More generally, if $I = (d_1, \ldots, d_k)$, one may write $a \equiv b \pmod{d_1, \ldots, d_k}$.

If we just consider R as an additive group, then as we saw in §8.3, we can form the additive group R/I of cosets, where $(a + I) + (b + I) := (a + b) + I$. By considering also the multiplicative structure of R, we can also view R/I as a ring. To do this, we need the following fact.

Theorem 9.48 *If $a \equiv a' \pmod{I}$ and $b \equiv b' \pmod{I}$, then $ab \equiv a'b' \pmod{I}$.*

Proof. If $a' = a + x$ for $x \in I$ and $b' = b + y$ for $y \in I$, then $a'b' = ab + ay + bx + xy$. Since I is closed under multiplication by elements of R, we see that $ay, bx, xy \in I$, and since it is closed under addition, $ay + bx + xy \in I$. Hence, $a'b' - ab \in I$. □

This theorem is perhaps one of the main motivations for the definition of an ideal. It allows us to define multiplication on R/I as follows: for $a, b \in R$,

$$(a + I) \cdot (b + I) := ab + I.$$

The above theorem is required to show that this definition is unambiguous. It is trivial to show that R/I satisfies the properties defining of a ring, using the corresponding properties for R.

This ring is called the **quotient ring** or **residue class ring of R modulo I**.

As a matter of notation, for $a \in R$, we define $[a \bmod I] := a + I$, and if $I = dR$, we may write this simply as $[a \bmod d]$. If I is clear from context, we may also just write $[a]$.

Example 9.49 For $n \geq 1$, the ring $\mathbb{Z}_n$ as we have defined it is precisely the quotient ring $\mathbb{Z}/n\mathbb{Z}$. □

Example 9.50 Let f be a monic polynomial over R with $\deg(f) = \ell > 0$, and consider the quotient ring $S = R[\mathtt{X}]/(f)$. Every element of S can be written uniquely as $[a \bmod f]$, where a is a polynomial over R of degree less than ℓ.

This follows from the division with remainder property for polynomials. Indeed, for every $b \in R[\mathtt{X}]$, there exist polynomials $q, a \in R[\mathtt{X}]$ with $\deg(a) < \ell$ and $b = fq + a$. Since $b - a = fq$, we have $b \equiv a \pmod{f}$. Moreover, there cannot be two distinct polynomials $a, a' \in R[\mathtt{X}]$, both of degree less than ℓ, such that $a \equiv a' \pmod{f}$, as this would imply $a - a' = fg$ for some non-zero polynomial $g \in R[\mathtt{X}]$, and this would imply $\ell > \deg(a - a') = \deg(f) + \deg(g) \geq \ell$, a contradiction. as this would uniqueness part of the division with remainder property. □

Exercise 9.51 Let p be a prime. Let R be the set of all rational numbers which can be expressed as a/b, where a and b are integers, and b is not divisible by p.

(a) Show that R is a subring of the field of rational numbers.

(b) Show that $a/b \in R$ (with $p \nmid b$) is a unit in R if and only if $p \nmid a$.

(c) Show that every ideal in R is of the form (p^i), for some integer $i \geq 0$.

□

Exercise 9.52 Show that if I is a subset of $R[\mathtt{X}]$ that is closed under addition, multiplication by elements of R, and multiplication by $\mathtt{X}$, then I is an ideal of $R[\mathtt{X}]$. □

Exercise 9.53 An ideal I of R is called **proper** if $I \subsetneq R$. A proper ideal I of R is called **prime** if for all $a, b \in R$, $ab \in I$ implies $a \in I$ or $b \in I$. A proper ideal I of R is called **maximal** if there are no proper ideals of J such that $I \subsetneq J$.

(a) Show that a proper ideal I is prime if and only if R/I is an integral domain.

(b) Show that a proper ideal I is maximal if and only if R/I is a field.

□

Exercise 9.54 Let R be a ring, and S a subset (possibly infinite) of R. Define the set $S \cdot R$ to be the set of all finite sums of the form

$$x_1 r_1 + \cdots + x_\ell r_\ell \quad (\text{with } x_k \in S,\ r_k \in R \text{ for } k = 1, \ldots, \ell).$$

Show that $S \cdot R$ is an ideal in R, and is the smallest ideal of R containing S. □

Exercise 9.55 Let I and J be two ideals in a ring R. We define the **product** $I \cdot J$ of I and J as the set containing all finite sums of the form

$$x_1 y_1 + \cdots + x_\ell y_\ell \quad (\text{with } x_k \in I,\ y_k \in J \text{ for } k = 1, \ldots, \ell).$$

(a) Show that $I \cdot J$ is an ideal.

(b) Show that if I and J are principal ideals, with $I = aR$ and $J = bR$, then $I \cdot J = abR$, and so is also a principal ideal.

(c) Show that $I \cdot J \subset I \cap J$.

(d) Show that if $I + J = R$, then $I \cdot J = I \cap J$.

□

Exercise 9.56 Suppose S is a subring of R, and I is an ideal of R. Show that $I \cap S$ is an ideal of S. □

9.4 Ring Homomorphisms and Isomorphisms

Throughout this section, R and R' denote rings.

Definition 9.57 *A function ρ from R to R' is called a **ring homomorphism** if it is a group homomorphism with respect to the underlying additive groups of R and R', and if in addition,*

1. *$\rho(ab) = \rho(a)\rho(b)$ for all $a, b \in R$, and*
2. *$\rho(1_R) = 1_{R'}$.*

*Moreover, if ρ is a bijection, then it is called a **ring isomorphism** of R with R', and if in addition, $R = R'$, then it is called a **ring automorphism** on R.*

Note that some texts do not require that $\rho(1_R) = 1_{R'}$.

It is easy to see (verify) that if $\rho : R \to R'$ and $\rho' : R' \to R''$ are ring homomorphisms, then so is their composition $\rho' \circ \rho : R \to R''$.

It is also easy to see (verify) that if ρ is a ring isomorphism of R with R', then the inverse function ρ^{-1} is a ring isomorphism of R' with R. If such an isomorphism exists, we say that R is **isomorphic** to R', and write $R \cong R'$. We stress that an isomorphism of R with R' is essentially just a "renaming" of elements; in particular, units map to units and zero divisors map to zero divisors.

A ring homomorphism ρ from R to R' is also a group homomorphism from the additive group of R to the additive group of R'. We may therefore adopt the terminology of kernel and image, as defined in §8.4, and note that all the results of Theorem 8.51 apply as well here. In particular, $\rho(a) = \rho(b)$ if and only if $a \equiv b \pmod{\ker(\rho)}$, and ρ is injective if and only if $\ker(\rho) = \{0_R\}$. However, we may strengthen Theorem 8.51 as follows:

Theorem 9.58 *Let $\rho : R \to R'$ be a ring homomorphism.*

1. *For any subring S of R, $\rho(S)$ is a subring of R'.*
2. *For any ideal I of R, $\rho(I)$ is an ideal of $\mathrm{im}(\rho)$.*
3. *$\ker(\rho)$ is an ideal of R.*
4. *For any ideal I' of R', $\rho^{-1}(I')$ is an ideal of R (and contains $\ker(\rho)$).*

Proof. Exercise. □

An injective ring homomorphism $\rho : R \to R'$ is called an **embedding** of R in R'. In this case, $\mathrm{im}(\rho)$ is a subring of R' and $R \cong \mathrm{im}(\rho)$, As a slight abuse of terminology, we shall often say that "R is a subring of R'" in this case, if the particular "canonical" embedding is clear from context.

Theorems 8.52, 8.53, and 8.54 also have natural analogs; to prove these theorems, all one has to show is that the homomorphisms on the underlying additive groups in those theorems are are also ring homomorphisms.

Theorem 9.59 *If I is an ideal R, then the map $\rho : R \to R/I$ given by $\rho(a) = a + I$ is a surjective ring homomorphism whose kernel is I. This is sometimes called the "natural" map from R to R/I.*

Proof. Exercise. □

Theorem 9.60 *Let ρ be a ring homomorphism from R into R'. Then the map $\bar{\rho} : R/\ker(\rho) \to \mathrm{im}(\rho)$ that sends the coset $a + \ker(\rho)$ for $a \in R$ to $\rho(a)$ is unambiguously defined and is a ring isomorphism of $R/\ker(\rho)$ with $\mathrm{im}(\rho)$.*

Proof. Exercise. □

Theorem 9.61 *Let ρ be a ring homomorphism from R into R'. Then for any ideal I contained in $\ker(\rho)$, the map $\bar{\rho} : R/I \to \mathrm{im}(\rho)$ that sends the coset $a + I$ for $a \in R$ to $\rho(a)$ is unambiguously defined and is a ring homomorphism from R/I onto $\mathrm{im}(\rho)$ with kernel $\ker(\rho)/I$.*

Proof. Exercise. □

Example 9.62 For $n \geq 1$, the natural map ρ from $\mathbb{Z}$ to $\mathbb{Z}_n$ sends $a \in \mathbb{Z}$ to the residue class $[a \bmod n]$. In Example 8.56 we noted that this is a surjective group homomorphism on the underlying additive groups, with kernel $n\mathbb{Z}$; however, this map is also a ring homomorphism. □

Example 9.63 As we saw in Example 8.57, if $n_1, \ldots, n_k$ are integers, all greater than 1, such that $\gcd(n_i, n_j) = 1$ for all $1 \leq i < j \leq k$, then the map from $\mathbb{Z}$ to $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$ that sends $x \in \mathbb{Z}$ to $([x \bmod n_1], \ldots, [x \bmod n_k])$ is a surjective group homomorphism on the underlying additive groups, with kernel $n\mathbb{Z}$, where $n = \prod_{i=1}^{k} n_i$. However, this map is also a ring homomorphism. Therefore, by Theorem 9.60, the map that sends $[x \bmod n] \in \mathbb{Z}_n$ to $([x \bmod n_1], \ldots, [x \bmod n_k])$ is a ring isomorphism of the ring $\mathbb{Z}_n$ with the ring $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$. It follows that the restriction of this map to $\mathbb{Z}_n^*$ yields a *group* isomorphism of the *multiplicative* groups $\mathbb{Z}_n^*$ and $\mathbb{Z}_{n_1}^* \times \cdots \times \mathbb{Z}_{n_k}^*$. □

Example 9.64 As we saw in Example 8.58, if n_1, n_2 are positive integers with $n_1 > 1$ and $n_1 \mid n_2$, then the map $\bar{\rho} : \mathbb{Z}_{n_2} \to \mathbb{Z}_{n_1}$ that sends $[a \bmod n_2]$ to $[a \bmod n_1]$ is a surjective group homomorphism on the underlying additive groups with kernel $n_1\mathbb{Z}_{n_2}$. This map is also a ring homomorphism. The map $\bar{\rho}$ can also be viewed as the map obtained from Theorem 9.61 applied to the natural map ρ from $\mathbb{Z}$ to $\mathbb{Z}_{n_1}$ and the ideal $n_2\mathbb{Z}$ of $\mathbb{Z}$, which is contained in $\ker(\rho) = n_1\mathbb{Z}$. □

Example 9.65 Let R be a subring of S, and fix $\alpha \in S$. The "polynomial evaluation map" ρ that sends $a \in R[\mathtt{X}]$ to $a(\alpha) \in S$ is a ring homomorphism from $R[\mathtt{X}]$ into S (see Theorem 9.36). The image of ρ consists of all polynomial expressions in α with coefficients in R, and is denoted $R[\alpha]$. Note that $R[\alpha]$ is a subring of S containing $R \cup \{\alpha\}$, and is the smallest such subring of S. □

Example 9.66 We can generalize the previous example to multi-variate polynomials. If R is a subring of S and $\alpha_1, \ldots, \alpha_n \in R$, then the map $\rho : R[\mathtt{X}_1, \ldots, \mathtt{X}_n] \to S$ that sends $a \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ to $a(\alpha_1, \ldots, \alpha_n)$ is a ring homomorphism. Its image consists of all polynomial expressions in $\alpha_1, \ldots, \alpha_n$ with coefficients in R, and is denoted $R[\alpha_1, \ldots, \alpha_n]$. Moreover, this image is a subring of S containing $R \cup \{\alpha_1, \ldots, \alpha_n\}$, and is the smallest such subring of S. □

Example 9.67 As in Example 9.50, let f be a monic polynomial over R with $\deg(f) = \ell > 0$, and consider the natural map ρ from $R[\mathtt{X}]$ to $S = R[\mathtt{X}]/(f)$ that sends $a \in R[\mathtt{X}]$ to $[a \bmod f]$. If we restrict ρ to the subring R of $R[\mathtt{X}]$, we obtain an embedding of R into S. Since this is a very natural embedding, one usually simply regards R as a subring of S in this case, and so the map ρ may be viewed as the polynomial evaluation map, as in the previous example, that sends $a \in R[\mathtt{X}]$ to $a(\eta) \in S$, where $\eta = [\mathtt{X} \bmod f]$. Note that we have $S = R[\eta]$; moreover, every element of S can be expressed uniquely as $a(\eta)$ for some $a \in R[\mathtt{X}]$ of degree less than ℓ, and more generally, for arbitrary $a, b \in R[\mathtt{X}]$, we have $a(\eta) = b(\eta)$ if and only if $a \equiv b \bmod f$. □

Example 9.68 If $\rho : R \to R'$ is a ring homomorphism, then we can extend ρ in a natural way to a ring homomorphism from $R[\mathtt{X}]$ to $R'[\mathtt{X}]$, by defining $\rho(\sum_i a_i \mathtt{X}^i) := \sum_i \rho(a_i)\mathtt{X}^i$. We leave it to the reader to verify that this indeed is a ring homomorphism. As concrete examples, this yields natural ring homomorphisms from $\mathbb{Z}[\mathtt{X}]$ to $\mathbb{Z}_n[\mathtt{X}]$ for any $n \geq 1$, and for positive integers $n_1 \mid n_2$, we obtain a natural ring homomorphism from $\mathbb{Z}_{n_2}[\mathtt{X}]$ to $\mathbb{Z}_{n_1}[\mathtt{X}]$. □

Example 9.69 Let $\rho : R \to R'$ be a ring homomorphism, extended to a ring homomorphism from $R[\mathtt{X}]$ to $R'[\mathtt{X}]$ as in the previous example. Let $f \in R[x]$ be a monic polynomial, and let f' denote the image of f in $R'[\mathtt{X}]$ under ρ. Then we get a natural ring homomorphism σ from $R[\mathtt{X}]$ to $R'[\mathtt{X}]/(f')$, sending $a \in R[\mathtt{X}]$ to $[\rho(a) \bmod f']$. Moreover, since $f \in \ker(\sigma)$, by Theorem 9.61, we get a natural ring homomorphism $\bar{\sigma}$ from $R[\mathtt{X}]/(f)$ to $R'[\mathtt{X}]/(f')$, sending $[a \bmod f]$ to $[\rho(a) \bmod f']$. □

Example 9.70 Let $f := \mathtt{X}^2 + 1 \in \mathbb{R}[\mathtt{X}]$, and consider the quotient ring $\mathbb{R}[\mathtt{X}]/(f)$. If we set $i := [\mathtt{X} \bmod f] \in \mathbb{R}[\mathtt{X}]/(f)$, then every element of $R[\mathtt{X}]/(f)$ can be expressed uniquely as $a + bi$, where $a, b \in \mathbb{R}$. Moreover, we have $i^2 = -1$, and more generally, for $a, b, a', b' \in \mathbb{R}$, we have

$$(a + bi) + (a' + b'i) = (a + a') + (b + b')i \quad \text{and} \quad (a + bi) \cdot (a' + b'i) = (aa' - bb') + (ab' - a'b)i.$$

Thus, the rules for arithmetic in $\mathbb{R}[\mathtt{X}]/(f)$ are precisely the familiar rules of complex arithmetic, and so $\mathbb{C}$ and $\mathbb{R}[\mathtt{X}]/(f)$ are essentially the same, as rings. Indeed, the "algebraically correct" way of defining the complex numbers $\mathbb{C}$ is simply to define them to be the quotient ring $\mathbb{R}[\mathtt{X}]/(f)$ in the first place. This will be our point of view from now on.

Consider the polynomial evaluation map $\rho : \mathbb{R}[\mathtt{X}] \to \mathbb{C}$ that sends $g \in \mathbb{R}[\mathtt{X}]$ to $g(-i)$. This is a ring homomorphism, and f is clearly in the kernel of ρ, since $(-i)^2 + 1 = 0$. By Theorem 9.61, the map $\bar{\rho}$ that sends $[g \bmod f]$ to $g(-i)$ is a well-defined ring homomorphism. Note that $\bar{\rho}(a + bi) = a - bi$, for $a, b \in \mathbb{R}$. Thus, it is clear that $\bar{\rho}$ is both injective and surjective, and indeed, it is none other than the complex conjugation map. Indeed, this is the "algebraically correct" way of defining complex conjugation in the first place. □

Example 9.71 We defined the ring $\mathbb{Z}[i]$ of Gaussian integers (see Example 9.27) as a subring of $\mathbb{C}$; however, it can also be constructed directly as $\mathbb{Z}[\mathtt{X}]/(\mathtt{X}^2+1)$. Indeed, the map $\rho : \mathbb{Z}[\mathtt{X}] \to \mathbb{C} := \mathbb{R}[\mathtt{X}]/(\mathtt{X}^2+1)$ that sends $a \in \mathbb{Z}[\mathtt{X}]$ to $a+(\mathtt{X}^2+1)\mathbb{R}[\mathtt{X}]$ is a ring homomorphism whose kernel is evidently $(\mathtt{X}^2+1)\mathbb{Z}[\mathtt{X}]$. Therefore, the image of ρ, which is clearly equal to $\mathbb{Z}[i]$, is isomorphic to $\mathbb{Z}[\mathtt{X}]/(\mathtt{X}^2+1)$.

Likewise the field $\mathbb{Q}[i]$ (see Exercise 9.32) can be constructed directly as $\mathbb{Q}[\mathtt{X}]/(\mathtt{X}^2+1)$. Such direct constructions are appealing in that they are purely "elementary," as they do not appeal to anything so "sophisticated" as the real numbers. □

Example 9.72 Consider the field F of 4 elements defined in Example 9.15. The reader may verify that F is isomorphic (as a ring) to the $\mathbb{Z}_2[\mathtt{X}]/(\mathtt{X}^2+\mathtt{X}+1)$ via the map that sends the bit pair $(a, b) \in F$ to $[a\mathtt{X} + b \bmod \mathtt{X}^2 + \mathtt{X} + 1] \in \mathbb{Z}_2[\mathtt{X}]/(\mathtt{X}^2 + \mathtt{X} + 1)$. It should also be pointed out that even though F and $\mathbb{Z}_4$ are both rings with 4 elements, they are by no means isomorphic as rings — indeed, $\mathbb{Z}_4$ is not a field. □

Example 9.73 For any ring R, consider the map $\rho : \mathbb{Z} \to R$ that sends $m \in \mathbb{Z}$ to $m \cdot 1_R$ in R. This is clearly a ring homomorphism (verify). If $\ker(\rho) = \{0\}$, then $\mathrm{im}(\rho) \cong \mathbb{Z}$, and so the ring $\mathbb{Z}$ is embedded in R, and R has characteristic zero. If $\ker(\rho) = n\mathbb{Z}$ for $n > 0$, then $\mathrm{im}(\rho) \cong \mathbb{Z}_n$, and so the ring $\mathbb{Z}_n$ is embedded in R, and R has characteristic n. Note that have $n = 1$ if and only if R is trivial.

Note that $\mathrm{im}(\rho)$ is the smallest subring of R; indeed, since any subring of R must contain 1_R and be closed under addition, it must contain $\mathrm{im}(\rho)$.

Now suppose that R is an integral domain of non-zero characteristic n. Then $n > 1$ and R contains an isomorphic copy of $\mathbb{Z}_n$. Since any subring of an integral domain must itself be an integral domain, it follows that n must be prime. We conclude: *the characteristic of an integral domain is either zero or prime.* □

Example 9.74 Let R be a ring of prime characteristic p. For any $a, b \in R$, we have (c.f., Exercise 9.9)

$$(a + b)^p = \sum_{k=0}^{p} \binom{p}{k} a^{p-k} b^k.$$

However, by Exercise 1.16, all of the binomial coefficients are multiples of p, except for $k = 0$ and $k = p$, and hence in the ring R, all of these terms vanish, leaving us with

$$(a + b)^p = a^p + b^p.$$

This result is often jokingly referred to as the "freshman's dream," for somewhat obvious reasons.

Of course, as always, we have

$$(ab)^p = a^b b^p \quad \text{and} \quad 1_R^p = 1_R,$$

and so it follows that the map $\rho : R \to R$ that sends $a \in R$ to a^p is a ring homomorphism. It also immediately follows that for any integer $e \geq 1$, the map $\rho^e : R \to R$ that sends $a \in R$ to a^{p^e} is also a ring homomorphism.

□

Example 9.75 For the more formalistically minded, one can make our construction of the ring $R[\mathtt{X}]$ of polynomials over a ring R more rigorous as follows. One defines $R[\mathtt{X}]$ to be the set of all infinite sequences $(a_0, a_1, a_2, \ldots)$ of elements of R, where only finitely many of the a_i's may be non-zero. The interpretation is that such a sequence represents the polynomial $\sum_i a_i \mathtt{X}^i$, and the rules for arithmetic are defined on these sequences so as to be consistent with this interpretation. Under this interpretation, the indeterminate $\mathtt{X}$ is simply the special sequence $(0_R, 1_R, 0_R, 0_R, \ldots)$. Also, we have a natural embedding $\rho : R \to R[\mathtt{X}]$ that sends $a \in R$ to the sequence $(a, 0_R, 0_R, \ldots)$. Thus, strictly speaking, R is not a subring of $R[\mathtt{X}]$, but rather, is embedded in $R[\mathtt{X}]$ via the map ρ. □

Exercise 9.76 Let ρ be a ring homomorphism from R into R'. Show that the ideals of R containing $\ker(\rho)$ are in one-to-one correspondence with the ideals of $\mathrm{im}(\rho)$, where the the ideal I in R containing $\ker(\rho)$ corresponds to the ideal $\rho(I)$ in $\mathrm{im}(\rho)$. □

Exercise 9.77 Show that if F is a field, then the only ideals in F are $\{0_F\}$ and F. From this, conclude the following: if $\rho : F \to R$ is a ring homomorphism from F into a non-trivial ring R, then ρ must be an embedding. □

Exercise 9.78 Suppose I and J are two ideals in a ring R such that $I + J = R$. Show that the map $\rho : R \to R/I \times R/J$ that sends $a \in R$ to $([a \bmod I], [a \bmod J])$ is a surjective ring homomorphism with kernel $I \cdot J$. Conclude that $R/(I \cdot J)$ is isomorphic to $R/I \times R/J$. □

Exercise 9.79 Let F be a field and let d be an element of F that is not a perfect square (i.e., there does not exist $e \in F$ such that $e^2 = d$). Let $E := F[\mathtt{X}]/(\mathtt{X}^2 - d)$, and let $\eta := [\mathtt{X} \bmod (\mathtt{X}^2 - d)]$, so that $E = F[\eta] = \{a + b\eta : a, b \in F\}$.

(a) Show that the quotient ring E is a field, and write down the formula for the inverse of $a + b\eta \in E$.

(b) Show that the map that sends $a + b\eta \in E$ to $a - b\eta$ is a ring automorphism on E.

□

Chapter 10

Probabilistic Primality Testing

In this chapter, we discuss some simple and efficient probabilistic tests for primality.

10.1 Trial Division

Suppose we are given a number n, and we want to determine whether n is prime or composite. The simplest algorithm to describe and to program is **trial division**. We simply divide n by 2, 3, and so on, testing if any of these numbers evenly divide n. Of course, we don't need to go any farther than $\sqrt{n}$, since if n has any nontrivial factors, it must have one that is no greater than $\sqrt{n}$. Not only does this algorithm determine whether n is prime or composite, it also produces the complete prime factorization of n.

Of course, the drawback of this algorithm is that it is terribly inefficient: it requires $O(\sqrt{n})$ arithmetic operations, which is exponential in the binary length of n. Thus, for practical purposes, this algorithm is limited to quite small n. Suppose, for example, that n has 100 decimal digits, and that a computer can perform 1 billion divisions per second (this is much faster than any computer existing today). Then it would take 3×10^{35} *years* to perform $\sqrt{n}$ divisions.

In this chapter, we discuss a much faster primality test that allows 100 decimal digit numbers to be tested for primality less than a second. Unlike the above test, however, this test does not find a factor of n when n is composite. Moreover, the algorithm is probabilistic, and may in fact make a mistake. However, the probability that it makes a mistake can be made so small as to be irrelevant for all practical purposes. Indeed, we can easily make the probability of error as small as 2^{-100} — should one really care about an event that happens with such a miniscule probability?

10.2 The Structure of $\mathbb{Z}_n^*$

Before going any further, we have to have a firm understanding of the group $\mathbb{Z}_n^*$. As we know, $\mathbb{Z}_n^*$ consists of those elements $[a \bmod n] \in \mathbb{Z}_n$ such that a is an integer relatively

prime to n. Suppose $n = p_1^{e_1} \cdots p_r^{e_r}$ is the factorization of n into primes. By the Chinese Remainder Theorem, we have the ring isomorphism

$$\mathbb{Z}_n \cong \mathbb{Z}_{p_1^{e_1}} \times \cdots \times \mathbb{Z}_{p_r^{e_r}}$$

which induces a group isomorphism

$$\mathbb{Z}_n^* \cong \mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_r^{e_r}}^*.$$

Thus, to determine the structure of the group $\mathbb{Z}_n^*$ for general n, it suffices to determine the structure for $n = p^e$, where p is prime. By Theorem 2.14, we already know the order of the group $\mathbb{Z}_{p^e}^*$, namely, $\phi(p^e) = p^{e-1}(p-1)$.

The main result of this section is the following:

Theorem 10.1 *If p is an odd prime, then for any positive integer e, the group $\mathbb{Z}_{p^e}^*$ is cyclic. The group $\mathbb{Z}_{2^e}^*$ is cyclic for $e = 1$ or 2, but not for $e \geq 3$. For $e \geq 3$, $\mathbb{Z}_{2^e}^*$ is isomorphic to the group $\mathbb{Z}_2 \times \mathbb{Z}_{2^{e-2}}$.*

In the case where $e = 1$, this theorem is a special case of the following theorem:

Theorem 10.2 *Let F be a field and G a subgroup of F^* of finite order. Then G is cyclic.*

Proof. Let n be the order of G, and suppose G is not cyclic. Then by Theorem 8.83, we have that the exponent m of G is strictly less than n. It follows that $\alpha^m = 1_F$ for all $\alpha \in G$. That is, all the elements of G are roots of the polynomial $\mathtt{X}^m - 1_F \in F[\mathtt{X}]$. But since a polynomial of degree m over a field has at most m roots, this contradicts the fact that $m < n$. □

To deal with the case $e > 1$, we need a few simple facts.

Theorem 10.3 *For $e \geq 1$, if $a \equiv b \pmod{p^e}$, then $a^p \equiv b^p \pmod{p^{e+1}}$.*

Proof. We have $a = b + cp^e$ for some $c \in \mathbb{Z}$. Thus, $a^p = b^p + pb^{p-1}cp^e + dp^{2e}$ for an integer d. It follows that $a^p \equiv b^p \pmod{p^{e+1}}$. □

Theorem 10.4 *Let $e \geq 1$ and assume $p^e > 2$. If $a \equiv 1 + p^e \pmod{p^{e+1}}$, then $a^p \equiv 1 + p^{e+1} \pmod{p^{e+2}}$.*

Proof. By Theorem 10.3, $a^p \equiv (1 + p^e)^p \pmod{p^{e+2}}$. Expanding $(1 + p^e)^p$, we have

$$(1 + p^e)^p = 1 + p \cdot p^e + \sum_{k=2}^{p-1} \binom{p}{k} p^{ek} + p^{ep}.$$

By Exercise 1.16, all of the terms in the sum on k are divisible by p^{1+2e}, and $1 + 2e \geq e + 2$ for all $e \geq 1$. For the term p^{ep}, the assumption that $p^e > 2$ means that either $p \geq 3$ or $e \geq 2$, which implies $ep \geq e + 2$. □

Now consider Theorem 10.1 in the case where p is odd. We have already proven that $\mathbb{Z}_p^*$ is cyclic, so we may assume $e > 1$. Let $x \in \mathbb{Z}$ be chosen so that $[x \bmod p]$ generates $\mathbb{Z}_p^*$. Suppose the order of $[x \bmod p^e] \in \mathbb{Z}_{p^e}^*$ is m. Then as $x^m \equiv 1 \pmod{p^e}$ implies $x^m \equiv 1 \pmod{p}$, it must be the case that $p-1$ divides m, and so $[x^{m/(p-1)} \bmod p^e]$ has order exactly $p-1$. By Theorem 8.80, if we find an integer y such that $[y \bmod p^e]$ has order p^{e-1}, then $[x^{m/(p-1)}y \bmod p^e]$ has order $(p-1)p^{e-1}$, and we are done. We claim that $y = 1 + p$ does the job. Any integer between 0 and $p^e - 1$ can be expressed as an e-digit number in base p; for example, $y = (0 \cdots 0\,1\,1)_p$. If we compute successive p-th powers of y modulo p^e, then by Theorem 10.4 we have:

$$\begin{aligned}
y \operatorname{rem} p^e &= (0 \cdots \quad 0\,1\,1)_p \\
y^p \operatorname{rem} p^e &= (* \cdots \quad *\,1\,0\,1)_p \\
y^{p^2} \operatorname{rem} p^e &= (* \cdots \; *\,1\,0\,0\,1)_p \\
&\vdots \\
y^{p^{e-2}} \operatorname{rem} p^e &= (1\,0 \cdots \quad 0\,1)_p \\
y^{p^{e-1}} \operatorname{rem} p^e &= (0 \cdots \quad 0\,1)_p
\end{aligned}$$

Here, "$*$" indicates an arbitrary digit. From this table of values, it is clear (c.f., Theorem 8.79) that $[y \bmod p^e]$ has order p^{e-1}. That proves Theorem 10.1 for odd p.

We now prove Theorem 10.1 in the case $p = 2$. For $e = 1$ and $e = 2$, the theorem is clear. Suppose $e \geq 3$. Consider the subgroup $G \subset \mathbb{Z}_{2^e}^*$ generated by $[5 \bmod 2^e]$. Expressing integers between 0 and $2^e - 1$ as e-digit binary numbers, and applying Theorem 10.4, we have:

$$\begin{aligned}
5 \operatorname{rem} 2^e &= (0 \cdots \quad 0\,1\,0\,1)_2 \\
5^2 \operatorname{rem} 2^e &= (* \cdots \; *\,1\,0\,0\,1)_2 \\
&\vdots \\
5^{2^{e-3}} \operatorname{rem} 2^e &= (1\,0 \cdots \quad 0\,1)_2 \\
5^{2^{e-2}} \operatorname{rem} 2^e &= (0 \cdots \quad 0\,1)_2
\end{aligned}$$

So it is clear (c.f., Theorem 8.79) that $[5 \bmod 2^e]$ has order 2^{e-2}. We claim that $[-1 \bmod 2^e] \notin G$. If it were, then since it has order 2, and since any cyclic group of even order has precisely one element of order 2 (c.f., Theorem 8.75), it must be equal to $[5^{2^{e-3}} \bmod 2^e]$; however, it is clear from the above calculation that $5^{2^{e-3}} \not\equiv -1 \pmod{2^e}$. Let $H \subset \mathbb{Z}_{2^e}^*$ be the subgroup generated by $[-1 \bmod 2^e]$. Then from the above, $G \cap H = \{[1 \bmod 2^e]\}$, and hence by Theorem 8.55, $G \times H$ is isomorphic to the subgroup $G \cdot H$ of $\mathbb{Z}_{2^e}^*$. But since the orders of $G \times H$ and $\mathbb{Z}_{2^e}^*$ are equal, we must have $G \cdot H = \mathbb{Z}_{2^e}^*$. That proves the theorem.

Exercise 10.5 This exercise develops an alternative proof of Theorem 10.2. Let n be the order of the group. Using Theorem 9.41, show that for all $d \mid n$, there are at most d elements in the group whose order divides d. From this, deduce that for all $d \mid n$, the number of elements of order d is either 0 or $\phi(d)$. Now use Theorem 2.12 to deduce that for all $d \mid n$ (and in particular, for $d = n$), the number of elements of order d is equal to $\phi(d)$. □

Exercise 10.6 Let $n = pq$, where p and q are distinct primes such that $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are themselves prime. Show that $\mathbb{Z}_n^*$ is not a cyclic group, while the subgroup $(\mathbb{Z}_n^*)^2$ of squares is a cyclic group of order $p'q'$. □

Exercise 10.7 Let $n = pq$, where p and q are distinct primes such that $p \nmid (q-1)$ and $q \nmid (p-1)$. Show that the map that sends $[a \bmod n] \in \mathbb{Z}_n^*$ to $[a^n \bmod n^2] \in (\mathbb{Z}_{n^2}^*)^n$ is a group isomorphism. Consider the element $\alpha = [1 + n \bmod n^2] \in \mathbb{Z}_{n^2}^*$; show that for any non-negative integer k, $\alpha^k = [1 + kn \bmod n^2]$, and conclude that α has order n. Show that the map from $\mathbb{Z}_n \times \mathbb{Z}_n^*$ to $\mathbb{Z}_{n^2}^*$ that sends $([k \bmod n], [a \bmod n])$ to $[(1 + kn)a^n \bmod n^2]$ is a group isomorphism. □

10.3 The Miller-Rabin Test

We describe in this section a fast (polynomial time) test for primality, known as the **Miller-Rabin algorithm**. The algorithm, however, is probabilistic, and may (with small probability) make a mistake.

We assume for the remainder of this section that the number n we are testing for primality is odd.

Several probabilistic primality tests, including the the Miller-Rabin algorithm, have the following general structure. Define $\mathbb{Z}_n^{\neq}$ to be the set of non-zero elements of $\mathbb{Z}_n$; thus, $|\mathbb{Z}_n^{\neq}| = n - 1$ and if n is prime, $\mathbb{Z}_n^{\neq} = \mathbb{Z}_n^*$. Suppose also that we define a set $L_n \subset \mathbb{Z}_n^{\neq}$ such that

- there is an efficient algorithm that on input n and $\alpha \in \mathbb{Z}_n^{\neq}$, determines if $\alpha \in L_n$;
- if n is prime, then $L_n = \mathbb{Z}_n^*$;
- if n is composite, $|L_n| \leq c(n-1)$ for some constant $c < 1$.

To test n for primality, we set an "error parameter" t, and choose random elements $\alpha_1, \ldots, \alpha_t \in \mathbb{Z}_n^{\neq}$. If $\alpha_i \in L_n$ for all $1 \leq i \leq t$, then we output *true*; otherwise, we output *false*.

It is easy to see that if n is prime, this algorithm always outputs *true*, and if n is composite this algorithm outputs *true* with probability at at most c^t. If $c = 1/2$ and t is chosen large enough, say $t = 100$, then the probability that the output is wrong is so small that for all practical purposes, it is "just as good as zero."

We now make a first attempt at defining a suitable set L_n. Let us define $L_n = \{\alpha \in \mathbb{Z}_n^{\neq} : \alpha^{n-1} = 1\}$. Note that $L_n \subset \mathbb{Z}_n^*$, since if $\alpha^{n-1} = 1$, then α has a multiplicative inverse, namely, α^{n-2}. Using a repeated-squaring algorithm, we can test if $\alpha \in L_n$ in time $O(\text{len}(n)^3)$.

Theorem 10.8 *If n is prime, then $L_n = \mathbb{Z}_n^*$. If n is composite and $L_n \subsetneq \mathbb{Z}_n^*$, then $|L_n| \leq (n-1)/2$.*

Proof. Note that L_n is the kernel of the $(n-1)$-power map on $\mathbb{Z}_n^*$, and hence is a subgroup of $\mathbb{Z}_n^*$.

If n is prime, then we know that $\mathbb{Z}_n^*$ is a group of order $n-1$. Hence, $\alpha^{n-1} = 1$ for all $\alpha \in \mathbb{Z}_n^*$. That is, $L_n = \mathbb{Z}_n^*$.

Suppose that n is composite and $L_n \subsetneq \mathbb{Z}_n^*$. Since the order of a subgroup divides the order of the group, we have $|\mathbb{Z}_n^*| = m|L_n|$ for some integer $m > 1$. From this, we conclude that

$$|L_n| = \frac{1}{m}|\mathbb{Z}_n^*| \leq \frac{1}{2}|\mathbb{Z}_n^*| \leq \frac{n-1}{2}.$$

□

Unfortunately, there are odd composite numbers n such that $L_n = \mathbb{Z}_n^*$. The smallest such number is

$$561 = 3 \cdot 11 \cdot 17.$$

Such numbers are called **Carmichael numbers**. They are extremely rare, but it is known that there are infinitely many of them, so we can not ignore them. The following theorem puts some constraints on such numbers.

Theorem 10.9 *A Carmichael number n is of the form $n = p_1 \cdots p_r$, where the p_i's are distinct primes, $r \geq 3$, and $(p_i - 1) \mid (n-1)$ for $1 \leq i \leq r$.*

Proof. Let $n = p_1^{e_1} \cdots p_r^{e_r}$ be a Carmichael number. By the Chinese Remainder Theorem, we have an isomorphism of $\mathbb{Z}_n^*$ with the group

$$\mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_r^{e_r}}^*,$$

and we know that each group $\mathbb{Z}_{p_i^{e_i}}^*$ is cyclic of order $p_i^{e_i-1}(p_i - 1)$. Thus, the power $n-1$ kills the group $\mathbb{Z}_n^*$ if and only if it kills all the groups $\mathbb{Z}_{p_i^{e_i}}^*$, which happens if and only if $p_i^{e_i-1}(p_i-1) \mid (n-1)$. Now, on the one hand, $n \equiv 0 \pmod{p_i}$. On the other hand, if $e_i > 1$, we would have $n \equiv 1 \pmod{p_i}$, which is clearly impossible. Thus, we must have $e_i = 1$.

It remains to show that $r \geq 3$. Suppose $r = 2$, so that $n = p_1 p_2$. We have

$$n - 1 = p_1 p_2 - 1 = (p_1 - 1)p_2 + (p_2 - 1).$$

Since $(p_1 - 1) \mid (n-1)$, we must have $(p_1 - 1) \mid (p_2 - 1)$. By a symmetric argument, $(p_2 - 1) \mid (p_1 - 1)$. Hence, $p_1 = p_2$, a contradiction. □

To obtain a good primality test, we need to define a different set L'_n, which we do as follows. Let $n - 1 = 2^h m$, where m is odd (and $h \geq 1$ since n is assumed odd). To determine if a given $\alpha \in \mathbb{Z}_n^{\neq}$ is in L'_n, we consider the following sequence:

$$\alpha^{m2^j} \quad (j = 0, \ldots, h).$$

Membership of α in L'_n is determined by the following rules:

1. If $\alpha^{m2^h} \neq 1$, then α *is not* in L'_n;
2. otherwise, if $\alpha^{m2^j} = 1$ for all $0 \leq j \leq h$, then α *is* in L'_n;
3. otherwise, consider the greatest index j such that $\alpha^{m2^j} \neq 1$; if $\alpha^{m2^j} = -1$, then α *is* in L'_n;
4. otherwise, α *is not* in L'_n.

The Miller-Rabin algorithm uses this set L'_n, in place of the set L_n defined above. It is clear that membership in L'_n can be determined in time $O(\text{len}(n)^3)$ using a repeated-squaring algorithm.

Note that L'_n is a subset of L_n: if $\alpha^m = 1$, then certainly $\alpha^{n-1} = (\alpha^m)^{2^h} = 1$, and if $\alpha^{m2^j} = -1$ for some $0 \leq j < h$, then $\alpha^{n-1} = (\alpha^{m2^j})^{2^{h-j}} = 1$.

Theorem 10.10 *If n is prime, then $L'_n = \mathbb{Z}_n^*$. If n is composite, then $|L'_n| \leq (n-1)/4$.*

The rest of this section is devote to a proof of this theorem.

Let $n - 1 = m2^h$, where m is odd.

First, suppose n is prime. By Fermat's Little Theorem, for $\alpha \in \mathbb{Z}_n^*$, we know that $\alpha^{m2^h} = \alpha^{n-1} = 1$. Moreover, for $\beta := \alpha^{m2^j}$, if $\beta^2 = \alpha^{m2^{j+1}} = 1$, then the only possible choices for β are ± 1 — this is because $\mathbb{Z}_n^*$ is cyclic of even order and so there are exactly 2 elements whose order divides 2, namely $[\pm 1 \bmod n]$. From this, it follows from the definition that $\alpha \in L'_n$.

Now suppose that n is an odd composite.

Our strategy will be to first show that L'_n is contained in a particular subgroup G of $\mathbb{Z}_n^*$. We will then show that the order of G is suitably small.

Let

$$n = p_1^{e_1} \cdots p_r^{e_r}$$

be the prime factorization of n. Further, let

$$\rho : \mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_r^{e_r}}^* \to \mathbb{Z}_n^*$$

be the isomorphism provided by the Chinese Remainder Theorem. Also, let $\phi(p_i^{e_i}) = m_i 2^{h_i}$, with m_i odd, for $1 \leq i \leq r$, and let $\ell := \min\{h, h_1, \ldots, h_r\}$.

Let $\alpha \in L'_n$ be given. We have already argued that $\alpha \in \mathbb{Z}_n^*$, so let $\alpha = \rho(\alpha_1, \ldots, \alpha_r)$.

Claim 1: We have

$$\alpha^{m2^\ell} = 1.$$

Proof of claim. The claim may be restated as

$$\alpha^{m2^j} = 1 \quad (j = \ell, \ldots, h).$$

The claim is clearly true by the definition of L'_n for $j = h$. If $\ell = h$, there is nothing more to prove, so assume that $\ell < h$, and in particular, that $\ell = h_i$ for some $1 \leq i \leq r$. We may

then prove the claim by induction, assuming that it is true for some j, with $\ell + 1 \leq j \leq h$, and proving it for $j-1$. Since $\alpha \in L'_n$ and $\alpha^{m2^j} = 1$, we must have $\alpha^{m2^{j-1}} = \pm 1$. Suppose, by way of contradiction, that $\alpha^{m2^{j-1}} = -1$. Since $-1 = \rho(-1, \ldots, -1)$, we must have $\alpha_i^{m2^{j-1}} = -1$. However, observe that since $\alpha_i^{m2^j} = 1$, the order of α_i must divide $m2^j$, and since this order also divides $m_i 2^{h_i}$, it must also divide $m2^{h_i}$, and since $j - 1 \geq \ell = h_i$, this order must divide $m2^{j-1}$. Hence, $\alpha_i^{m2^{j-1}} = 1$, a contradiction. That proves the claim.

So we have shown that

$$L'_n \subset G := \{\alpha \in \mathbb{Z}_n^* : \alpha^{m2^{\ell-1}} = \pm 1\} \subset \mathbb{Z}_n^*\{m2^\ell\} \subset \mathbb{Z}_n^*\{m2^h\} \subset \mathbb{Z}_n^*,$$

where G is clearly a subgroup of $\mathbb{Z}_n^*$, as it is the pre-image of the subgroup $\{\pm 1\}$ of $\mathbb{Z}_n^*$ under the $(m2^{\ell-1})$-power map. Recall that for any group H, written multiplicatively, and any integer k, $H\{k\}$ denotes the kernel of the k-power map on H.

Claim 2: We have

$$[\mathbb{Z}_n^*\{m2^\ell\} : G] = 2^{r-1}. \tag{10.1}$$

Proof of claim. First, consider any fixed index $1 \leq i \leq r$. We have

$$|\mathbb{Z}^*_{p_i^{e_i}}\{m2^\ell\}| = \gcd(m2^\ell, m_i 2^{h_i}) \quad \text{and} \quad |\mathbb{Z}^*_{p_i^{e_i}}\{m2^{\ell-1}\}| = \gcd(m2^{\ell-1}, m_i 2^{h_i}) = |\mathbb{Z}^*_{p_i^{e_i}}\{m2^\ell\}|/2,$$

since $\ell \leq h_i$. Also, again since $\ell \leq h_i$, $\mathbb{Z}^*_{p_i^{e_i}}$ contains an element β_i of order 2^ℓ, so that $\beta_i^{m2^{\ell-1}}$ has order 2, and hence

$$\beta_i^{m2^{\ell-1}} = -1.$$

From the above observations and the Chinese Remainder Theorem, we see that

$$[\mathbb{Z}_n^*\{m2^\ell\} : \mathbb{Z}_n^*\{m2^{\ell-1}\}] = 2^r. \tag{10.2}$$

Also from the above, we see that $\beta := \rho(\beta_1, \ldots, \beta_r) \in \mathbb{Z}_n^*$ is a pre-image of -1 under the $(m2^{\ell-1})$-power map. So we see that the $(m2^{\ell-1})$-power map maps G *onto* the subgroup $\{\pm 1\}$ in $\mathbb{Z}_n^*$, from which it follows (see Theorem 8.53) that $G/(\mathbb{Z}_n^*\{m2^{\ell-1}\}) \cong \{\pm 1\}$; in particular,

$$[G : \mathbb{Z}_n^*\{m2^{\ell-1}\}] = 2. \tag{10.3}$$

So we have a tower of subgroups

$$\mathbb{Z}_n^*\{m2^{\ell-1}\} \subset G \subset \mathbb{Z}_n^*\{m2^\ell\},$$

and (10.1) follows from (10.2) and (10.3). That proves the claim.

Now we are almost done with the proof of the theorem. There are four cases to consider. In the first three cases, we show that $[\mathbb{Z}_n^* : G] \geq 4$, from which it follows that $|L'_n|/|\mathbb{Z}_n^{\neq}| \leq 1/4$.

Case 1: $r \geq 3$. In this case, we have

$$[\mathbb{Z}_n^* : G] = [\mathbb{Z}_n^* : \mathbb{Z}_n^*\{m2^\ell\}]\,[\mathbb{Z}_n^*\{m2^\ell\} : G] \geq 1 \cdot 2^{r-1} \geq 4.$$

Case 2: $r = 2$. In this case, we know by Theorem 10.9 that n is not a Carmichael number, and hence $[\mathbb{Z}_n^* : \mathbb{Z}_n^*\{m2^h\}] \geq 2$. Hence

$$[\mathbb{Z}_n^* : G] = [\mathbb{Z}_n^* : \mathbb{Z}_n^*\{m2^h\}]\,[\mathbb{Z}_n^*\{m2^h\} : \mathbb{Z}_n^*\{m2^\ell\}]\,[\mathbb{Z}_n^*\{m2^\ell\} : G] \geq 2 \cdot 1 \cdot 2 = 4.$$

Case 3: $r = 1$ *and* $n \neq 9$. In this case, we have $n = p^e$ with $e > 1$, and $\mathbb{Z}_n^*\{n-1\} = \gcd(p^e - 1, p^{e-1}(p-1)) = p - 1$. Hence, $[\mathbb{Z}_n^* : \mathbb{Z}_n^*\{m2^h\}] = p^{e-1}$, and so

$$[\mathbb{Z}_n^* : G] = [\mathbb{Z}_n^* : \mathbb{Z}_n^*\{m2^h\}]\,[\mathbb{Z}_n^*\{m2^h\} : G] \geq p^{e-1} \cdot 1 = p^{e-1} \geq 5,$$

since we are assuming that either $p > 3$ or $e > 2$.

Case 4: $n = 9$. In this case, one can check that $L'_9 = \{\pm 1\}$, and so $|L'_9|/|\mathbb{Z}_9^{\neq}| = 2/8 = 1/4$.

That completes the proof of Theorem 10.10

Exercise 10.11 Show that an integer $n > 1$ is prime if and only if there exists an element in $\mathbb{Z}_n^*$ of order $n - 1$. □

Exercise 10.12 Let p be a prime. Show that $n := 2p + 1$ is a prime if and only if $2^{n-1} \equiv 1 \pmod{n}$. □

Exercise 10.13 Here is another primality test that takes as input a positive, odd integer n, and a positive integer parameter t. The algorithm chooses $\alpha_1, \ldots, \alpha_t \in \mathbb{Z}_n^{\neq}$ at random, and computes

$$\beta_i := \alpha_i^{(n-1)/2} \quad (i = 1, \ldots, t).$$

If $(\beta_1, \ldots, \beta_t)$ is of the form $([\pm 1], [\pm 1], \ldots, [\pm 1])$, and is not equal to $([u], [u], \ldots, [u])$ for $u \in \{\pm 1\}$, the algorithm outputs *true*; otherwise, the algorithm outputs *false*. Show that if n is prime, then the algorithm outputs *false* with probability at most 2^{-t}, and if n is composite, the algorithm outputs *true* with probability at most 2^{-t}. □

In the terminology of §7.2, the algorithm in the above exercise is an example of an "Atlantic City" algorithm for the language of prime numbers (or equivalently, the language of composite numbers), while the Miller-Rabin algorithm is an example of a "Monte Carlo" algorithm for the language of *composite* numbers.

10.4 Generating Random Primes using the Miller-Rabin Test

The Miller-Rabin test is the most practical test known for testing primality, and because of this, it is widely used in many applications, especially cryptographic applications where one needs to generate large, random primes. In this section, we discuss how one uses in Miller-Rabin test in several practically relevant scenarios where one must generate large primes.

10.4.1 Generating a random prime between 1 and M

Suppose one is given an integer $M \geq 2$, and wants to generate a random prime between 1 and M. We can do this by simply picking numbers at random until one of them passes a primality test. We discussed this problem in some detail in §7.5, where we assumed that we had a primality test *IsPrime*. The reader should review §7.5, and §7.5.1 in particular. In this section, we discuss aspects of this problem that are specific to the situation where the Miller-Rabin test us used to implement *IsPrime*.

To be more precise, let us define the following algorithm $MR(n, t)$, which takes as input positive integers n and t, and runs as follows:

```
if n = 1 then return false
if n = 2 then return true
if n is even then return false

repeat t times
    α ←_R {1, ..., n − 1}
    if α ∉ L'_n return false

return true
```

So we shall implement $IsPrime(\cdot)$ as $MR(\cdot, t)$, where t is an auxiliary parameter. By Theorem 10.10, if n is prime, the output of $MR(n, t)$ is always *true*, while if n is composite, the output is *true* with probability at most 4^{-t}. Thus, this implementation of *IsPrime* satisfies the assumptions in §7.5.1, with $\epsilon = 4^{-t}$,

Let $\gamma(M, t)$ be the probability that the output of algorithm RP in §7.5 — using this implementation of *IsPrime* — is composite. Then as we discussed in §7.5.1,

$$\gamma(M, t) \leq 4^{-t} \frac{M}{\pi(M)} = O(4^{-t}k), \tag{10.4}$$

where $k = \mathrm{len}(M)$. Furthermore, if the output of algorithm RP is prime, then every prime is equally likely; i.e., conditioning on the event that the output is prime, the conditional output distribution is uniform over all primes.

Let us now consider the expected running time of algorithm RP. As was shown in §7.5.1, this is $O(kW'_M)$, where W'_M is the expected running time of *IsPrime* where the average is taken with respect to the random choice of input $n \in \{1, \ldots, M\}$ and the random choices of the primality test itself. Clearly, we have $W'_M = O(tk^3)$, since MR executes at most t iterations of the Miller-Rabin test, and each such test takes time $O(k^3)$. This leads to a expected total running time bound of $O(tk^4)$. However, this estimate for W'_M is overly pessimistic. Intuitively, this is because when n is composite, we expect to perform very few Miller-Rabin tests — only when n is prime do we actually perform all t of them. To make a rigorous argument, consider the experiment in which n is chosen at random from $\{1, \ldots, M\}$, and $MR(n, t)$ is executed. Let Y be the number of times the basic Miller-Rabin

test is actually executed. Conditioned on any fixed, prime value of n, the value of Y is always t. Conditioned on any fixed, composite value of n, the distribution of Y is geometric with an associated success probability of at least $3/4$; thus, the conditional expectation of Y is at most $4/3$ in this case. Thus, we have

$$\mathsf{E}[Y] = \mathsf{E}[Y \mid n \text{ prime}]\mathsf{P}[n \text{ prime}] + \mathsf{E}[Y \mid n \text{ not prime}]\mathsf{P}[n \text{ not prime}] \leq t\pi(M)/M + 4/3.$$

Thus, $\mathsf{E}[Y] \leq 4/3 + O(t/k)$, from which it follows that $W'_M = O(k^3 + tk^2)$, and hence the expected total running time of algorithm RP is actually $O(k^4 + tk^3)$.

Note that the above estimate (10.4) for $\gamma(M, t)$ is actually quite pessimistic. This is because the error probability 4^{-t} is a worst-case estimate; in fact, for "most" composite integers n, the probability that $MR(n, t)$ outputs *true* is much smaller than this. In fact, $\gamma(M, 1)$ is *very* small for large M. For example, the following is known:

Theorem 10.14 *We have*

$$\gamma(M, 1) \leq \exp(-(1 + o(1)) \log(M) \log(\log(\log(M)))/ \log(\log(M))).$$

Proof. Literature — see §10.8. □

The bound in the above theorem goes to zero quite quickly — faster than $(\log M)^{-c}$ for any positive constant c. While the above theorem is asymptotically very good, in practice, one needs explicit bounds. For example, the following *lower* bounds for $-\log_2(\gamma(2^k, 1))$ are known:

k	200	300	400	500	600
	3	19	37	55	74

Given an upper bound on $\gamma(M, 1)$, we can bound $\gamma(M, t)$ for $t \geq 2$ using the following inequality:

$$\gamma(M, t) \leq \frac{\gamma(M, 1)}{1 - \gamma(M, 1)} 4^{-t+1}. \tag{10.5}$$

To prove (10.5), it is not hard to see that on input M, the output distribution of algorithm RP is the same as that of the following algorithm:

```
repeat
    repeat
        n ←_R {1, ..., M}
    until MR(n, 1)
    n_1 ← n
until MR(n_1, t − 1)
output n_1
```

Consider for a moment a single execution of the outer loop of the above algorithm. Let β be the probability that n_1 is composite, and let α be the conditional probability that $MR(n_1, t-1)$ outputs *true*, given that n_1 is composite. Evidently, $\beta = \gamma(M, 1)$ and $\alpha \leq 4^{-t+1}$.

Now, using *exactly* the same reasoning as was used to derive equation (7.2) in §7.5.1, we find that

$$\gamma(M,t) = \frac{\alpha\beta}{\alpha\beta + (1-\beta)} \leq \frac{\alpha\beta}{1-\beta} \leq \frac{4^{-t+1}\gamma(M,1)}{1-\gamma(M,1)},$$

which proves (10.5).

Given that $\gamma(M, 1)$ is so small, for large M, algorithm RP actually exhibits the following behavior in practice: it generates a random value $n \in \{1, \ldots, M\}$; if n is odd and composite, then the very *first* iteration of the Miller-Rabin test will detect this with overwhelming probability, and no more iterations of the test are performed on this n; otherwise, if n is prime, the algorithm will perform $t-1$ more iterations of the Miller-Rabin test, "just to make sure."

Exercise 10.15 Consider the problem of generating a random Sophie Germain prime between 1 and M (see §5.5.4). One algorithm to do this is as follows:

```
repeat
    n ←_R {1, ..., M}
    if MR(n, t) then
        if MR(2n + 1, t) then
            output n and halt
forever
```

Assuming Conjecture 5.44, show that this algorithm runs in expected time $O(k^5 + tk^4)$, and outputs a number that is not a Sophie Germain prime with probability $O(4^{-t}k^2)$. As usual, $k := \mathrm{len}(M)$. □

Exercise 10.16 Improve the algorithm in the previous exercise, so that under the same assumptions, it runs in expected time $O(k^5 + tk^3)$, and outputs a number that is not a Sophie Germain prime with probability $O(4^{-t}k^2)$, or even better, show that this probability is at most $\gamma(M, t)\pi^*(M)/\pi(M) = O(\gamma(M, t)k)$, where $\pi^*(M)$ is defined as in §5.5.4. □

Exercise 10.17 Suppose in algorithm RFN in §7.7 we implement algorithm *IsPrime*$(\cdot)$ as $MR(\cdot, t)$, where t is a parameter satisfying $4^{-t}(2 + \log M) \leq 1/2$, if M is the input to RFN. Show that the expected running time of algorithm RFN in this case is $O(k^5 + tk^4\,\mathrm{len}(k))$. Hint: use Exercise 7.15. □

10.4.2 Sieving up to a small bound

In generating a random prime, most candidates n will in fact be composite, and so it makes sense to cast these out as quickly as possible. Significant efficiency gains can be achieved

by testing if a given integer n is divisible by any small primes up to a given bound s, before we subject n to a Miller-Rabin test. This strategy makes sense, since for a small, "single precision" prime p, we can test if $p \mid n$ in time $O(\text{len}(n))$, while a single iteration of the Miller-Rabin test takes time $O(\text{len}(n)^3)$ steps.

To be more precise, let us define the following algorithm $MRS(n, t, s)$, which takes as input positive integers n, t, and s, where $s \geq 2$, and runs as follows:

```
if n = 1 then return false
for each prime p ≤ s do
    if p | n then
        if p = n then return true else return false

repeat t times
    α ←_R {1, ..., n − 1}
    if α ∉ L′_n return false

return true
```

In an implementation of the above algorithm, one would most likely use the Sieve of Eratosthenes (see §5.4) to generate the small primes.

Note that $MRS(\cdot, \cdot, 2)$ is equivalent to $MR(\cdot, \cdot)$. Also, it is clear that the probability that MRS makes a mistake on a given n is no more than the probability that MR makes a mistake. Therefore, using MRS in place of MR will not increase the probability that the output of algorithm RP is a composite — indeed, it is likely that this probability decreases significantly.

Let us now analyze the impact on the running time. To do this, we need to estimate the probability $\tau(M, s)$ that a randomly chosen number between 1 and M is not divisible by any primes up to s. If M is sufficiently large with respect to s, the following heuristic argument can be made rigorous, as we will discuss below. The probability that a random number is divisible by a prime p is about $1/p$, so the probability that it is not divisible by p is about $1 - 1/p$. Assuming that these events are essentially independent for different values of p (this is the heuristic part), we estimate

$$\tau(M, s) \approx \prod_{p \leq s} (1 - 1/p) \sim B_1 / \log s,$$

where $B_1 \approx 0.56146$ is the constant from Exercise 5.27 (see also Theorem 5.35).

Of course, performing the trial division takes some time, so let us also estimate the expected number $\kappa(M, s)$ of trial divisions performed. If $p_1, p_2, \ldots, p_r$ are the primes up to s, then for $1 \leq i \leq r$, the probability that we perform at least i trial divisions is precisely $\tau(M, p_i - 1)$. From this, it follows that

$$\kappa(M, s) = \sum_{p \leq s} \tau(M, p - 1) \approx \sum_{p \leq s} B_1 / \log s.$$

Using Exercise 5.22 and the Prime Number Theorem, we obtain

$$\kappa(M, s) \approx \sum_{p \leq s} B_1 / \log s \sim B_1 \pi(s) / \log s \sim B_1 s / (\log s)^2.$$

If $k = \mathrm{len}(M)$, the expected amount of time spent within *MRS* performing the Miller-Rabin test is now $\Theta(k^3/\mathrm{len}(s) + tk^2)$. The expected running time of trial division up to s is $O(ks/\mathrm{len}(s)^2)$. This estimate does not take into account the time to generate the small primes using the Sieve of Eratosthenes. These values might be pre-computed, in which case this time is zero, but even if we compute them on the fly, this takes time $O(s\,\mathrm{len}(\mathrm{len}(s)))$, which is dominated by $O(ks/\mathrm{len}(s)^2))$ for any reasonable value of s (in particular, for $s \leq k^{O(1)}$).

So provided $s = o(k^2 \mathrm{len}(k))$, the running time of *MRS* will be dominated by the Miller-Rabin test, which is what we want, of course — if we spend as much time sieving as the time it would take to perform a Miller-Rabin test, we might as well just perform the Miller-Rabin test. In practice, one would use a very conservative bound for s, probably no more than k^2, since getting s arbitrarily close to optimal does not really provide that much benefit, while if we choose s too large, it can actually do significant harm.

From the above estimates, we can conclude that with $k \leq s \leq k^2$, the expected running time W'_M of $MRS(n, t, s)$, with respect to a randomly chosen n between 1 and M, is

$$W'_M = O(k^3/\mathrm{len}(k) + tk^2). \tag{10.6}$$

From this, it follows that the expected running time of algorithm RP on input M is $O(k^4/\mathrm{len}(k) + tk^3)$. Thus, we effectively reduce the running time by a factor of $\mathrm{len}(k)$, which is a very real and noticeable improvement in practice.

As we already mentioned, the above analysis is heuristic, but the results are correct. To make the analysis rigorous, we need prove that the estimate $\tau(M, s) \approx \prod_{p \leq s}(1 - 1/p)$ is indeed accurate. Proving such estimates takes us into the realm of "sieve theory." The larger M is with respect to s, the easier it is to prove such estimates. We shall prove only the simplest and most naive such estimates, but it is still good enough for our purposes, if we do not care too much about hidden 'O'-constants.

Before stating any results, let us restate the problem slightly. For real $y \geq 0$, let us call a positive integer "y-rough" it is not divisible by any primes p up to y. For real $x \geq 0$, let us define $R(x, y)$ to be the number of y-rough integers up to x. Thus, $\tau(M, s) = R(M, s)/M$.

Theorem 10.18 *For any real $x \geq 0$ and $y \geq 0$, we have*

$$\left| R(x, y) - x \prod_{p \leq y}(1 - 1/p) \right| \leq 2^{\pi(y)}.$$

Proof. To simplify the notation, we shall use the Möbius function μ (see §2.5). Also, for a real number u, let us write $u = \lfloor u \rfloor + \{u\}$, where $0 \leq \{u\} < 1$. Let P be the product of the primes up to the bound y.

Now, there are $\lfloor x \rfloor$ positive integers up to x, and of these, for each prime p dividing P, precisely $\lfloor x/p \rfloor$ are divisible by p, for each pair p, p' of distinct primes dividing P, precisely $\lfloor x/pp' \rfloor$ are divisible by pp', etc. By inclusion/exclusion, we have

$$R(x,y) = \sum_{d|P} \mu(d)\lfloor x/d \rfloor = \sum_{d|P} \mu(d)(x/d) - \sum_{d|P} \mu(d)\{x/d\}.$$

Moreover,

$$\sum_{d|P} \mu(d)(x/d) = x \sum_{d|P} \mu(d)/d = x \prod_{p \le y}(1 - 1/p),$$

and

$$|\sum_{d|P} \mu(d)\{x/d\}| \le \sum_{d|P} 1 = 2^{\pi(y)}.$$

That proves the theorem. □

This theorem only says something non-trivial when y is quite small. Nevertheless, using Chebyshev's Theorem on the density of primes, along with Mertens' Theorem, it is not hard to see that this theorem implies that $\tau(M,s) = O(1/\log s)$ when $s = O(\log M \log\log M)$, which implies the estimate (10.6) above. We leave the details as an exercise for the reader.

Exercise 10.19 Prove the claim made above that $\tau(M,s) = O(1/\log s)$ when $s = O(\log M \log\log M)$. More precisely, show that there exist constants c, d, and s_0, such that for all M and d satisfying $s_0 \le s \le c \log M \log\log M$, we have $\tau(M,s) \le d/\log s$. From this, derive the estimate (10.6) above. □

Exercise 10.20 Let f be a polynomial with integer coefficients. For real $x \ge 0$ and $y \ge 0$, define $R_f(x,y)$ to be the number of integers m up to x such that $f(m)$ is y-rough. For positive integer M, define $\omega_f(M)$ to be the number of integers $m \in \{0, \ldots, M-1\}$ such that $f(m) \equiv 0 \pmod{M}$. Show that

$$\left| R_f(x,y) - x \prod_{p \le y}(1 - \omega_f(p)/p) \right| \le \prod_{p \le y}(1 + \omega_f(p)).$$

□

Exercise 10.21 Using a sieve, design an algorithm that improves the running time of the algorithm in Exercise 10.16 to $O(k^5/\operatorname{len}(k)^2 + tk^3)$ — under the same assumptions, and achieving the same error probability bound. Hint: first show that the previous exercise implies that the number of positive integers m up to x such that both m and $2m+1$ are y-rough is at most

$$x \cdot \frac{1}{2} \prod_{2 < p \le y} (1 - 2/p) + 3^{\pi(y)}.$$

□

Exercise 10.22 Design an algorithm that takes as input a prime q and a bound M, and outputs a random prime p between 1 and M such that $p \equiv 1 \pmod{q}$. Clearly, we need to assume that M is sufficiently large with respect to q. Analyze your algorithm assuming Conjecture 5.41. State how large M must be with respect to q, and under these assumptions, show that your algorithm runs in time $O(k^4/\operatorname{len}(k) + tk^3)$, and that its output is incorrect with probability $O(4^{-t}k)$. As usual, $k := \operatorname{len}(M)$. □

10.4.3 Generating a random k-bit prime

In some applications, we want to generate a random prime of fixed size, e.g., a random 1024-bit prime. More generally, let us consider the following problem: given integer $k \geq 3$, generate a random k-bit prime, i.e., a prime in the interval $[2^{k-1}, 2^k)$.

Theorem 5.11 (Bertrand's Postulate) implies that there exists a constant $c > 0$ such that $\pi(2^k) - \pi(2^{k-1}) \geq c2^{k-1}/k$ for all $k \geq 3$.

Now let us modify algorithm RP so that it takes as input integer $k \geq 3$, and repeatedly generates a random n in the interval $\{2^{k-1}, \ldots, 2^k - 1\}$ until *IsPrime*(n) returns *true*. Let us call this variant algorithm RP′. Further, let us implement *IsPrime*$(\cdot)$ as $MR(\cdot, t)$, for some auxiliary parameter t, and define $\gamma'(k, t)$ to be the probability that the output of algorithm RP′ — with this implementation of *IsPrime* — is composite.

Then using exactly the same reasoning as above,

$$\gamma'(k, t) \leq 4^{-t} \frac{2^{k-1}}{\pi(2^k) - \pi(2^{k-1})} = O(4^{-t}k).$$

As before, if the output of algorithm RP′ is prime, then every k-bit prime is equally likely, and the expected running time is $O(k^4 + tk^3)$. By using a sieve as in the previous section, this can be reduced to $O(k^4/\operatorname{len}(k) + tk^3)$.

The function $\gamma'(k, t)$ has been studied a good deal; for example, the following is known:

Theorem 10.23 *For all $k \geq 3$, we have*

$$\gamma'(k, 1) \leq k^2 4^{2-\sqrt{k}}.$$

Proof. Literature — see §10.8. □

Upper bounds for $\gamma'(k, t)$ for specific values of k and t have been computed. The following table lists some known *lower* bounds for $-\log_2(\gamma'(k, t))$ for various values of k and t:

$t \backslash k$	200	300	400	500	600
1	11	19	37	56	75
2	25	33	46	63	82
3	34	44	55	70	88
4	41	53	63	78	95
5	47	60	72	85	102

Using exactly the same reasoning as the derivation of (10.5), one sees that

$$\gamma'(k,t) \leq \frac{\gamma'(k,1)}{1-\gamma'(k,1)} 4^{-t+1}.$$

10.5 Perfect Power Testing and Prime Power Factoring

Consider the following problem: we are given a integer $n \geq 2$, and want to determine if n is a **perfect power**, i.e., if $n = d^e$ for integers d and e, both greater than 1. Certainly, if such d and e exist, then we must be the case that $2^e \leq n$, so we can try all possible candidate values of e, running from 2 to $\lfloor \log_2 n \rfloor$. For each such candidate value of e, we can test if $n = d^e$ for some d as follows. Suppose n is a k-bit number, i.e., $2^{k-1} \leq n < 2^k$. Then $2^{(k-1)/e} \leq n^{1/e} < 2^{k/e}$. So any integer eth root of n must lie in the set $\{u, \ldots, v-1\}$, where $u = 2^{\lfloor (k-1)/e \rfloor}$ and $v = 2^{\lceil k/e \rceil}$. Using u and v as starting values, we can perform a binary search:

1. if $u \geq v$, declare that n is not a perfect eth power;
2. set $w \leftarrow \lfloor (u+v)/2 \rfloor$;
3. set $z \leftarrow w^e$;
4. (a) if $z = n$, then declare than $n = w^e$ is an a perfect eth power;
   (b) otherwise, if $z < n$, recursively apply binary search using $(w+1, v)$ in place of (u, v);
   (c) otherwise, if $z > n$, recursively apply binary search using (u, w) in place of (u, v).

If $n = d^e$ for some integer d, then the following invariant holds (verify): at the beginning of each recursive step, we have $u \leq d < v$. Thus, if n is a perfect eth power, this will be discovered. That proves the correctness of the algorithm.

As to its running time, note that with each recursive step, the length $v - u$ of the search interval decreases by a factor of at least 2 (verify). Therefore, after t steps the interval will be of length at most $2^{k/e+1}/2^t$, so after at most $k/e + 2$ steps, the interval will be of length less than 1, and hence of length zero, and the algorithm will halt. So the number of recursive steps is $O(k/e)$. The power w^e computed in each step is no more than $2^{(k/e+1)e} = 2^{k+e} \leq 2^{2k}$, and hence can be computed in time $O(k^2)$ (see Exercise 3.17). Hence the overall cost of testing if n is an eth power using this algorithm is $O(k^3/e)$.

Trying all candidate values of e from 1 to $\lfloor \log_2 n \rfloor$ yields an overall running time for perfect power testing of $O(\sum_e k^3/e) = O(k^3 \operatorname{len}(k))$. To find the largest possible value of e for which n is an eth power, we should examine the candidates from highest to lowest.

Using the above algorithm for perfect power testing and an efficient primality test, we can determine if an integer n is a prime power p^e, and if so, compute p and e: we find the largest positive integer e (possibly 1) such that $n = d^e$ for integer d, and test if d is a prime using an efficient primality test.

10.6 Factoring and Computing Euler's ϕ-Function are Equivalent

In this section, we use some of the ideas developed to analyze the Miller-Rabin test to prove that the problem of factoring n and the problem of computing $\phi(n)$ are equivalent. By equivalent, we mean that given an efficient algorithm to solve one problem, we can efficiently solve the other, and *vice versa*.

Clearly, one direction is easy: if we can factor n into primes, so

$$n = p_1^{e_1} \cdots p_r^{e_r}, \tag{10.7}$$

then we can simply compute $\phi(n)$ using the formula

$$\phi(n) = p_1^{e_1-1}(p_1 - 1) \cdots p_r^{e_r-1}(p_r - 1).$$

For the other direction, it is more convenient to prove a stronger result: given any multiple of the *exponent* of $\mathbb{Z}_n^*$, we can efficiently factor n. In particular, this will show that we can efficiently factor Carmichael numbers.

We first introduce some notation; namely, let $\lambda(n)$ denote the exponent of $\mathbb{Z}_n^*$. If the prime factorization of n is as in (10.7), then by the Chinese Remainder Theorem, we have

$$\lambda(n) = \operatorname{lcm}(\lambda(p_1^{e_1}), \ldots, \lambda(p_r^{e_r})).$$

Moreover, for any prime power p^e, by Theorem 10.1, we have

$$\lambda(p^e) = \begin{cases} p^{e-1}(p-1) & \text{if } p \neq 2 \text{ or } e \leq 2, \\ 2^{e-2} & \text{if } p = 2 \text{ and } e \geq 3. \end{cases}$$

In particular, if $m \mid n$, then $\lambda(m) \mid \lambda(n)$.

Now, returning to our factorization problem, we are given n and a multiple f of $\lambda(n)$, and want to factor n. We may as will assume that n is odd; otherwise, we can pull out all the factors of 2, obtaining n' such that $n = 2^e n'$, where n' is odd and f is a multiple of $\lambda(n')$, thus, reducing to the odd case.

So now, assume n is odd and f is a multiple of $\lambda(n)$. Assume that f is of the form $f = 2^h m$, where m is odd. Our factoring algorithm, which we describe recursively, runs as follows (recall the notation "rep($\cdot$)" from §3.4).

if n is a prime power p^e then
    output e copies of p and return
generate a random, nonzero element α of $\mathbb{Z}_n$
$d_1 \leftarrow \gcd(\mathrm{rep}(\alpha), n)$
if $d_1 \neq 1$, then recursively factor d_1 and n/d_1 (using the same f), and return
$\alpha \leftarrow \alpha^m$
for $j \leftarrow 0$ to $h-1$ do
    $d_2 \leftarrow \gcd(\mathrm{rep}(\alpha)+1, n)$
    if $d_2 \notin \{1, n\}$, then recursively factor d_2 and n/d_2 (using the same f), and return
    $\alpha \leftarrow \alpha^2$
recursively factor n (using the same f)

It is clear that when the algorithm terminates, its output consists of the list of all primes (including duplicates) dividing n, assuming the primality test does not make a mistake.

To analyze the running time of the algorithm, assume that the prime factorization of n is as in (10.7). By the Chinese Remainder Theorem, we have an isomorphism of groups

$$\rho : \mathbb{Z}^*_{p_1^{e_1}} \times \cdots \times \mathbb{Z}^*_{p_r^{e_r}} \to \mathbb{Z}^*_n.$$

Let $\lambda(p_i^{e_i}) = m_i 2^{h_i}$, where m_i is odd, for $1 \leq i \leq r$, and let $\ell := \max\{h_1, \ldots, h_r\}$. Note that since $\lambda(n) \mid f$, we have $\ell \leq h$.

Consider one execution of the body of the recursive algorithm. If n is a prime power, this will be detected immediately, and the algorithm will return. Here, even if we are using probabilistic primality test, such as the Miller-Rabin test, that always says that a prime is a prime, the algorithm will certainly halt. So assume that n is not a prime power; i.e., $r \geq 2$. If the chosen value of α is not in $\mathbb{Z}^*_n$, then d_1 will be a nontrivial divisor of n. Otherwise, conditioning on the event that $\alpha \in \mathbb{Z}^*_n$, the distribution of α is uniform over $\mathbb{Z}^*_n$. Consider the value $\beta := \alpha^{m2^{\ell-1}}$.

We claim that with probability at least $1/2$, $\gcd(\mathrm{rep}(\beta)+1, n)$ is a nontrivial divisor of n. To prove this claim, let us write

$$\beta = \rho(\beta_1, \ldots, \beta_r),$$

where $\beta_i \in \mathbb{Z}^*_{p_i^{e_i}}$. Note that for those i with $h_i < \ell$, the $m2^{\ell-1}$-power map kills the group $\mathbb{Z}^*_{p_i^{e_i}}$, while for those i with $h_i = \ell$, the image of $\mathbb{Z}^*_{p_i^{e_i}}$ under the $m2^{\ell-1}$-power is $[\pm 1]$. Without loss of generality, assume that the indices i such that $h_i = \ell$ are numbered $1, \ldots, r'$, where $1 \leq r' \leq r$. The values β_i for $1 \leq i \leq r'$ are uniformly and independently distributed over $[\pm 1]$, while for all $i > r'$, $\beta_i = [1]$. Thus, the value of $\gcd(\mathrm{rep}(\beta)+1, n)$ is the product of all prime powers $p_i^{e_i}$, with $\beta_i = [-1]$, which will be nontrivial unless either (1) all the β_i are $[1]$, or (2) $r' = r$ and all the β_i are $[-1]$. Consider two cases. First, if $r' < r$, then only event (1) is possible, and this occurs with probability $2^{-r'} \leq 1/2$. Second, if $r' = r$, then each of

events (1) and (2) occur with probability 2^{-r}, and so the probability that either occurs is $2^{-r+1} \leq 1/2$. That proves the claim.

From the claim, it follows that with probability at least $1/2$, we will obtain a nontrivial divisor d_2 of n when $j = \ell - 1$ (if not before).

So we have shown that with probability at least $1/2$, one execution of the body will succeed in splitting n into nontrivial factors. After at most $\log_2 n$ such successes, we will have completely factored n. Therefore, the expected number of recursive invocations of the algorithm is $O(\text{len}(n))$, and hence the expected running time of the algorithm is $O(\text{len}(n)^4)$.

10.7 The RSA Cryptosystem

Algorithms for testing and generating large primes have numerous applications in cryptography. One of the most well known and important such applications is the RSA cryptosystem, named after its inventors Rivest, Shamir, and Adleman. We give a brief overview of this system here.

Suppose that Alice wants to send a secret message to Bob over an insecure network. An adversary may be able to eavesdrop on the network, and so sending the message "in the clear" is not an option. Using older, more traditional cryptographic techniques would require that Alice and Bob share a secret key between them; however, this creates the problem of securely generating such a shared secret. The RSA cryptosystem is an example of a "public key" cryptosystem. To use the system, Bob simply places a "public key" in the equivalent of an electronic telephone book, while keeping a corresponding "private key" secret. To send a secret message to Bob, Alice obtains Bob's public key fro the telephone book, and uses this to encrypt her message. Upon receipt of the encrypted message, Bob uses his secret key to decrypt it, obtaining the original message.

Here is how the RSA cryptosystem works. To generate a public key/private key pair, Bob generates two very large random primes p and q. To be secure, p and q should be quite large — typically, they are chosen to be around 512 bits in length. We require that $p \neq q$, but the probability that two random 512-bit primes are equal is negligible, so this is hardly an issue. Next, Bob computes $n := pq$. Bob also selects an integer $e > 1$ such that $\gcd(e, \phi(n)) = 1$. Here, $\phi(n) = (p-1)(q-1)$. Finally, Bob computes the multiplicative inverse d of e modulo $\phi(n)$, i.e., d satisfies $ed \equiv 1 \pmod{\phi(n)}$. The public key is the pair (n, e), and the private key is the pair (n, d). The integer e is called the "encryption exponent" and d is called the "decryption exponent."

After Bob publishes his public key (n, e), Alice may send a secret message to Bob as follows. Suppose that a message is encoded in some canonical way as a number between 0 and $n-1$ — we can always interpret a bit string of length less than $\text{len}(n)$ as such a number. Thus, we may assume that a message is an element α of $\mathbb{Z}_n$. To encrypt the message α, Alice simply computes $\beta := \alpha^e$. The encrypted message is β. When Bob received β, he computes $\gamma := \beta^d$, and interprets γ as a message.

The most basic requirement of any encryption scheme is that decryption should "undo"

encryption. In this case, this means that for all $\alpha \in \mathbb{Z}_n$, we should have

$$(\alpha^e)^d = \alpha. \tag{10.8}$$

If $\alpha \in \mathbb{Z}_n^*$, then this is clearly the case, since we have $ed = 1 + \phi(n)k$ for some positive integer k, and hence

$$(\alpha^e)^d = \alpha^{ed} = \alpha^{1+\phi(n)k} = \alpha \cdot \alpha^{\phi(n)k} = \alpha,$$

where the last equality follows from the fact that the multiplicative order of α divides the order of the group, $\phi(n)$. Even if $\alpha \notin \mathbb{Z}_n^*$, equation (10.8) still holds. To see this, let $\alpha = [a \bmod n]$, with $\gcd(a, n) \neq 1$. There are three possible cases. First, if $a \equiv 0 \pmod{n}$, then trivially, $a^{ed} \equiv 0 \pmod{n}$. Second, if $a \equiv 0 \pmod{p}$ but $a \not\equiv 0 \pmod{q}$, then trivially $a^{ed} \equiv 0 \pmod{p}$, and

$$a^{ed} \equiv a^{1+\phi(n)k} \equiv a \cdot a^{\phi(n)k} \equiv a \pmod{q},$$

where the last congruence follows from the fact that $\phi(n)k$ is a multiple of $q - 1$, and so is a multiple of the order of $[a \bmod q] \in \mathbb{Z}_q^*$. The third case, where $a \not\equiv 0 \pmod{p}$ and $a \equiv 0 \pmod{q}$, is treated in the same way as the second. Thus, we have shown that equation (10.8) holds for all $\alpha \in \mathbb{Z}_n$.

Note that in place of d, one could also use as a decryption exponent any d' such that $ed' \equiv 1 \pmod{\lambda(n)}$, where $\lambda(n) = \operatorname{lcm}(p - 1, q - 1)$ is the exponent of the group $\mathbb{Z}_n^*$.

Of course, the interesting question about the RSA cryptosystem is whether or not it really is secure. Now, if an adversary, given only the public key (n, e), were able to compute the decryption exponent d, then since $ed - 1$ is a multiple of $\phi(n)$, then by the results in the previous section, the adversary would already be able to factor n. The same holds if the adversary is able to compute any "equivalent" decryption exponent d', with $ed' \equiv 1 \pmod{\lambda(n)}$.

Thus, we can say that as long as factoring n is computationally infeasible, then recovering a decryption exponent, given only the public key, is also computationally infeasible. However, even if we assume that factoring large numbers is infeasible, this is not enough to guarantee that for a given encrypted message β, the adversary is unable to compute β^d. Nevertheless, nobody knows how to efficiently compute β^d for arbitrary β, without first factoring n.

The reader should be warned that the proper notion of security for an encryption scheme is quite subtle, and a detailed discussion of this is well beyond the scope of this text. Indeed, the simple version of the RSA cryptosystem presented here is in fact inadequate from a security point of view, and because of this, actual implementations of public-key encryption schemes based on RSA are somewhat more complicated.

10.8 Notes

The Miller-Rabin test is due to Miller [48], and Rabin [58]. The paper by Miller defined the set L'_n, but did not give a probabilistic analysis. Rather, Miller showed that under a generalization of the Riemann Hypothesis, for composite n, the least $\alpha \in \mathbb{Z}_n^{\neq} \setminus L'_n$ is at most

$O((\log n)^2)$, thus giving rise to a deterministic primality test whose correctness depends on the above unproved hypothesis. The later paper by Rabin re-interprets Miller's result in the context of probabilistic algorithms.

Bach [9] gives an explicit version of Miller's result, showing that under the same assumptions, the least $\alpha \in \mathbb{Z}_n^{\neq} \setminus L'_n$ is at most $2(\log n)^2$. The first efficient probabilistic primality test was invented by Solovay and Strassen [72] (their paper was actually submitted for publication in 1974). Later, in §21, we shall discuss a recently discovered, deterministic, polynomial-time (though not very practical) primality test, whose analysis does not rely on any unproved hypothesis.

Carmichael numbers are named after R. D. Carmichael, who was the first to discuss them in work published in the early 20th century. Alford, Granville, and Pomerance [6] proved that there are infinitely many Carmichael numbers.

Theorem 10.14, as well as the table of values just below it, are from Kim and Pomerance [40]. In fact, these bounds hold for the weaker test based on L_n.

Theorem 10.18 and its generalization in Exercise 10.20 are certainly not the best results possible in this area. The general goal of "sieve theory" is to prove useful upper and lower bounds for quantities like $R_f(x, y)$ that hold when y is as large as possible with respect to x. For example, using a technique known as Brun's Pure Sieve, one can show that for $\log y < \sqrt{\log x}$, there exist β and β', both of absolute value at most 1, such that

$$R_f(x, y) = (1 + \beta e^{-\sqrt{\log x}})x \prod_{p \leq y} (1 - \omega_f(p)/p) + \beta' \sqrt{x}.$$

Thus, this gives us very sharp estimates for $R_f(x, y)$ when x tends to infinity, and y is bounded by any fixed polynomial in $\log x$. For a proof of this result, see §2.2 of Halberstam and Richert [31] (the result itself is stated as equation 2.16). Brun's Pure Sieve is really just the first non-trivial sieve result, developed in the early 20th century; even stronger results, extending the useful range of y (but with larger error terms), have subsequently been proved.

Theorem 10.23, as well as the table of values immediately below it, are from Damgård, Landrock, and Pomerance [24].

The RSA cryptosystem was invented by Rivest, Shamir, and Adleman [60]. There is a vast literature on cryptography. One starting point is the book by Menesez, van Oorschot, and Vanstone [47].

Chapter 11

Computing Generators and Discrete Logarithms in $\mathbb{Z}_p^*$

As we have seen in the previous chapter, for a prime p, $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$. This means that there exists a generator $\gamma \in \mathbb{Z}_p^*$, such that for all $\alpha \in \mathbb{Z}_p^*$, α can be written uniquely as $\alpha = \gamma^x$ for $0 \leq x < p-1$; the integer x is called the **discrete logarithm** of α to the base γ, and is denoted $\log_\gamma \alpha$.

This chapter discusses some elementary considerations regarding the computational aspects of this situation; namely, how to efficiently find a generator γ, and given γ and α, how to compute $\log_\gamma \alpha$.

More generally, if γ generates a subgroup G of $\mathbb{Z}_p^*$ of order q, where $q \mid (p-1)$, and $\alpha \in G$, then $\log_\gamma \alpha$ is defined to be the unique integer x with $0 \leq x < q$ and $\alpha = \gamma^x$. In some situations it is more convenient to view $\log_\gamma \alpha$ as an element of $\mathbb{Z}_q$. Also for $x \in \mathbb{Z}_q$, with $x = [a \bmod q]$, one may write γ^x to denote γ^a. There can be no confusion, since if $x = [a' \bmod q]$, then $\gamma^{a'} = \gamma^a$. However, in this chapter, we shall view $\log_\gamma \alpha$ as an integer.

Although we work in the group $\mathbb{Z}_p^*$, all of the algorithms discussed in this chapter trivially generalize to any finite cyclic group that has a suitably compact representation of group elements and an efficient algorithm for performing the group operation on these representations.

11.1 Finding a Generator for $\mathbb{Z}_p^*$

There is no efficient algorithm known for this problem, unless the prime factorization of $p-1$ is given, and even then, we must resort to the use of a probabilistic algorithm. Of course, factoring in general is believed to be a very difficult problem, so it may not be easy to get the prime factorization of $p-1$. However, if our goal is to construct a large prime p, together with a generator for $\mathbb{Z}_p^*$, then we may use the algorithm in §7.7 to generate a random factored number n in some range, test $n+1$ for primality, and then repeat until we get a factored number n such that $p = n+1$ is prime. In this way, we can generate a

random prime p in a given range along with the factorization of $p-1$.

We now present an efficient probabilistic algorithm that takes as input an odd prime p, along with the prime factorization

$$p - 1 = \prod_{i=1}^{r} q_i^{e_i},$$

and outputs a generator for $\mathbb{Z}_p^*$. It runs as follows:

```
for i ← 1 to r do
    repeat
        choose α ∈ Z_p^* at random
        compute β ← α^{(p-1)/q_i}
    until β ≠ 1

    γ_i ← α^{(p-1)/q_i^{e_i}}

γ ← ∏_{i=1}^{r} γ_i
output γ
```

First, let us analyze the correctness of this algorithm. When the ith loop iteration terminates, by construction, we have

$$\gamma_i^{q_i^{e_i}} = 1 \quad \text{but} \quad \gamma_i^{q_i^{e_i-1}} \neq 1.$$

It follows (c.f., Theorem 8.79) that γ_i has order $q_i^{e_i}$. From this, it follows (c.f., Theorem 8.80) that γ has order $p-1$.

Thus, we have shown that if the algorithm terminates, its output is always correct.

Let us now analyze the running time of this algorithm. Consider the repeat/until loop in the ith iteration of the outer loop. Since α is chosen at random from $\mathbb{Z}_p^*$, the value of β is uniformly distributed over the image of the $(p-1)/q_i$-power map (c.f., Exercise 8.65), and since the latter is a subgroup of order q_i, we see that $\beta = 1$ with probability $1/q_i$. It follows that the expected number of iterations of the repeat/until loop is $O(1)$, and therefore, the expected running time of the entire algorithm is $O(r\,\mathrm{len}(p)^3)$, and since $r \leq \log_2 p$, this is $O(\mathrm{len}(p)^4)$.

Exercise 11.1 Suppose we are not given the prime factorization of $p-1$, but rather, just a prime q dividing $p-1$, and we want to find an element of order q in $\mathbb{Z}_p^*$. Design and analyze an efficient algorithm to do this. □

Exercise 11.2 Suppose we are given a prime p, the prime factorization $p - 1 = \prod_{i=1}^{r} q_i^{e_i}$, and an element $\alpha \in \mathbb{Z}_p^*$.

(a) Show how to compute the order of α in time $O(r\,\mathrm{len}(p)^3)$.

(b) Improve the running time bound to $O(\mathrm{len}(r)\,\mathrm{len}(p)^3)$.

□

11.2 Computing Discrete Logarithms $\mathbb{Z}_p^*$

In this section, we consider algorithms for computing the discrete logarithm of $\alpha \in \mathbb{Z}_p^*$ to a given base γ. The algorithms we present here are in the worst case exponential-time algorithms, and are by no means the best possible; however, in some special cases, these algorithms are not so bad.

11.2.1 Brute-force search

Suppose that $\gamma \in \mathbb{Z}_p^*$ generates a subgroup G of order q (not necessarily prime), and we are given p, q, γ, and $\alpha \in G$, and wish to compute $\log_\gamma \alpha$.

The simplest algorithm to solve the problem is **brute-force search**:

$\beta \leftarrow 1$
$i \leftarrow 0$
while $\beta \neq \alpha$ do
    $\beta \leftarrow \beta \cdot \gamma$
    $i \leftarrow i + 1$
output i

This algorithm is clearly correct, and the main loop will always halt after at most q iterations (assuming, as we are, that $\alpha \in G$). So the total running time is $O(q\,\mathrm{len}(p)^2)$.

11.2.2 Baby step/giant step method

As above, suppose that $\gamma \in \mathbb{Z}_p^*$ generates a subgroup G of order q (not necessarily prime), and we are given p, q, γ, and $\alpha \in G$, and wish to compute $\log_\gamma \alpha$.

A faster algorithm than brute-force search is the **baby step/giant step method**. It works as follows.

Let us choose an approximation m to $q^{1/2}$. It does not have to be a very good approximation — we just need $m = \Theta(q^{1/2})$. Also, let $m' = \lfloor q/m \rfloor$, so that $m' = \Theta(q^{1/2})$ as well.

The idea is to compute all the values γ^i for $0 \leq i < m$ (the "baby steps") and to build a "lookup table" L that contains all the pairs (γ^i, i). Using an appropriate data structure, such as a *search trie*, we can build the table in time $O(m\,\mathrm{len}(p)^2)$, and we can perform a lookup in time $O(\mathrm{len}(p))$. By a lookup, we mean that given $\beta \in \mathbb{Z}_p^*$, we can determine if $\beta = \gamma^i$ for some i, and if so, determine the value of i. Let us define $L(\beta) := i$ if $\beta = \gamma^i$ for some i; and otherwise, $L(\beta) := -1$.

After building the lookup table, we execute the following procedure:

$\gamma' \leftarrow \gamma^{-m}$
$\beta \leftarrow \alpha$; $j \leftarrow 0$; $i \leftarrow L(\beta)$
while $i = -1$ do
    $\beta \leftarrow \beta \cdot \gamma'$; $j \leftarrow j + 1$; $i \leftarrow L(\beta)$

$x \leftarrow jm + i$
output x

To analyze this procedure, suppose that $\alpha = \gamma^x$ for $0 \le x < q$. Now, x can be written in a unique way as $x = vm + u$, where $0 \le u < m$ and $0 \le v \le m'$. In the jth loop iteration, for $j = 0, 1, \ldots$, we have

$$\beta = \alpha\gamma^{-mj} = \gamma^{(v-j)m+u}.$$

So we will find that $i \neq -1$ precisely when $j = v$, in which case $i = u$. Thus, the output will be correct, and the total running time of the algorithm is easily seen to be $O(q^{1/2}\,\mathrm{len}(p)^2)$.

While this algorithm is much faster than brute-force search, it has the drawback that it requires a table of size $O(q^{1/2})$. Of course, there is a "time/space trade-off" here: by choosing m smaller, we get a table of size $O(m)$, but the running time will be proportional to $O(q/m)$. In §11.2.5 below, we discuss an algorithm that runs (at least heuristically) in time proportional to $O(q^{1/2})$, but which requires only a constant amount of space.

11.2.3 Groups of order q^e

Suppose that $\gamma \in \mathbb{Z}_p^*$ generates a subgroup G of order q^e, where $q > 1$ and $e \ge 1$, and we are given p, q, γ, and $\alpha \in G$, and wish to compute $\log_\gamma \alpha$.

There is a simple algorithm that allows one to reduce this problem to the problem of computing discrete logarithms in a subgroup of order q.

It is perhaps easiest to describe the algorithm recursively.

The base case is when $e = 1$, in which case, we use an algorithm for the subgroup of order q.

Suppose now that $e > 1$. We choose an integer f with $0 < f < e$. Different strategies for choosing f yield different algorithms — we discuss this below. Suppose $\alpha = \gamma^x$, where $0 \le x < q^e$. Then we can write $x = q^f v + u$, where $0 \le u < q^f$ and $0 \le v < q^{e-f}$. Therefore,

$$\alpha^{q^{e-f}} = \gamma^{q^{e-f}u}.$$

Note that $\gamma^{q^{e-f}}$ has order q^f, and so if we recursively compute the discrete logarithm of $\alpha^{q^{e-f}}$ to the base $\gamma^{q^{e-f}}$, we obtain u.

Having obtained u, observe

$$\alpha/\gamma^u = \gamma^{q^f v}.$$

Note also that γ^{q^f} has order q^{e-f}, and so if we recursively compute the discrete logarithm of α/γ^u to the base γ^{q^f}, we obtain u, from which we then compute $x = q^f v + u$.

To analyze the running time of this algorithm, note that we recursively reduce the discrete logarithm problem to a base of order q^e to two discrete logarithm problems: one to a base of order q^f and the other to a base of order q^{e-f}. The running time of the body of one recursive invocation (not counting the running time of the recursive calls it makes) is $O(e \operatorname{len}(q) \cdot \operatorname{len}(p)^2)$.

To calculate the total running time, we have to sum up the running times of all the recursive calls plus the running times of all the base cases.

Regardless of the strategy for choosing f, the total number of base case invocations is e. Note that for $e > 1$, all the base cases compute discrete logarithms are to the base $\gamma^{q^{e-1}}$. Assuming we implement the base case using the baby step/giant step algorithm, the total running time for all the base cases is therefore $O(eq^{1/2} \operatorname{len}(p)^2)$.

The running time for the recursive calls depends on the strategy used to choose f. If we always choose $f = 1$ or $f = e - 1$, then the running time is for all the recursive calls is $O(e^2 \operatorname{len}(q) \cdot \operatorname{len}(p)^2)$. However, if we use a "balanced" divide-and-conquer strategy, choosing $f \approx e/2$, then we get $O(e \operatorname{len}(e) \operatorname{len}(q) \cdot \operatorname{len}(p)^2)$.

In summary, the total running time is:

$$O((eq^{1/2} + e \operatorname{len}(e) \operatorname{len}(q)) \cdot \operatorname{len}(p)^2).$$

11.2.4 Discrete logarithms in $\mathbb{Z}_p^*$

Suppose that we are given a prime p, along with the prime factorization

$$p - 1 = \prod_{i=1}^{r} q_i^{e_i},$$

a generator γ for $\mathbb{Z}_p^*$, and $\alpha \in \mathbb{Z}_p^*$. We wish to compute $\log_\gamma \alpha$.

Suppose that $\alpha = \gamma^x$, where $0 \leq x < p - 1$. Then for $1 \leq i \leq r$,

$$\alpha^{(p-1)/q_i^{e_i}} = \gamma^{(p-1)/q_i^{e_i} x}.$$

Note that $\gamma^{(p-1)/q_i^{e_i}}$ has order $q_i^{e_i}$, and if x_i is the discrete logarithm of $\alpha^{(p-1)/q_i^{e_i}}$ to the base $\gamma^{(p-1)/q_i^{e_i}}$, then we have $0 \leq x_i < q_i^{e_i}$ and $x \equiv x_i \pmod{q_i^{e_i}}$.

Thus, if we compute the values $x_1, \ldots, x_r$, using the algorithm in §11.2.3, we can obtain x using the algorithm of the Chinese Remainder Theorem. If we define $q := \max\{q_i : 1 \leq i \leq r\}$, then the running time of this algorithm will be bounded by $q^{1/2} \operatorname{len}(p)^{O(1)}$.

11.2.5 A space-efficient square-root time algorithm

We present a more space-efficient alternative to the algorithm in §11.2.2, the analysis of which we leave as a series of exercises to the reader.

The algorithm makes a somewhat heuristic assumption that we have a function that "behaves" for all practical purposes like a random function. Such functions can indeed be constructed using cryptographic techniques under reasonable intractability assumptions.

Let p be a prime, q a prime dividing $p-1$, $\gamma \in \mathbb{Z}_p^*$ an element of $\mathbb{Z}_p^*$ that generates a subgroup G of order q, and $\alpha \in G$. Let F be a function mapping elements of $\mathbb{Z}_p^*$ to $\{0, \ldots, q-1\}$. Define H to be the function from G to G that sends β to $\beta\alpha\gamma^{F(\beta)}$.

The algorithm runs as follows:

$i \leftarrow 1$
$x \leftarrow 0$, $\beta \leftarrow \alpha$,
$x' \leftarrow F(\beta)$, $\beta' \leftarrow H(\beta)$
while $\beta \neq \beta'$ do
    $x \leftarrow (x + F(\beta)) \text{ rem } q$, $\beta \leftarrow H(\beta)$
    $x' \leftarrow (x' + F(\beta')) \text{ rem } q$, $\beta' \leftarrow H(\beta')$
    $x' \leftarrow (x' + F(\beta')) \text{ rem } q$, $\beta' \leftarrow H(\beta')$
    $i \leftarrow i + 1$
if $i < q$ then
    output $(x - x')i' \text{ rem } q$, where $ii' \equiv 1 \pmod{q}$
else output "fail"

Define $\beta_1, \beta_2, \ldots,$ as follows: $\beta_1 = \alpha$ and for $i > 1$, $\beta_i = H(\beta_{i-1})$.

Exercise 11.3 Show that each time the main loop of the algorithm is entered, we have $\beta = \beta_i = \gamma^x \alpha^i$, and $\beta' = \beta_{2i} = \gamma^{x'} \alpha^{2i}$. □

Exercise 11.4 Show that if the loop terminates with $i < q$, the the value output is equal to $\log_\gamma \alpha$. □

Exercise 11.5 Let j be the smallest index such that $\beta_j = \beta_k$ for some index $k < j$. Show that $j \leq q + 1$ and that the loop terminates after less than j loop iterations, i.e., the value of i when the loop terminates is less than j (and in particular, $i \leq q$). □

Exercise 11.6 Assume F is a random function, that is, the random variables $F(\beta)$, as β ranges over G, are mutually independent and uniformly distributed over $\{0, \ldots, q-1\}$. Show that this implies that H is a random function, that is, the random variables $H(\beta)$ are mutually independent and uniformly distributed over G. □

Exercise 11.7 Assuming that F is a random function as in the previous exercise, show that for any fixed, positive integer k, the probability that $j \geq k$ (where j is as defined in Exercise 11.5) is at most $e^{-k(k-1)/2q}$. □

Exercise 11.8 From part the previous exercise, conclude that the expected value of j is $O(q^{1/2})$, and hence the expected running time of the algorithm is $O(q^{1/2})$ times a polynomial in $\text{len}(p)$. □

11.3 The Diffie-Hellman Key Establishment Protocol

One of the main motivations for studying algorithms for computing discrete logarithms is the relation between this problem and the problem of breaking a protocol called the Diffie-Hellman Key Establishment Protocol, named after its inventors.

In this protocol, Alice and Bob need never to have talked to each other before, but nevertheless, can establish a shared secret key that nobody else can easily compute. To use this protocol, a third party must provide a "telephone book," which contains the following information:

- p, q, and γ, where p and q are primes with $q \mid (p-1)$, and γ is an element generating a subgroup G of order q in $\mathbb{Z}_p^*$;
- an entry for each user, such as Alice or Bob, that contains the user's name, along with a "public key" for that user, which is an element of the group G.

To use this system, Alice posts her public key in the telephone book, which is of the form $\alpha = \gamma^x$, where $x \in \{0, \ldots, q-1\}$ is chosen by Alice at random. The value of x is Alice's "secret key," which Alice never divulges to anybody. Likewise, Bob posts his public key, which is of the form $\beta = \gamma^y$, where $y \in \{0, \ldots, q-1\}$ is chosen by Bob at random, and is his secret key.

To establish a shared key known only between them, Alice retrieves Bob's public key β from the bulletin board, and computes $\kappa_A := \beta^x$. Likewise, Bob retrieves Alice's public key α, and computes $\kappa_B := \alpha^y$. It is easy to see that

$$\kappa_A = \beta^x = (\gamma^y)^x = \gamma^{xy} = (\gamma^x)^y = \alpha^y = \kappa_B,$$

and hence Alice and Bob share the same secret key $\kappa = \kappa_A = \kappa_B$.

Using this shared secret key, they can then use standard methods for encryption and message authentication to hold a secure conversation. We shall not go any further into how this is done; rather, we briefly discuss some aspects (but only superficially) of the security of the key establishment protocol itself. Clearly, if an attacker obtains α and β from the telephone book, and computes $x = \log_\gamma \alpha$, then he can compute Alice and Bob's shared key as $\kappa = \beta^x$ — in fact, given x, an attacker can efficiently compute *any* key shared between Alice and another user.

Thus, if this system is to be secure, it should be very difficult to compute discrete logarithms. However, the assumption that computing discrete logarithms is hard is not enough to guarantee security. Indeed, it is not entirely inconceivable that the discrete logarithm problem is hard, and yet the problem of computing κ from α and β is easy. The latter problem — computing κ from α and β — is called the *Diffie-Hellman problem*.

As in the discussion of the RSA cryptosystem in §10.7, the reader is warned that the above discussion about security is a bit of an oversimplification. A complete discussion of all the security issues related to the above protocol is beyond the scope of this text.

For the following exercise, we need the following notions from complexity theory:

- We say problem A is **deterministic poly-time reducible** to problem B if there exists a deterministic algorithm R for solving problem A that makes calls to a subroutine for problem B, where the running time of R (not including the running time for the subroutine for B) is polynomial in the input length.
- We say that A and B are **deterministic poly-time equivalent** if A is deterministic poly-time reducible to B and B is deterministic poly-time reducible to A.

Exercise 11.9 Show that the following problems are deterministic poly-time equivalent:

(a) Given a prime p, a prime q that divides $p-1$, an element $\gamma \in \mathbb{Z}_p^*$ generating a subgroup G of order q, and two elements $\alpha, \beta \in G$, compute γ^{xy}, where $x = \log_\gamma \alpha$ and $y = \log_\gamma \beta$. This is the *Diffie-Hellman problem.*

(b) Given a prime p, a prime q that divides $p-1$, an element $\gamma \in \mathbb{Z}_p^*$ generating a subgroup G of order q, and an element $\alpha \in G$, compute γ^{x^2}, where $x = \log_\gamma \alpha$.

(c) Given a prime p, a prime q that divides $p-1$, an element $\gamma \in \mathbb{Z}_p^*$ generating a subgroup G of order q, and two elements $\alpha, \beta \in G$, with $\beta \neq [1 \bmod p]$, compute $\gamma^{xy'}$, where $x = \log_\gamma \alpha$ and y' is the multiplicative inverse modulo q of $y = \log_\gamma \beta$.

(d) Given a prime p, a prime q that divides $p-1$, an element $\gamma \in \mathbb{Z}_p^*$ generating a subgroup G of order q, and an element $\alpha \in G$, with $\alpha \neq [1 \bmod p]$, compute $\gamma^{x'}$, where x' is the multiplicative inverse modulo q of $x = \log_\gamma \alpha$.

□

11.4 Notes

As we already mentioned, all of the algorithms presented in this chapter are completely "generic," in the sense that they work in *any* finite cyclic group — we really did not exploit any properties about $\mathbb{Z}_p^*$ other than the fact that it is a cyclic group. In fact, as far as such "generic" algorithms go, the algorithms presented here for discrete logarithms are optimal [51, 71]. However, there are faster, "non-generic" algorithms (though still not polynomial time) for discrete logarithms in $\mathbb{Z}_p^*$. We shall examine one such algorithm in a later chapter.

Knuth [41] attributes the "baby step/giant step" algorithm in §11.2.2 to Dan Shanks. The algorithms in §11.2.3 and §11.2.4 are variants of an algorithm published by Pohlig and Hellman [54]. The algorithm in §11.2.5 is a variant of an algorithm of Pollard [55]; in fact, Pollard's algorithm is a bit more efficient than the one presented here, but the analysis of its running time depends on stronger heuristics.

The key establishment protocol in §11.3 is from Diffie and Hellman [25].

Chapter 12

Quadratic Residues and Quadratic Reciprocity

12.1 Quadratic Residues

For positive integer n, an integer a is called a **quadratic residue modulo** n if $\gcd(a, n) = 1$ and $x^2 \equiv a \pmod{n}$ for some integer x; in this case, we say that x is a **square root of** a **modulo** n.

The quadratic residues modulo n correspond exactly to the subgroup of squares $(\mathbb{Z}_n^*)^2$ of $\mathbb{Z}_n^*$; that is, a is a quadratic residue modulo n if and only if $[a \bmod n] \in (\mathbb{Z}_n^*)^2$.

Let us first consider the case where $n = p$, where p is an odd prime. In this case, we know that $\mathbb{Z}_p^*$ is cyclic of order $p - 1$. Recall that the subgroups any finite cyclic group are in one-to-one correspondence with the divisors of the order of the group.

For any $d \mid (p-1)$, consider the d-power map on $\mathbb{Z}_p^*$ that sends $\alpha \in \mathbb{Z}_p^*$ to α^d. The image of this map is the unique subgroup of $\mathbb{Z}_p^*$ of order $(p-1)/d$, and the kernel of this map is the unique subgroup of order d (c.f., Theorem 8.75). This means that the image of the 2-power map is of order $(p-1)/2$ and must be the same as the kernel of the $(p-1)/2$-power map. Since the image of the $(p-1)/2$-power map is of order 2, it must be equal to the subgroup $\{[\pm 1 \bmod p]\}$. The kernel of the 2-power map is of order 2, and so must also be equal to the subgroup $\{[\pm 1 \bmod p]\}$.

Translating from group-theoretic language to the language of congruences, we have shown:

Theorem 12.1 *For an odd prime p, the number of quadratic residues a modulo p, with $0 < a < p$, is $(p-1)/2$. Moreover, if x is a square root of a modulo p, then so is $-x$, and any square root y of a modulo p satisfies $y \equiv \pm x \pmod{p}$. Also, for any integer $a \not\equiv 0 \pmod{p}$, we have $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$, and moreover, a is a quadratic residue modulo p if and only if $a^{(p-1)/2} \equiv 1 \pmod{p}$.*

Now consider the case where $n = p^e$, where p is an odd prime and $e > 1$. We also know that $\mathbb{Z}_{p^e}^*$ is a cyclic group of order $p^{e-1}(p-1)$, and so everything that we said in

discussing the case $\mathbb{Z}_p^*$ applies here as well. Thus, for $a \not\equiv 0 \pmod{p}$, a is a quadratic residue modulo p^e if and only if $a^{p^{e-1}(p-1)/2} \equiv 1 \pmod{p^e}$. However, we can simplify this a bit. Note that $a^{p^{e-1}(p-1)/2} \equiv 1 \pmod{p^e}$ implies $a^{p^{e-1}(p-1)/2} \equiv 1 \pmod{p}$, and by Theorem 8.72 (Fermat's Little Theorem), this implies $a^{(p-1)/2} \equiv 1 \pmod{p}$. Conversely, by Theorem 10.3, $a^{(p-1)/2} \equiv 1 \pmod{p}$ implies $a^{p^{e-1}(p-1)/2} \equiv 1 \pmod{p^e}$. Thus, we have shown:

Theorem 12.2 *For an odd prime p and positive integer e, the number of quadratic residues a modulo p^e, with $0 < a < p^e$, is $p^{e-1}(p-1)/2$. Moreover, if x is a square root of a modulo p^e, then so is $-x$, and any square root y of a modulo p^e satisfies $y \equiv \pm x \pmod{p^e}$. Also, for any integer $a \not\equiv 0 \pmod{p}$, we have $a^{p^{e-1}(p-1)/2} \equiv \pm 1 \pmod{p}$, and moreover, a is a quadratic residue modulo p^e iff $a^{p^{e-1}(p-1)/2} \equiv 1 \pmod{p^e}$ iff $a^{(p-1)/2} \equiv 1 \pmod{p}$ iff a is a quadratic residue modulo p.*

Now consider an arbitrary odd positive integer n. Let $n = \prod_{i=1}^r p_i^{e_i}$ be its prime factorization. Recall the group isomorphism implied by the Chinese Remainder Theorem:

$$\mathbb{Z}_n^* \cong \mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_r^{e_r}}^*.$$

Now,

$$(\alpha_1, \ldots, \alpha_r) \in \mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_r^{e_r}}^*$$

is a square if and only if there exist $\beta_1, \ldots, \beta_r$ with $\beta_i \in \mathbb{Z}_{p_i^{e_i}}^*$ and $\alpha_i = \beta_i^2$ for $1 \leq i \leq k$, in which case, we see that the square roots of $(\alpha_1, \ldots, \alpha_r)$ comprise the 2^r elements $(\pm\beta_1, \ldots, \pm\beta_r)$. Thus we have:

Theorem 12.3 *Let n be odd positive integer n with prime factorization $n = \prod_{i=1}^r p_i^{e_i}$. The number of quadratic residues a modulo n, with $0 < a < n$, is $\phi(n)/2^r$. Moreover, if a is a quadratic residue modulo n, then there are precisely 2^r distinct integers x, with $0 < x < n$, such that $x^2 \equiv a \pmod{n}$. Also, an integer a is a quadratic residue modulo n if and only if it is a quadratic residue modulo p_i for $1 \leq i \leq r$.*

That completes our investigation of the case where n is an odd positive integer. We shall not investigate the case where n is even, as it is a bit cumbersome, and is not of particular importance.

12.2 The Legendre Symbol

For an odd prime p and an integer a with $\gcd(a, p) = 1$, the **Legendre symbol** $(a \mid p)$ is defined to be 1 if a is a quadratic residue modulo p, and -1 otherwise. For completeness, one defines $(a \mid p) = 0$ if $p \mid a$.

Theorem 12.4 *Let p be an odd prime, and let $a, b \in \mathbb{Z}$, both not divisible by p. Then*

1. $(a \mid p) \equiv a^{(p-1)/2} \pmod{p}$; *in particular,* $(-1 \mid p) = (-1)^{(p-1)/2}$;
2. $(a \mid p)(b \mid p) = (ab \mid p)$;
3. $a \equiv b \pmod{p}$ *implies* $(a \mid p) = (b \mid p)$;
4. $(2 \mid p) = (-1)^{(p^2-1)/8}$;
5. *if q is an odd prime different from p, then*

$$(p \mid q)(q \mid p) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}.$$

Part (5) of this theorem is called the Law of Quadratic Reciprocity.

Part (1) follows from Theorem 12.1. Part (2) is an immediate consequence of part (1), and part (3) is clear from the definition.

The rest of this section is devoted to a proof of parts (4) and (5) of this theorem. The proof is completely elementary, although a bit technical.

Theorem 12.5 (Gauss' Lemma) *Let p be an odd prime and a relatively prime to p. Define $\alpha_j := ja \bmod p$ for $1 \leq j \leq (p-1)/2$, and let n be the number of indices j for which $\alpha_j > p/2$. Then $(a \mid p) = (-1)^n$.*

Proof. Let $r_1, \ldots, r_n$ denote the α_j's exceeding $p/2$, and let $s_1, \ldots, s_k$ denote the remaining α_j's. The r_i and s_i are all distinct and non-zero. We have $0 < p - r_i < p/2$ for $1 \leq i \leq n$, and no $p - r_i$ is an s_j; indeed, if $p - r_i = s_j$, then $s_j \equiv -r_j \pmod{p}$, and writing $s_j = k_1 a$ and $r_j = k_2 a$ for $1 \leq k_1, k_2 \leq (p-1)/2$, we have $k_1 a \equiv -k_2 a \pmod{p}$, which implies $k_1 \equiv -k_2 \pmod{p}$, which is impossible.

It follows that the sequence of numbers $s_1, \ldots, s_k, p - r_1, \ldots, p - r_n$ is just a re-ordering of $1, \ldots, (p-1)/2$. Then we have

$$\begin{aligned} ((p-1)/2)! &\equiv s_1 \cdots s_k(-r_1) \cdots (-r_n) \equiv (-1)^n s_1 \cdots s_k r_1 \cdots r_n \\ &\equiv (-1)^n ((p-1)/2)! a^{(p-1)/2} \pmod{p}, \end{aligned}$$

and canceling the factor $((p-1)/2)!$, we obtain $a^{(p-1)/2} \equiv (-1)^n \pmod{p}$, and the result follows from the fact that $(a \mid p) \equiv a^{(p-1)/2} \pmod{p}$. □

Theorem 12.6 *If p is an odd prime and $\gcd(a, 2p) = 1$, then $(a \mid p) = (-1)^t$ where $t = \sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor$. Also, $(2 \mid p) = (-1)^{(p^2-1)}/8$.*

Proof. Let a be an integer relatively prime to p (not necessarily odd), and let us adopt the same notation as in the proof of Theorem 12.5. Note that $ja = p\lfloor ja/p \rfloor + \alpha_j$, for $1 \leq j \leq k$, so we have

$$\sum_{j=1}^{(p-1)/2} ja = \sum_{j=1}^{(p-1)/2} p\lfloor ja/p \rfloor + \sum_{j=1}^{n} r_j + \sum_{j=1}^{k} s_j.$$

Also, we saw in the proof of Theorem 12.5 that the integers $s_1, \ldots, s_k, p - r_1, \ldots, p - p_n$ are a re-ordering of $1, \ldots, (p-1)/2$, and hence

$$\sum_{j=1}^{(p-1)/2} j = \sum_{j=1}^{n}(p - r_j) + \sum_{j=1}^{k} s_j = np - \sum_{j=1}^{n} r_j + \sum_{j=1}^{k} s_j.$$

Subtracting, we get

$$(a-1)\sum_{j=1}^{(p-1)/2} j = p\left(\sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor - n\right) + 2\sum_{j=1}^{n} r_j.$$

Note that

$$\sum_{j=1}^{(p-1)/2} j = \frac{p^2-1}{8},$$

which implies

$$(a-1)\frac{p^2-1}{8} \equiv \sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor - n \pmod 2.$$

If a is odd,this implies

$$n \equiv \sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor \pmod 2.$$

If $a = 2$, this — along with the fact that $\lfloor 2j/p \rfloor = 0$ for $1 \leq j \leq (p-1)/2$ — implies

$$n \equiv \frac{p^2-1}{8} \pmod 2.$$

The theorem now follows from Theorem 12.5. □

Note that this last theorem proves part (4) of Theorem 12.4. The next theorem proves part (5).

Theorem 12.7 *If p and q are distinct odd primes, then*

$$(p \mid q)(q \mid p) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}.$$

Proof. Let S be the set of pairs of integers (x, y) with $1 \leq x \leq (p-1)/2$ and $1 \leq y \leq (q-1)/2$. Note that S contains no pair (x, y) with $qx = py$, so let us partition S into two subsets: S_1 contains all pairs (x, y) with $qx > py$, and S_2 contains all pairs (x, y) with $qx < py$. Note that $(x, y) \in S_1$ if and only if $1 \leq x \leq (p-1)/2$ and $1 \leq y \leq \lfloor qx/p \rfloor$. So $|S_1| = \sum_{x=1}^{(p-1)/2} \lfloor qx/p \rfloor$. Similarly, $|S_2| = \sum_{y=1}^{(q-1)/2} \lfloor py/q \rfloor$. So we have

$$\frac{p-1}{2}\frac{q-1}{2} = |S| = |S_1| + |S_2| = \sum_{x=1}^{(p-1)/2} \lfloor qx/p \rfloor + \sum_{y=1}^{(q-1)/2} \lfloor py/q \rfloor,$$

and Theorem 12.6 implies

$$(p \mid q)(q \mid p) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}.$$

That proves the first statement of the theorem. The second statement follows immediately. □

12.3 The Jacobi Symbol

Let a, n be integers, where n is positive and odd, so that $n = q_1 \cdots q_k$, where the q_i are odd primes, not necessarily distinct. Then the **Jacobi symbol** $(a \mid n)$ is defined as

$$(a \mid n) := (a \mid q_1) \cdots (a \mid q_k),$$

where $(a \mid q_j)$ is the Legendre symbol. Note that $(a \mid 1) = 1$ for all $a \in \mathbb{Z}$. Thus, the Jacobi symbol essentially extends the domain of definition of the Legendre symbol. Note that $(a \mid n) \in \{0, \pm 1\}$.

Theorem 12.8 *Let m, n be positive, odd integers, an let a, b be integers. Then*

1. $(ab \mid n) = (a \mid n)(b \mid n)$;
2. $(a \mid mn) = (a \mid m)(a \mid n)$;
3. $a \equiv b \pmod{n}$ *implies* $(a \mid n) = (b \mid n)$;
4. $(-1 \mid n) = (-1)^{(n-1)/2}$;
5. $(2 \mid n) = (-1)^{(n^2-1)/8}$;
6. *if* $\gcd(m, n) = 1$, *then*
$$(m \mid n)(n \mid m) = (-1)^{\frac{m-1}{2}\frac{n-1}{2}}.$$

Proof. Parts (1)–(3) follow directly from the definition (exercise).

For parts (4) and (6), one can easily verify (exercise) that for odd integers $n_1, \ldots, n_k$,

$$\sum_{i=1}^{k} (n_i - 1)/2 \equiv (n_1 \cdots n_k - 1)/2 \pmod{2}.$$

Part (4) easily follows from this fact, along with part (2) of this theorem and part (1) of Theorem 12.4 (exercise). Part (6) easily follows from this fact, along with parts (1) and (2) of this theorem, and part (5) of Theorem 12.4 (exercise).

For part (5), one can easily verify (exercise) that for odd integers $n_1, \ldots, n_k$,

$$\sum_{1 \le i \le k} (n_i^2 - 1)/8 \equiv (n_1^2 \cdots n_k^2 - 1)/8 \pmod{2}.$$

Part (5) easily follows from this fact, along with part (2) of this theorem, and part (4) of Theorem 12.4 (exercise). □

As we shall see later, this theorem is extremely useful from a computational point of view — with it, one can efficiently compute $(a \mid n)$, without having to know the prime factorization of either a or n. Also, in applying this theorem it is useful to observe that for odd integers m, n,

- $(-1)^{(n-1)/2} = 1$ iff $n \equiv 1 \pmod 4$;
- $(-1)^{(n^2-1)/8} = 1$ iff $n \equiv \pm 1 \pmod 8$;
- $(-1)^{((m-1)/2)((n-1)/2)} = 1$ iff $m \equiv 1 \pmod 4$ or $n \equiv 1 \pmod 4$.

Finally, we note that if a is a quadratic residue modulo n, then $(a \mid n) = 1$; however, $(a \mid n) = 1$ does not imply that a is a quadratic residue modulo n.

12.4 Notes

The proof we present here of Theorem 12.4 is essentially the one from Niven and Zuckerman [52]. Our proof of Theorem 12.8 is essentially the one found in in Bach and Shallit [11].

Chapter 13

Computational Problems Related to Quadratic Residues

13.1 Computing the Jacobi Symbol

Suppose we are given an odd, positive integer n, along with an integer a, and we want to compute the Jacobi symbol $(a \mid n)$. Theorem 12.8 suggests the following algorithm:

$t \leftarrow 1$
repeat
    — *loop invariant: n is odd and positive*

    $a \leftarrow a \bmod n$
    if $a = 0$
        if $n = 1$ return t else return 0

    compute a', h such that $a = 2^h a'$ and a' is odd
    if $h \not\equiv 0 \pmod 2$ and $n \not\equiv \pm 1 \pmod 8$ then $t \leftarrow -t$
    if $a' \not\equiv 1 \pmod 4$ and $n \not\equiv 1 \pmod 4$ then $t \leftarrow -t$
    $(a, n) \leftarrow (n, a')$
forever

That this algorithm correctly computes the Jacobi symbol $(a \mid n)$ follows directly from Theorem 12.8. Using an analysis similar to that of Euclid's algorithm, one easily sees that the running time of this algorithm is $O(\text{len}(a)\,\text{len}(n))$.

Exercise 13.1 Develop a "binary" Jacobi symbol algorithm, i.e., one that uses only addition, subtractions, and "shift" operations, analogous to the binary gcd algorithm in Exercise 4.4. □

13.2 Testing Quadratic Residuosity

13.2.1 Prime modulus

For an odd prime p, we can test if a is a quadratic residue modulo p by either performing the exponentiation $a^{(p-1)/2} \operatorname{rem} p$ or by computing the Legendre symbol $(a \mid p)$. Using a standard repeated squaring algorithm, the former method takes time $O(\mathrm{len}(p)^3)$, while using the Euclidean-like algorithm of the previous section, the latter method takes time $O(\mathrm{len}(p)^2)$. So presumably, the latter method is to be preferred.

13.2.2 Prime-power modulus

For an odd prime p, we know that a is a quadratic residue modulo p^e if and only if a is a quadratic residue modulo p. So this case immediately reduces to the previous case.

13.2.3 Composite modulus

For odd, composite n, if we know the factorization of n, then we can also determine if a is a quadratic residue modulo n by determining if it is a quadratic residue modulo each prime divisor p of n. However, without knowledge of this factorization (which is in general believed to be hard to compute), there is no efficient algorithm known. We can compute the Jacobi symbol $(a \mid n)$; if this is -1 or 0, we can conclude that a is not a quadratic residue; otherwise, we cannot conclude much of anything.

13.3 Computing Modular Square Roots

13.3.1 Prime modulus

Let p be an odd prime, and suppose that $(a \mid p) = 1$. Here is one way to compute a square root of a modulo p, assuming we have at hand an integer y such that $(y \mid p) = -1$.

Let $\alpha = [a \bmod p] \in \mathbb{Z}_p^*$ and $\gamma = [y \bmod p] \in \mathbb{Z}_p^*$. The above problem is equivalent to finding $\beta \in \mathbb{Z}_p^*$ such that $\beta^2 = \alpha$.

Let us write $p - 1 = 2^h m$, where m is odd. For any $\delta \in \mathbb{Z}_p^*$, δ^m has order dividing 2^h. Since $\alpha^{2^{h-1}m} = 1$, α^m has order dividing 2^{h-1}. Since $\gamma^{2^{h-1}m} = [-1 \bmod p]$, γ^m has order precisely 2^h. Since there is only one subgroup in $\mathbb{Z}_p^*$ of order 2^h, it follows that γ^m generates this subgroup, and that $\alpha^m = \gamma^{mx}$ for $0 \leq x < 2^h$ and x is even. We can find x by computing the discrete logarithm of α^m to the base γ^m, using the algorithm in §11.2.3. Setting $\kappa = \gamma^{mx/2}$, we have

$$\kappa^2 = \alpha^m.$$

We are not quite done, since we now have a square root of α^m, and not of α. However, because $\gcd(m, 2) = 1$, we can find integers s, t such that $ms + 2t = 1$. In fact, $s = 1$ and $t = -\lfloor m/2 \rfloor$ do the job. It then follows that

$$(\kappa^s \alpha^t)^2 = \kappa^{2s} \alpha^{2t} = \alpha^{ms} \alpha^{2t} = \alpha^{ms+2t} = \alpha.$$

Thus, $\kappa^s \alpha^t$ is a square root of α.

The total amount of work done outside the discrete logarithm calculation amounts to just a handful of exponentiations modulo p, and so takes time $O(\text{len}(p)^3)$. The time to compute the discrete logarithm is $O(h\,\text{len}(h)\,\text{len}(p)^2)$. So the total running time of this procedure is

$$O(\text{len}(p)^3 + h\,\text{len}(h)\,\text{len}(p)^2).$$

The above procedure assumed we had at hand a non-square γ. If $h = 1$, i.e., $p \equiv 3 \pmod 4$, then -1 is a quadratic residue modulo p, and so we are done. In fact, in this case, the the output of the above procedure is simply $\alpha^{(p+1)/4}$, no matter what value of γ is used. One can easily show directly that $\alpha^{(p+1)/4}$ is a square root of α, without analyzing the above procedure.

If $h > 1$, we can find a non-square γ using a probabilistic algorithm. Simply choose γ at random, test if it is a square, and repeat if not. The probability that a random element of $\mathbb{Z}_p^*$ is a square is $1/2$; thus, the expected number of trials is $O(1)$, and hence the expected running time of this probabilistic algorithm is $O(\text{len}(p)^2)$.

Example 13.2 Of course, we can combine any algorithms for testing quadratic residuosity and computing square roots modulo p with the familiar "quadratic formula" (see Exercise 9.43) to find the roots of arbitrary quadratic polynomials modulo p. That is, given a prime p along with $\alpha, \beta, \gamma \in \mathbb{Z}_p$ with $\alpha \neq 0$, we can determine the roots of the polynomial $\alpha \mathtt{X}^2 + \beta \mathtt{X} + \gamma$ by computing $\delta := \beta^2 - 4\alpha\gamma$, and testing if $\delta \in (\mathbb{Z}_p)^2$ (of course, $\delta = 0$ is allowed); if not, the polynomial has no roots in $\mathbb{Z}_p$; otherwise, we can compute a square root ζ of δ, and compute the roots of the polynomial as $(-\beta + \zeta)/(2\alpha)$ and $(-\beta - \zeta)/(2\alpha)$ (which will be the same, of course, if and only if $\zeta = \delta = 0$). □

Exercise 13.3 Show that the following two problems are deterministic, poly-time equivalent (see discussion just above Exercise 11.9 in §11.3):

(a) Given an odd prime p and $\alpha \in (\mathbb{Z}_p^*)^2$, find $\beta \in \mathbb{Z}_p^*$ such that $\beta^2 = \alpha$.

(b) Given an odd prime p, find an element of $\mathbb{Z}_p^* \setminus (\mathbb{Z}_p^*)^2$.

□

Exercise 13.4 Design and analyze an efficient, deterministic algorithm that takes as input primes p and q, such that $q \mid (p-1)$, along with an element $\alpha \in \mathbb{Z}_p^*$, and determines whether or not α is a perfect qth power, i.e., whether or not there exists $\beta \in \mathbb{Z}_p^*$ such that $\beta^q = \alpha$. □

Exercise 13.5 Design and analyze a probabilistic algorithm that takes as input primes p and q, such that $q \mid (p-1)$, along with an element $\alpha \in \mathbb{Z}_p^*$ that is a perfect qth power, and returns a qth root of α, i.e., an element $\beta \in \mathbb{Z}_p^*$ such that $\beta^q = \alpha$. Your algorithm should have an expected running time that is bounded by $q^{1/2}$ times a polynomial in $\text{len}(p)$. □

13.3.2 Prime-power modulus

Again, for an odd prime p, we know that a is a quadratic residue modulo p^e if and only if a is a quadratic residue modulo p.

Suppose we have found an integer z such that $z^2 \equiv a \pmod{p}$, using, say, the procedure described above. From this, we can easily compute a square root of a modulo p^e using the following technique, which is known as **Hensel lifting**.

More generally, suppose we have integers a, z such that $z^2 \equiv a \pmod{p^f}$, for $f \geq 1$, and we want to find an integer $\hat{z}$ such that $\hat{z}^2 \equiv a \pmod{p^{f+1}}$. Clearly, if $\hat{z}^2 \equiv a \pmod{p^{f+1}}$, then $\hat{z}^2 \equiv a \pmod{p^f}$, and so $\hat{z} \equiv \pm z \pmod{p^f}$. So let us set $\hat{z} = z + up^f$, and solve for u. We have

$$\hat{z}^2 \equiv (z + up^f)^2 \equiv z^2 + 2p^f u + u^2 p^{2f} \equiv z^2 + 2p^f u \pmod{p^{f+1}}.$$

So we want to find integer u such that

$$2p^f u \equiv a - z^2 \pmod{p^{f+1}}.$$

Since $p^f \mid (z^2 - a)$, by Theorem 2.3, the above congruence holds if and only if

$$2u \equiv \frac{a - z^2}{p^f} \pmod{p}.$$

From this, we can easily compute the desired value u.

By iterating the above procedure, starting with a square root of a modulo p, we can quickly find a square root of a modulo p^e. We leave a detailed analysis of the running time of this procedure to the reader.

13.3.3 Composite modulus

To find square roots modulo n, where n is an odd composite modulus, if we know the prime factorization of n, then we can use the above procedures for finding square roots modulo primes and prime powers, and then use the algorithm of the Chinese Remainder Theorem to get a square root modulo n.

However, if the factorization of n is not known, then there is no efficient algorithm known for computing square roots modulo n. In fact, one can show that the problem of finding square roots modulo n is at least as hard as the problem of factoring n, in the sense that if there is an efficient algorithm for computing square roots modulo n, then there is an efficient (probabilistic) algorithm for factoring n.

Here is an algorithm to factor n, using a modular square-root algorithm as a subroutine. For simplicity, we assume that n is of the form $n = pq$, where p and q are distinct primes. Choose β to be a random, non-zero element of $\mathbb{Z}_n$. If $d := \gcd(\mathrm{rep}(\beta), n) > 1$, then output d (recall the notation "rep$(\cdot)$" from §3.4). Otherwise, set $\alpha := \beta^2$, and feed n and α to the modular square-root algorithm, obtaining a square root $\beta' \in \mathbb{Z}_n^*$ of α. If the square-root algorithm returns $\beta' \in \mathbb{Z}_n^*$ such that $\beta' = \pm\beta$, then output "failure"; otherwise, output $\gcd(\mathrm{rep}(\beta - \beta'), n)$, which is a non-trivial divisor of n.

Let us analyze this algorithm. If $d > 1$, we split n, so assume that $d = 1$. In this case, β is uniformly distributed over $\mathbb{Z}_n^*$, and α is uniformly distributed over $(\mathbb{Z}_n^*)^2$. Let us condition on an a *fixed* value of α. If $\rho : \mathbb{Z}_p \times \mathbb{Z}_q \to \mathbb{Z}_n$ is the ring isomorphism of the Chinese Remainder Theorem, and $\tilde{\beta} = \rho(\tilde{\beta}_1, \tilde{\beta}_2)$ is any fixed square root of α, then in this conditional probability distribution, β is uniformly distributed over the four square roots of α, namely, $\rho(\pm\tilde{\beta}_1, \pm\tilde{\beta}_2)$. Since the square-root algorithm receives no information about β other than the value α, the probability that $\beta' = \pm\beta$ is $1/2$, in which case we output "failure"; however, if $\beta' \neq \pm\beta$, and if we write $\beta = \rho(\beta_1, \beta_2)$, then we have either

$$\beta' = \rho(\beta_1, -\beta_2) \quad \text{or} \quad \beta' = \rho(-\beta_1, \beta_2),$$

and hence $\beta - \beta'$ is either $\rho(0, 2\beta_2)$ or $\rho(2\beta_1, 0)$. Since neither $2\beta_1$ nor $2\beta_2$ are zero, it follows that $\gcd(\mathrm{rep}(\beta - \beta'), n)$ is either p or q.

Exercise 13.6 Generalize the algorithm above to efficiently factor arbitrary integers, given a subroutine that computes arbitrary modular square roots. □

Chapter 14

Vector Spaces and Algebras

In this chapter, we introduce the basic definitions and results concerning vector spaces over an arbitrary field F. Many readers have likely seen these notions before, but perhaps only in the context of a specific field, such as the real or complex numbers, and not in the context of, say, finite fields, like $\mathbb{Z}_p$. The point of this chapter is to generalize these notions to arbitrary fields.

14.1 Definitions, Properties, and Some Examples

Definition 14.1 *An F-**vector space** is an abelian group V, which we shall write using additive notation, together a **scalar multiplication operation** that maps $a \in F$ and $\alpha \in V$ to an element $a\alpha \in V$, such that the following properties are satisfied for all $a, b \in F$ and $\alpha, \beta \in V$:*

1. $a(b\alpha) = (ab)\alpha$,
2. $(a + b)\alpha = a\alpha + b\alpha$,
3. $a(\alpha + \beta) = a\alpha + a\beta$,
4. $1_F\alpha = \alpha$.

One may also call an F-vector space V a **vector space over** F. Elements of V are often called **vectors**, and elements of F, **scalars**.

Note that for an F-vector space V, for fixed $a \in F$, the map that sends $\alpha \in V$ to $a\alpha \in V$ is a group homomorphism with respect to the additive group operation of V; likewise, for fixed $\alpha \in V$, the map that sends $a \in F$ to $a\alpha \in V$ is a group homomorphism from the additive group of F into the additive group of V.

The following theorem summarizes a few basic facts which follow directly from the observations in the previous paragraph, and basic facts about group homomorphisms (see Theorem 8.51):

Theorem 14.2 *If V is a vector space over F, then for all $a \in F$ and $\alpha \in V$, we have:*

1. $0_F\alpha = 0_V$,
2. $a0_V = 0_V$,
3. $(-a)\alpha = -(a\alpha) = a(-\alpha)$.

Proof. Exercise. □

The definition of a vector space includes the possibility of the **trivial** vector space, consisting of just the zero element 0_V.

Example 14.3 A simple but extremely important example of an F-vector space is the set $F^{\times n}$ of n-tuples of elements of F, where addition is defined component-wise, and scalar multiplication is defined in the natural way: the product of $a \in F$ and $(a_1, \ldots, a_n) \in F^{\times n}$ is $(aa_1, \ldots, aa_n)$. □

Example 14.4 If F is a field, then the ring of polynomials $F[\mathtt{X}]$ over F forms a vector space in the natural way, with addition and scalar multiplication defined in terms of the addition and multiplication operations of the ring. □

Example 14.5 If f is a monic polynomial over F of degree $\ell \geq 0$, then the quotient ring $R = F[\mathtt{X}]/(f)$ is a vector space over F, with addition defined in terms of the addition operation of R, and scalar multiplication defined by $a\alpha := [a \bmod f]\alpha$, for $a \in F$ and $\alpha \in R$. If $f = 1$, then R is just the trivial ring consisting of only the zero element. □

Example 14.6 If $V_1, \ldots, V_n$ are F-vector spaces, the so is the direct product $V_1 \times \cdots \times V_n$, where addition and scalar product are defined component-wise. □

Example 14.7 Let G be any group of prime exponent p, written additively. Then we may define a scalar multiplication that maps $[m \bmod p] \in \mathbb{Z}_p$ and $\alpha \in G$ to $m\alpha$. That this map is unambiguously defined follows from the fact that G has exponent p, so that if $m \equiv m' \pmod{p}$, we have $m\alpha - m'\alpha = (m - m')\alpha = 0_G$, since $p \mid (m - m')$. It follows immediately (see parts (7), (8), and (9) of Theorem 8.17) that this scalar multiplication operations makes G into a $\mathbb{Z}_p$-vector space. □

14.2 Subspaces and Quotient Spaces

The notions of subgroups and quotient groups extend in the obvious way to vector spaces.

Definition 14.8 *Let V be an F-vector space. A subset W is a **subspace** of V if*

- *W is an additive subgroup of V, and*

- *W is closed under scalar multiplication, i.e., for all $a \in F$ and $\alpha \in W$, we have $a\alpha \in W$.*

It is easy to see that a subspace W of V is also an F-vector space in its own right, with addition and scalar multiplication operations inherited from V.

If $\alpha_1, \ldots, \alpha_n$ are elements of V, then we can form the set, denoted $\mathrm{Span}_F(\alpha_1, \ldots, \alpha_n)$, of all **linear combinations** of $\alpha_1, \ldots, \alpha_n$, with coefficients taken from F:

$$\mathrm{Span}_F(\alpha_1, \ldots, \alpha_n) := \{a_1\alpha_1 + \cdots + a_n\alpha_n : a_1, \ldots, a_n \in F\}.$$

It is not hard to see (verify) that $\mathrm{Span}_F(\alpha_1, \ldots, \alpha_n)$ is a subspace of V, and is called the subspace **spanned** or **generated** by $\alpha_1, \ldots, \alpha_n$.

If W_1 and W_2 are subspaces of V, then $W_1 + W_2$ and $W_1 \cap W_2$ are not only a subgroups of V, they are also subspaces of V (verify).

If W is a subspace of V, then in particular, it is also a subgroup of V, and so we can form the quotient group V/W in the usual way (see §8.3). Moreover, because W is closed under scalar multiplication, we can also define a scalar multiplication on V/W in a natural way. Namely, for $a \in F$ and $\alpha \in V$, we define

$$a(\alpha + W) := (a\alpha) + W.$$

As usual, one must check that this definition is unambiguous, that is, that is, if $\alpha \equiv \alpha' \pmod{W}$, then $a\alpha \equiv a\alpha' \pmod{W}$. But this follows from the fact that W is closed under scalar multiplication (verify). One can also easily check (verify), that with scalar multiplication defined in this way, V/W is an F-vector space; it is called the **quotient space of V modulo W**.

Exercise 14.9 Show that a subset W of an F-vector space V is a subspace of V if (1) for all $\alpha, \beta \in W$, $\alpha + \beta \in W$, and (2) for all $a \in F$ and $\alpha \in W$, $a\alpha \in W$. □

14.3 Vector Space Homomorphisms and Isomorphisms

The notions of group homomorphisms and isomorphisms extend in the obvious way to vector spaces.

Definition 14.10 *Let V and V' be vector spaces over F. An F-**vector space homomorphism** from V to V' is a map $\rho : V \to V'$, such that*

- *ρ is a group homomorphism from V to V', and*
- *for all $a \in F$ and $\alpha \in V$, we have $\rho(a\alpha) = a\rho(\alpha)$.*

*If ρ is bijective, then it is called an F-**vector space isomorphism** of V with V', and if in addition, $V = V'$, then it is called an F-**vector space automorphism** on V.*

An F-vector space homomorphism is also called an **F-linear map**. We shall give preference this terminology from now on.

Just as for groups, it is easy to see (verify) that if $\rho : V \to V'$ and $\rho' : V' \to V''$ are F-linear maps, then so is their composition $\rho' \circ \rho : V \to V''$; also, if ρ is an isomorphism of V with V' (as vector spaces), then the inverse function ρ^{-1} is an isomorphism of V' with V (again, as vector spaces — verify), and we write $V \cong V'$.

Example 14.11 The vector spaces in Examples 14.3 and 14.5 are isomorphic, provided $n = \ell$. Indeed, one isomorphism is the map that sends $(a_1, \ldots, a_n) \in F^{\times n}$ to $[\sum_i a_i \mathtt{X}^{i-1} \bmod m] \in F[\mathtt{X}]/(m)$. $\square$

Example 14.12 If R and R' are ring extensions of a field F, and if ρ is a homomorphism of R into R' over F, then ρ is an F-linear map from R into R'. Indeed, for any $a \in F$ and $\alpha, \beta \in R$, we have $\rho(\alpha + \beta) = \rho(\alpha) + \rho(\beta)$ and $\rho(a\alpha) = \rho(a)\rho(\alpha) = a\rho(\alpha)$. $\square$

Since a vector space homomorphism is also a group homomorphism, all of the statements in Theorem 8.51 apply. In particular, an F-linear map is injective if and only if the kernel is trivial (i.e., contains only the zero vector). However, in the case of vector space homomorphisms, we can extend Theorem 8.51, as follows:

Theorem 14.13 *Let $\rho : V \to V'$ be an F-linear map.*

1. *For any subspace of V, $\rho(V)$ is a subspace of V'.*
2. *$\ker(\rho)$ is a subspace of V.*
3. *For any subspace W' of V', $\rho^{-1}(W')$ is a subspace of V (and contains $\ker(\rho)$).*

Theorems 8.52, 8.53, 8.54, and 8.55 (for abelian groups) generalize immediately to vector spaces: all one has to check is that the relevant group homomorphisms are also vector vector space homomorphisms.

Theorem 14.14 *If W is a subspace of an F-vector space V, then the map $\rho : V \to V/W$ given by $\rho(\alpha) = \alpha + W$ is a surjective F-linear map whose kernel is W. This is sometimes called the "natural" map from V to V/W.*

Theorem 14.15 *Let ρ be an F-linear map from V into V'. Then the map $\bar{\rho} : V/\ker(\rho) \to \mathrm{im}(\rho)$ that sends the coset $\alpha + \ker(\rho)$ for $\alpha \in V$ to $\rho(\alpha)$ is unambiguously defined and is an F-vector space isomorphism of $V/\ker(\rho)$ with $\mathrm{im}(\rho)$.*

Theorem 14.16 *Let ρ be an F-linear map from V into V'. Then for any subspace W contained in $\ker(\rho)$, the map $\bar{\rho} : V/W \to \mathrm{im}(\rho)$ that sends the coset $\alpha + W$ for $\alpha \in V$ to $\rho(\alpha)$ is unambiguously defined and is an F-linear map from V/W onto $\mathrm{im}(\rho)$ with kernel $\ker(\rho)/W$.*

Theorem 14.17 *Let V be an F-vector space a with subspaces W_1, W_2 such that $W_1 \cap W_2 = \{0_V\}$. Then the map that sends $(\alpha_1, \alpha_2) \in W_1 \times W_2$ to $\alpha_1 + \alpha_2 \in W_1 + W_2$ is an F-vector space isomorphism of $W_1 \times W_2$ with $W_1 + W_2$.*

14.4 Linear Independence, Bases, and Dimension

Throughout this section, V is an F-vector space.

Definition 14.18 *We say that V is **finite dimensional** if it is spanned by a finite number of vectors, i.e., if $V = \mathrm{Span}_F(\alpha_1, \ldots, \alpha_n)$ for some $\alpha_1, \ldots, \alpha_n \in V$.*

*We say that a collection of vectors $\alpha_1, \ldots, \alpha_n$ in V is **linearly dependent (over F)** if there exist $a_1, \ldots, a_n \in F$, not all zero, such that $a_1\alpha_1 + \cdots a_n\alpha_n = 0_V$; otherwise, we say that $\alpha_1, \ldots, \alpha_n$ are **linearly independent (over F)**.*

*We say that a collection $\alpha_1, \ldots, \alpha_n$ of vectors in V is a **basis for V (over F)** if it is linearly independent and spans V.*

As a matter of definition, we consider the space spanned by the empty set of vectors to be the trivial subspace $\{0_V\}$. If V itself is the trivial vector space, then the empty set is the only basis for V.

Example 14.19 Consider the vector space $F^{\times 3}$. The vectors $(1,0,0), (0,1,0), (0,0,1)$ form a basis, as do the vectors $(1,1,1), (0,1,0), (-1,0,1)$. The vectors $(1,1,1), (0,1,0), (1,0,1)$ do not form a basis, as they are linearly dependent: the third vector is equal to the first minus the second. □

Example 14.20 The ring of polynomials $F[\mathtt{X}]$ is not finite dimensional, since any finite set of polynomials spans only polynomials of some bounded degree. □

Example 14.21 Consider again the ring $R = F[\mathtt{X}]/(f)$, where $f \in F[\mathtt{X}]$ is monic of degree $\ell \geq 0$, and consider the element $\eta = [\mathtt{X} \bmod f]$. If $f = 1$, then R is trivial, and so has dimension 0; otherwise, $1, \eta, \eta^2, \ldots, \eta^{\ell-1}$ form a basis for R over F. □

Example 14.22 If $\alpha_1, \ldots, \alpha_n$ form a basis for V, then the map ρ that sends $(a_1, \ldots, a_n) \in F^{\times n}$ to $a_1\alpha_1 + \cdots + a_n\alpha_n \in V$ is an F-vector space isomorphism of $F^{\times n}$ with V. To show this, one has to show (1) that ρ is an F-linear map, which follows immediately from the definitions, (2) that ρ is injective, which follows immediately from the linear independence of $\alpha_1, \ldots, \alpha_n$, and (3) that ρ is surjective, which follows immediately from the fact that $\alpha_1, \ldots, \alpha_n$ span V.

In particular, every element of V can be expressed in a unique way as $a_1\alpha_1 + \cdots + a_n\alpha_n$, for $a_1, \ldots, a_n \in F$. □

Exercise 14.23 Show that if a finite set S of vectors is linearly independent, then any subset of S is also linearly independent. □

Exercise 14.24 Show that if a finite collection of vectors contains the zero vector, or contains two identical vectors, then it is is not linearly independent. □

Exercise 14.25 Show that if S and S' are finite sets of vectors with $S \subset S'$, then the subspace spanned by S is contained in the subspace spanned by S'. □

Exercise 14.26 Show that if S and S' are finite sets of vectors such that every element of S can be expressed as a linear combination of elements in S', then the subspace spanned by S is contained in the subspace spanned by S'. □

The following two theorems are the keys to the theory of finite dimensional vector spaces.

Theorem 14.27 *If V is finite dimensional, then any finite set of vectors that spans V contains a subset which is a basis.*

Proof. We give an "algorithmic" proof. Let $\alpha_1, \ldots, \alpha_n$ be a given set of vectors that spans V. Let S_0 be the empty set, and for $i = 1, \ldots, n$, do the following: if α_i does not belong to the subspace spanned by S_{i-1}, set $S_i := S_{i-1} \cup \{\alpha_i\}$, and otherwise, set $S_i := S_{i-1}$. We claim that S_n is a basis for V.

First, we show that S_n spans V. To do this, first note that for $1 \leq i \leq n$, if α_i is not in S_n, then by definition, α_i is a linear combination of vectors in $S_{i-1} \subset S_n$. In any case, each α_i is a linear combination of the vectors in S_n. Since any element β of V is a linear combination of $\alpha_1, \ldots, \alpha_n$, and each α_i is a linear combination of elements of S_n, it follows (see Exercise 14.26) that β is a linear combination of elements of S_n.

Second, we show that S_n is linearly independent. Suppose it were not. Then we could express 0_V as a non-trivial linear combination of elements in S_n. Let us write this as

$$0_V = a_1\alpha_1 + a_2\alpha_2 + \cdots + a_n\alpha_n,$$

where the only non-zero coefficients a_i are those with $\alpha_i \in S_n$. If j is the highest index with $a_j \neq 0_F$, then by definition $\alpha_j \in S_n$. However, we see that α_j is in fact in the span of S_{j-1}; indeed,

$$\alpha_j = (-a_1 a_j^{-1})\alpha_1 + \cdots + (-a_{j-1}a_j^{-1})\alpha_{j-1},$$

and by definition, the only terms with non-zero coefficients are those corresponding to the vectors in S_{j-1}. This means that we would not have added α_j to S_j at step j, which means α_j is not in S_n, a contradiction. □

Theorem 14.28 *If V has a basis of size n, then any collection of $n+1$ elements of V is linearly dependent.*

Proof. Let $\alpha_1, \ldots, \alpha_n$ be a basis, and let $\beta_1, \ldots, \beta_{n+1}$ be any collection of $n+1$ vectors. We wish to show that $\beta_1, \ldots, \beta_{n+1}$ are linearly dependent.

Since the α_i's span V, we know that β_1 is a linear combination of the α_i's, say, $\beta_1 = a_1\alpha_1 + \cdots a_n\alpha_n$. If all the a_i's were zero, then we would have $\beta_1 = 0_V$, and so trivially, the β_j's would be linearly dependent (see Exercise 14.24). So assume that not all a_i's are zero, and for convenience, let us say that $a_1 \neq 0_F$. It follows that α_1 is a linear combination of $\beta_1, \alpha_2, \ldots, \alpha_n$, and hence $\beta_1, \alpha_2, \ldots, \alpha_n$ span V (see Exercise 14.26).

Next, consider β_2. This is a linear combination of $\beta_1, \alpha_2, \ldots, \alpha_n$, and we may assume that in this linear combination, the coefficient of one of $\alpha_2, \ldots, \alpha_n$ is non-zero (otherwise, we

find a linear dependence among the β_j's), and for convenience, let us say that the coefficient of α_2 is non-zero. As in the previous paragraph, it follows that $\beta_1, \beta_2, \alpha_3, \ldots, \alpha_n$ span V.

Continuing in this way, we find that $\beta_1, \ldots, \beta_n$ are either linearly dependent or they span V. In the latter case, we find that β_{n+1} is a linear combination of $\beta_1, \ldots, \beta_n$, and hence, the vectors $\beta_1, \ldots, \beta_n, \beta_{n+1}$ are linearly dependent. □

An important corollary of Theorem 14.28 is the following:

Theorem 14.29 *If V is finite dimensional, then any two bases have the same size.*

Proof. If one basis had more elements than another, then Theorem 14.28 would imply that the first basis was linearly dependent, which contradicts the definition of a basis. □

Theorem 14.29 allows us to make the following definition:

Definition 14.30 *If V is finite dimensional, the common size of any basis is called the* ***dimension*** *of V, and is denoted* $\dim_F(V)$.

To summarize the results in this section up to this point: if V is finite dimensional, it has a basis, and any two bases have the same size, which is called the dimension of V.

Another consequence of Theorem 14.28 is that if V is finite dimensional, and W is a subspace of V, then W is also finite dimensional, and $\dim_F(W) \leq \dim_F(V)$. To see this, suppose $\dim_F(V) = n$. Since any $n+1$ vectors in V are linearly dependent, there exists a maximal linearly independent set $\alpha_1, \ldots, \alpha_m$ of elements of W. Indeed, using a variant of the argument used in the proof of Theorem 14.27, we can take α_1 to be any non-zero vector in W, α_2 to be any vector in W not in the subspace spanned by α_1, and so on. Because of Theorem 14.28, this process must halt at some point with $m \leq n$. Now, it must be the case that $\alpha_1, \ldots, \alpha_m$ span W, since otherwise, if $\alpha_{m+1} \in W \setminus \mathrm{Span}_F(\alpha_1, \ldots, \alpha_m)$, the set $\alpha_1, \ldots, \alpha_m, \alpha_{m+1}$ would be a larger set of linearly independent elements of W, contradicting the maximality of $\alpha_1, \ldots, \alpha_m$.

Now suppose that V is finite dimensional, and that W is a subspace of V, and consider the quotient space V/W. It is clear that since V is finite dimensional, V/W is finite dimensional as well. Indeed, if S is a finite set of vectors that spans V, then $\{\alpha + W : \alpha \in S\}$ is a finite set of vectors that spans V/W. It follows from Theorem 14.27 that V/W has a basis, say, $\alpha_1 + W, \ldots, \alpha_\ell + W$. Suppose that $\beta_1, \ldots, \beta_m$ is a basis for W. Then it is straightforward to see (verify) that

$$\alpha_1, \ldots, \alpha_\ell, \ \beta_1, \ldots, \beta_m$$

is a basis for V. Thus, we have proved the following result:

Theorem 14.31 *If V is finite dimensional, and W is a subspace of V, then W and V/W are also finite dimensional, and*

$$\dim_F(V) - \dim_F(W) = \dim_F(V/W).$$

Example 14.32 Note that if V is finite (not just finite dimensional), then F must itself be finite, say, $|F| = q$. If $\dim_F(V) = n$, then $|V| = q^n$. If W is a subspace with $\dim_F(W) = m$, then $|W| = q^m$, and by Theorem 14.31, $\dim_F(V/W) = n - m$, and hence $|V/W| = q^{n-m}$. Just viewing V and W as additive groups, we know that the index of W in V is $[V : W] = |V/W| = |V|/|W| = q^{n-m}$, which agrees with the above calculations. □

The arguments in the two paragraphs preceding Theorem 14.31 also establish the following facts:

Theorem 14.33 *If V is of finite dimension n, then any set of n linearly independent elements of V form a basis for V, and any subset of less than n linearly independent elements of V can be extended to form a basis for V.*

We next consider the relation between the notions of dimension and linear maps.

Theorem 14.34 *If V is of finite dimension n, and V is isomorphic to V', then V' is also of finite dimension n.*

Proof. Let $\rho : V \to V'$ be an F-vector space isomorphism, and let $\alpha_1, \ldots, \alpha_n$ be a basis for V. Then it is easy to see (verify) that $\rho(\alpha_1), \ldots, \rho(\alpha_n)$ is a basis for V'. □

Theorem 14.31, together with Theorems 14.34 and 14.15, immediately imply the following:

Theorem 14.35 *If V is finite dimensional, and $\rho : V \to V'$ is an F-linear map, then* $\mathrm{im}(\rho)$ *is a finite dimensional vector space, and*

$$\dim_F(V) - \dim_F(\ker(\rho)) = \dim_F(\mathrm{im}(\rho)).$$

Intuitively, one way to think of Theorem 14.35 is as a "law of conservation" for dimension: any "dimensionality" going into ρ that is not "lost" to the kernel of ρ must show up in the image of ρ. An immediate corollary of Theorem 14.35 is:

Theorem 14.36 *If $\rho : V \to V'$ is an F-linear map, and if V and V' are finite dimensional with* $\dim_F(V) = \dim_F(V')$*, then we have:*

ρ is injective if and only if ρ surjective.

This last theorem turns out to be extremely useful in a number of settings. Generally, of course, if we have a function $f : A \to B$, injectivity does not imply surjectivity, nor does surjectivity imply injectivity. If A and B are finite sets of equal size, then these implications do indeed hold. Theorem 14.36 gives us another important setting where these implications hold, with finite dimensionality playing the role corresponding to finiteness.

Exercise 14.37 Show that if $V_1, \ldots, V_n$ are finite dimensional vector spaces, then $V_1 \times \cdots \times V_n$ has dimension $\sum_{i=1}^{n} \dim_F(V_i)$. □

Example 14.38 If V is a finite dimensional vector space with subspaces W_1 and W_2, such that $W_1 + W_2 = V$ and $W_1 \cap W_2 = \{0_V\}$, then $\dim_F(V) = \dim_F(W_1) + \dim_F(W_2)$. This follows immediately from Theorems 14.17 and 14.34, along with the previous exercise. □

14.5 Algebras

Examples 14.4 and 14.5 are important examples of vector spaces that are also rings. It is worthwhile investigating such structures in more detail. The concepts discussed here are actually quite simple, and are mainly just a useful way of organizing ideas.

14.5.1 Basic definitions

Definition 14.39 *An F-**algebra** is a ring A, together with a scalar multiplication operation that maps $a \in F$ and $\alpha \in A$ to $a\alpha \in A$, such that*

1. *the addition operation of A, together with the scalar multiplication operation, form an F-vector space, and*
2. *for all $a \in F$ and $\alpha, \beta \in A$, we have $a(\alpha\beta) = (a\alpha)\beta$.*

Example 14.40 As the reader may easily verify, the structures discussed in Examples 14.4 and 14.5 are in fact F-algebras. □

Example 14.41 If A is a ring containing F as a subring, then A forms an F-algebra in the natural way, with scalar multiplication defined in terms of the multiplication operation of the ring. □

Example 14.42 If $A_1, \ldots, A_n$ are F-algebras, then so is the direct product $A_1 \times \cdots \times A_n$, with all operations defined component-wise. □

We now develop an alternative characterization on F-algebras, which will in fact give us a rather simpler view of an F-algebra.

Theorem 14.43 *Let A be an F-algebra. Define the **unit map** $\tau : F \to A$ that sends $a \in F$ to $a \cdot 1_A \in A$. Then τ is a ring homomorphism, and moreover, for all $a \in F$ and $\alpha \in A$, we have $a\alpha = \tau(a)\alpha$.*

Proof. First, using the property (2) of the definition of a vector space, one checks that

$$\tau(a+b) = (a+b)1_A = a1_A + b1_A = \tau(a) + \tau(b).$$

Second, using property (4) of the definition of a vector space, one checks that

$$\tau(1_F) = 1_F 1_A = 1_A.$$

Third, using property (1) of the definition of a vector space and property (2) of the definition of an algebra, one checks that

$$\begin{aligned} \tau(ab) &= (ab)1_A = a(b1_A) = a(b(1_A 1_A)) = a((b1_A)1_A) \\ &= a(1_A(b1_A)) = (a1_A)(b1_A) = \tau(a)\tau(b). \end{aligned}$$

That proves that τ is a ring homomorphism. Finally, one sees that for all $a \in F$ and $\alpha \in A$, we have

$$\tau(a)\alpha = (a1_A)\alpha = a(1_A\alpha) = a\alpha.$$

□

Conversely, we have:

Theorem 14.44 *Let A be an arbitrary ring, and let $\tau : F \to A$ be a ring homomorphism. Define a scalar multiplication as follows:*

$$a\alpha := \tau(a)\alpha \quad \text{for } a \in F \text{ and } \alpha \in A.$$

Then with respect to this scalar multiplication operation, A is an F-algebra, and τ is the associated unit map.

Proof. Exercise. □

From the above results, we see that one could alternatively define an F-algebra as a ring A together with a ring homomorphism $\tau : F \to A$, viewing A as an F-vector space with the scalar multiplication operation that sends $a \in F$ and $\alpha \in A$ to $a\alpha := \tau(a)\alpha$. Moreover, for any F-algebra A, there are only two possibilities: either A is the trivial ring, or the unit map is an embedding of F into A (see Exercise 9.77).

Example 14.45 Let R be any ring of prime characteristic p. The map $\rho : \mathbb{Z} \to R$ that sends $m \in \mathbb{Z}$ to $m1_R$ is a ring homomorphism whose kernel is $p\mathbb{Z}$. Therefore, the map $\bar{\rho} : \mathbb{Z}_p \to R$ that sends $[m \bmod p]$ to $m1_R$ is an injective ring homomorphism from $\mathbb{Z}_p$ into R, and so we can make R into a $\mathbb{Z}_p$-algebra with unit map $\bar{\rho}$. □

A **subalgebra** B of A is a subring of A that is closed under scalar multiplication. Equivalently, a subset B of A is a subalgebra if and only if B is a subring containing the image of F under the unit map.

Example 14.46 Let A be an F-algebra. Any polynomial $g \in F[\mathtt{X}]$ naturally defines a function on A. For $\eta \in A$, let $F[\eta]$ denote the set of elements of A of the form $g(\eta)$, where $g \in F[\mathtt{X}]$. It is easy to see that $F[\eta]$ is a subalgebra of A, and is the smallest subalgebra containing η. Note that if A contains F as a subring, then the notation $F[\eta]$ has the same meaning as in Example 9.65. The F-algebras in Examples 14.4 and 14.5 are of the form $F[\eta]$, with $\eta = \mathtt{X}$ in Example 14.4 and $\eta = [\mathtt{X} \bmod f]$ in Example 14.5. □

Of course, for an F-algebra A, when one speaks of the dimension of A over F, one means the dimension of A viewed as an F-vector space.

14.5.2 Quotient algebras

Let A be an F-algebra, and let I be an ideal in A. Then we claim that as an F-vector space, I is subspace of A. Since I is a ideal, it is certainly an additive subgroup of A, so it suffices to show that I is closed under scalar multiplication. To see this, let $\tau : F \to A$ be the unit map. Then for all $a \in F$ and $\alpha \in I$, we have

$$a\alpha = \tau(a)\alpha \in I,$$

since I is closed under multiplication by all elements of A.

We can form the quotient ring A/I, and define the scalar multiplication

$$a(\alpha + I) := (a\alpha + I) \quad \text{for } a \in F \text{ and } \alpha \in A.$$

Since I is a subspace, this scalar multiplication is unambiguously defined, and it is easy to see that with scalar multiplication so defined, A/I forms an F-algebra, called the **quotient algebra of A modulo I**.

14.5.3 Algebra Homomorphisms

The notions of homomorphisms and isomorphisms extend in the obvious way to F-algebras.

Definition 14.47 *Let A and A' be F-algebras. An F-**algebra homomorphism** from A to A' is a map $\rho : A \to A'$, such that*

- *ρ is a ring homomorphism from A to A', and*
- *for all $a \in F$ and $\alpha \in A$, we have $\rho(a\alpha) = a\rho(\alpha)$.*

*If ρ is bijective, then it is called an F-**algebra isomorphism** of A with A', and if in addition $A = A'$, then it is called an F-**algebra automorphism on** A.*

Thus, an F-algebra homomorphism is precisely a map that is both a ring homomorphism and an F-linear map.

As usual, it is easy to see that if ρ is an isomorphism of A with A' (as F-algebras), then the inverse function ρ^{-1} is an isomorphism of A' with A (again, as F-algebras), and we write $A \cong A'$.

It is easy to see (verify) that a ring homomorphism $\rho : A \to A'$ is an F-algebra homomorphism if and only if $\rho(a1_A) = a1_{A'}$ for all $a \in F$. An important special situation that will arise later is the following: if A and A' are rings containing F as a subring, then both A and A' naturally form F-algebras, and a map $\rho : A \to A'$ is an F-algebra homomorphism if and only if ρ is a ring homomorphism that acts as the identity function of F.

Example 14.48 The complex conjugation map on $\mathbb{C}$ that sends $a+bi$ to $a-bi$, for $a, b \in \mathbb{R}$, is an $\mathbb{R}$-algebra automorphism on $\mathbb{C}$. □

Example 14.49 Let p be a prime, and let F be the field $\mathbb{Z}_p$. If A is an F-algebra, then the map $\rho : A \to A$ that sends $\alpha \in A$ to α^p is an F-algebra homomorphism. The fact that ρ is a ring homomorphism follows from Example 9.74. The fact that ρ is F-linear follows from Theorem 8.72 (Fermat's Little Theorem); indeed, for $a \in F$, we have $(a1_A)^p = a^p 1_A^p = a1_A$. □

Since an F-algebra homomorphism is both a ring homomorphism and an F-vector space homomorphism, all results pertaining to either type of homomorphism apply to an F-algebra homomorphism. In particular, the kernel of an F-algebra homomorphism is both an ideal and a subspace — however, as we have seen, any ideal is a subspace, and so there is nothing special about the kernel of an F-algebra homomorphism.

The reader may easily verify the following observations. First, an F-algebra homomorphism maps subalgebras to subalgebras. Second, Theorems 9.59, 9.60, and 9.61 carry over *verbatim*, substituting the term "F-algebra" for every occurrence of the term "ring."

We next state a very simple, but extremely useful, fact:

Theorem 14.50 *Let $\rho : A \to A'$ be an F-algebra homomorphism. Then for any $g \in F[\mathtt{X}]$ and $\alpha \in A$, we have*

$$\rho(g(\alpha)) = g(\rho(\alpha)).$$

Proof. Let $g = \sum_i g_i \mathtt{X}^i \in F[\mathtt{X}]$. Then we have

$$\rho(\sum_i g_i \alpha^i) = \sum_i g_i \rho(\alpha^i) = \sum_i g_i \rho(\alpha)^i,$$

where the first equality follows from the fact that ρ is an F-linear map, and the second follows from the fact that ρ is a ring homomorphism. □

As a special case of Theorem 14.50, if $A = F[\eta]$ for some $\eta \in A$, then every element of A can be expressed as $g(\eta)$ for some $g \in F[\mathtt{X}]$, and $\rho(g(\eta)) = g(\rho(\eta))$; hence, the action of ρ is completely determined by its action on η.

Example 14.51 Let $A := F[\mathtt{X}]/(f)$ for some monic polynomial $f \in F[\mathtt{X}]$, so that $A = F[\eta]$, where $\eta := [\mathtt{X} \bmod f]$, and let A' be any F-algebra.

Suppose that $\rho : A \to A'$ is an F-algebra homomorphism, and that $\eta' := \rho(\eta)$. The map ρ sends $g(\eta)$ to $g(\eta')$, for $g \in F[\mathtt{X}]$. Also, since $f(\eta) = 0_A$, we have $0_{A'} = \rho(f(\eta)) = f(\eta')$. Thus, η' must be a root of f.

Conversely, suppose that $\eta' \in A'$ is a root of f, i.e., $f(\eta') = 0$. Then the polynomial evaluation map $F[\mathtt{X}] \to A'$ that sends $g \in F[\mathtt{X}]$ to $g(\eta') \in A'$ is a surjective F-algebra homomorphism whose kernel contains f, and this gives rise to the F-algebra homomorphism $\rho : A \to A'$ that sends $g(\eta)$ to $g(\eta')$, for $g \in F[\mathtt{X}]$. One sees that complex conjugation is just a special case of this construction. □

Chapter 15

Matrices over Fields

In this chapter, we discuss the basic definitions and results concerning matrices over an arbitrary field F. Just as in the last chapter, the reader is no doubt familiar with these notions in the context of the real or complex numbers, and the point of this chapter is to generalize these notions to arbitrary fields.

The main goal of this chapter is to discuss "Gaussian elimination," which is an algorithm that allows us to efficiently compute bases for the image and kernel of an F-linear map.

In discussing the complexity of algorithms in this context, we shall treat F as an "abstract data type," so that the running times of algorithms will be stated in terms of the number of arithmetic operations in F. If F is a finite field, such as $\mathbb{Z}_p$, we can immediately translate this into a running time on a RAM (in later chapters, we will discuss other finite fields and efficient algorithms for doing arithmetic in them).

If F is, say, the field of rational numbers, a complete running time analysis would require an additional analysis of the sizes of the numbers that appear in the execution of the algorithm. We shall not attempt such an analysis here (although it can be done, and all the algorithms discussed in this chapter run in polynomial time in the setting of rational numbers, represented as factions in lowest terms). Another possible approach for dealing with rational numbers is to use floating point approximations — while this eliminates the size problem, it creates many new problems because of round-off errors. We shall not address any of these issues here.

15.1 Basic Definitions and Properties

For positive integers m and n, an $m \times n$ **matrix** A over a field F is a rectangular array

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix},$$

where each entry a_{ij} in the array is an element of F; the element a_{ij} is called the (i,j) **entry** of A, which we may denote $A(i,j)$. For $1 \leq i \leq m$, the i**th row** of A is

$$(a_{i1}, \ldots, a_{in}),$$

which we may denote $A(i)$, and for $1 \leq j \leq n$, the j**th column** of A is

$$\begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{pmatrix},$$

which we may denote $A(,j)$. We regard a row of A as a $1 \times n$ matrix, and a column of A as an $m \times 1$ matrix.

The set of all $m \times n$ matrices over F is denoted $F^{m \times n}$. Elements of $F^{1 \times n}$ are called **row vectors (of dimension** n**)** and elements of $F^{m \times 1}$ are called **column vectors (of dimension** m**)**. Elements of $F^{n \times n}$ are called **square matrices (of dimension** n**)**. We do not make a distinction between $F^{1 \times n}$ and $F^{\times n}$; that is, we view standard n-tuples as row vectors. We also do not make a distinction between $F^{1 \times 1}$ and F.

We can define the familiar operations of scalar multiplication, addition, and multiplication on matrices:

- If $A \in F^{m \times n}$ and $c \in F$, then cA is the $m \times n$ matrix whose (i,j) entry is $cA(i,j)$.
- If $A, B \in F^{m \times n}$, then $A + B$ is the $m \times n$ matrix whose (i,j) entry is $A(i,j) + B(i,j)$.
- If $A \in F^{m \times n}$ and $B \in F^{n \times p}$, then AB is the $m \times p$ matrix whose (i,k) entry is

$$\sum_{j=1}^{n} A(i,j)B(j,k).$$

We can also define the difference $A - B := A + (-1_F)B$ of matrices of the same dimension, which is the same as taking the difference of corresponding entries. These operations satisfy the usual properties:

Theorem 15.1 *If $A, B \in F^{m \times n}$, $U, V \in F^{n \times p}$, $Z \in F^{p \times q}$, and $c, d \in F$, then*

1. $c(dA) = (cd)A = d(cA)$,
2. $A + B = B + A$,
3. $c(A + B) = cA + cB$,
4. $(c + d)A = cA + dA$,

5. $(A+B)U = AU + BU$,
6. $A(U+V) = AU + AV$,
7. $c(AU) = (cA)U = A(cU)$,
8. $A(UZ) = (AU)Z$.

Proof. All of these are completely trivial, except the last one which requires just a bit of computation to show that the (i,ℓ) entry of both $A(UZ)$ and $(AU)Z$ is (verify)

$$\sum_{j=1}^{n}\sum_{k=1}^{p} A(i,j)U(j,k)Z(k,\ell).$$

□

Note that while matrix addition is commutative, matrix multiplication in general is not. Some simple but useful facts to keep in mind are the following:

- If $A \in F^{m\times n}$ and $B \in F^{n\times p}$, then the kth column of AB is equal to Av, where v is the kth column of B; also, the ith row of AB is equal to wB, where w is the ith row of A.
- If $A \in F^{m\times n}$ and $u \in F^{1\times m}$, then

$$uA = \sum_{i=1}^{m} u(,i)A(i).$$

In words: uA is a linear combination of the rows of A, with coefficients taken from the corresponding entries of u.

Similarly, if $v \in F^{n\times 1}$, then

$$Av = \sum_{j=1}^{n} v(j)A(,j),$$

i.e., Av is a linear combination of the columns of A, with coefficients taken from the corresponding entries of v.

If $A \in F^{m\times n}$, the the **transpose** of A, denoted $A^\top$, is defined to be the $n \times m$ matrix whose (j,i) entry is $A(i,j)$.

Theorem 15.2 *If $A \in F^{m\times n}$ and $B \in F^{n\times p}$, then $(A^\top)^\top = A$ and $(AB)^\top = B^\top A^\top$.*

Proof. Exercise. □

An $n \times n$ matrix A is called a **diagonal matrix** if $A(i,j) = 0_F$ for $i \neq j$, i.e., the entries off the "main diagonal" of A are all zero. A **scalar matrix** is a diagonal matrix whose diagonal entries are all the same. The scalar matrix I, where $A(i,i) = 1_F$, is called the $n \times n$ **identity matrix**. It is easy to see that if A is an $n \times n$ matrix, then $AI = IA = A$.

Algorithmic issues

For computational purposes, matrices are represented in the obvious way as arrays of elements of F. As remarked at the beginning of this chapter, we shall treat F as an "abstract data type," and not worry about how elements of F are actually represented; in discussing the complexity of algorithms, we shall simply count "operations in F," by which we mean additions, subtractions, multiplications, divisions, and tests for equality. In any real implementation, there will be other costs, such as incrementing counters, etc., which we may safely ignore, as long as their number is at most proportional to the number of operations in F.

The following statements are easy to verify:

- We can multiply an $m \times n$ matrix times a scalar using mn operations in F.
- We can add two $m \times n$ matrices using mn operations in F.
- We can multiply an $m \times n$ matrix and an $n \times p$ matrix using $O(mnp)$ operations in F.

It is also easy to see that given an $m \times m$ matrix A, and a non-negative integer e, we can adapt the repeated squaring algorithm discussed in §3.4 so as to compute A^e using $O(\text{len}(e))$ multiplications of $m \times m$ matrices, and hence $O(\text{len}(e)m^3)$ operations in F.

15.2 Matrices and Linear Maps

For any positive integer m the set of all row vectors $F^{1\times m}$ form a vector space over F in the natural way. If A is an $m \times n$ matrix over F, then the map that sends $v \in F^{1\times m}$ to $vA \in F^{1\times n}$ is easily seen to be an F-linear map. Also, the map that sends $w \in F^{n\times 1}$ to $Aw \in F^{m\times 1}$ is also an F-linear map. Thus, the matrix A defines in a natural way two different linear maps, one defined in terms of multiplying a row vector on the right by A, and the other in terms multiplying a column vector on the left by A.

With the above interpretations as a linear map, the definition of matrix multiplication makes a bit more sense. Let $A \in F^{m\times n}$ and $B \in F^{n\times p}$, and consider the product matrix $C = AB$. Let $\sigma_A, \rho_B, \sigma_C$ be the maps defined by by multiplication on the right by A, B, C, and let τ_A, τ_B, τ_C be the maps defined by multiplication on the left by A, B, C. Then it is easy to see (verify) that $\sigma_C = \sigma_B \circ \sigma_A$ and $\tau_C = \tau_A \circ \tau_B$, where $f \circ g$ denotes function composition, i.e., $(f \circ g)(x) = f(g(x))$.

So we have seen how matrix/vector multiplication defines a linear map on finite dimensional vector spaces. Conversely, we shall now see that the action of any linear map on finite dimensional vector spaces can be viewed as a matrix/vector multiplication.

Let V be an F-vector space of finite dimension m, and let $\mathcal{A} = (\alpha_1, \ldots, \alpha_m)$ be a basis for V. In this setting, the ordering of the basis elements is important, and so we refer to $\mathcal{A}$ as an **ordered basis**. Now, $\mathcal{A}$ defines a canonical F-vector space isomorphism ϵ that sends

$(a_1, \ldots, a_m) \in F^{1\times m}$ to $a_1\alpha_1 + \cdots + a_m\alpha_m \in V$. Thus, elements of V can be represented concretely as elements of $F^{1\times m}$; however, this representation depends on the choice $\mathcal{A}$ of the ordered basis. The vector $\epsilon^{-1}(\alpha)$ is called the **coordinate vector of α (with respect to $\mathcal{A}$)**.

Let W be an F-vector space of finite dimension n, and let $\mathcal{B} = (\beta_1, \ldots, \beta_n)$ be an ordered basis for W. Just as in the previous paragraph, $\mathcal{B}$ defines a canonical F-vector space isomorphism $\delta : F^{1\times n} \to W$.

Now let $\rho : V \to W$ be an arbitrary F-linear map. For any $\alpha \in V$, if $\alpha = \epsilon(a_1, \ldots, a_m)$, then because ρ is F-linear, we have

$$\rho(\alpha) = \sum_{i=1}^{m} \rho(a_i\alpha_i) = \sum_{i=1}^{m} a_i\rho(\alpha_i).$$

Thus, the action of ρ on V is completely determined by its action on the α_i's.

Let us now define an $m \times n$ matrix D whose ith row, for $1 \le i \le m$, is defined to be $\delta^{-1}(\rho(\alpha_i))$, that is, the coordinate vector of $\rho(\alpha_i)$ with respect to the ordered basis $\mathcal{B}$. With D defined in this way, then for any $\alpha \in V$ we have

$$\delta^{-1}(\rho(\alpha)) = \epsilon^{-1}(\alpha)D.$$

In words: if we multiply the coordinate vector of α on the right by D, we get the coordinate vector of $\rho(\alpha)$.

A special case of the above is when $V = F^{1\times m}$ and $W = F^{1\times n}$, and $\mathcal{A}$ and $\mathcal{B}$ are the **standard bases** for V and W, i.e., for $1 \le i \le m$, the ith vector of $\mathcal{A}$ has a 1 in position i and is zero everywhere else, and similarly for $\mathcal{B}$. In this case, the ith row of the matrix D is just the value of ρ applied to the ith vector in $\mathcal{A}$.

To summarize, we see that an F-linear map ρ from a finite dimensional vector space V to a finite dimensional vector space W, together with particular ordered bases for V and W, uniquely determine a matrix D such that the action of multiplication on the right by D implements the action of ρ with respect to the given ordered bases. There may be many ordered bases for V and W to choose from, and different choices will in general lead to different matrices. In any case, from a computational perspective, the matrix D gives us an efficient way to compute the map ρ, assuming elements of V and W are represented as coordinate vectors with respect to the given ordered basis.

Of course, if one prefers, by simply transposing everything, one can equally well represent the action of ρ in terms of the action of multiplication of a column vector on the left by a matrix.

Exercise 15.3 Let F be a finite field, and let A be a non-zero $m \times n$ matrix over F. Suppose one chooses a vector $v \in F^{1\times m}$ at random. Show that the probability that vA is the zero vector is at most $1/|F|$. □

Exercise 15.4 Design and analyze a probabilistic algorithm that takes as input three $m\times m$ matrices A, B, C over a finite field F, along with an error parameter $0 < \epsilon < 1$. The

algorithm should use $O(m^2\lceil t\rceil)$ operations in F, where $t := \log(1/\epsilon)/\log|F|$. The algorithm should output either "yes" or "no" so that the following holds:

- if $C = AB$, then the algorithm should always output "yes";
- if $C \neq AB$, then the algorithm should output "no" with probability at least $1 - \epsilon$.

□

15.3 The Inverse of a Matrix

Let $A \in F^{n\times n}$ be a square matrix. We call a matrix $X \in F^{n\times n}$ an **inverse** of A if $XA = AX = I$, where I is the $n \times n$ identity matrix.

It is easy to see that if A has an inverse, then the inverse is unique: if X and Y were inverses, then multiplying the equation $I = AX$ on the left by Y, we obtain $Y = Y(AX) = (YA)X = IX = X$.

Because the inverse of A is uniquely determined, we denote it by A^{-1}. If A has an inverse, we say that A is **invertible nonsingular**. If A is not invertible, it is sometimes called **singular**. We will use the terms "invertible" and "not invertible."

If A and B are invertible $n \times n$ matrices, then so is there product: in fact, it is easy to see that $(AB)^{-1} = B^{-1}A^{-1}$ (verify).

It is also easy to see that A is invertible if and only if the transposed matrix $A^\top$ is invertible, in which case $(A^\top)^{-1} = (A^{-1})^\top$. Indeed, $AX = I = XA$ holds if and only if $X^\top A^\top = I = A^\top X^\top$

Let us call X a **left inverse** of A if $XA = I$, and let us call Y a **right inverse** of A if $AY = I$.

It is easy to see that if A has both a left inverse X and a right inverse Y, then we must have $X = Y$, from which it follows that $X = A^{-1}$. To see this, again, multiply the equation $I = AY$ on the left by X, obtaining $X = X(AY) = (XA)Y = IY = Y$.

One question that remains, the answer to which is not trivially self evident from the definitions, is whether or not the existence of either a left or right inverse implies the existence of an inverse. The answer is yes, and we can argue this as follows. Let A be the given square matrix, and let ρ be the F-linear map from $F^{1\times n}$ to $F^{1\times n}$ that sends v to vA. If A has a left inverse X, so $I = XA$, then the map ρ is surjective: indeed, for any $v \in F^{1\times n}$, $v = vI = vXA = \rho(vX)$. If A has a right inverse Y, so that $I = AY$, then the map ρ is injective: indeed, if $\rho(v) = 0^{1\times n}$, then $v = vI = vAY = \rho(v)Y = 0^{1\times n}$. Now, by Theorem 14.36, the map ρ is a bijection if and only if it is either surjective or injective. So if A has either a left or a right inverse, the map ρ is a vector space isomorphism, and hence its inverse ρ^{-1} is also a vector space isomorphism. If we let Z be the matrix representing ρ^{-1} with respect to the standard basis for $F^{1\times n}$, then ZA is the matrix representing $\rho\circ\rho^{-1}$, and AZ is the matrix representing $\rho^{-1}\circ\rho$. Since both $\rho\circ\rho^{-1}$ and $\rho^{-1}\circ\rho$ are the identity function, it must be the case that $ZA = AZ = I$.

So we have shown that if A has either a left or right inverse, then the corresponding map ρ is an isomorphism, which implies that A is invertible. Conversely, if A has an inverse, then it is clear that the corresponding map ρ is a vector space isomorphism.

The above discussion also reveals the following important fact:

Theorem 15.5 *An square matrix invertible if and only if its rows are linearly independent if and only if its columns are linearly independent.*

Proof. As we saw above, A has an inverse if and only if the map ρ, defined by multiplication on the right by A, is bijective, which holds if and only if ρ is injective, which holds if and only if the the rows of A are linearly independent.

That proves the statement that the inverse exists if and only if the the rows are linearly independent. The corresponding statement about columns follows from the statement about rows, applied to the transposed matrix $A^\top$. □

Exercise 15.6 Show that if A and B are two square matrices such that their product AB is invertible, then both A and B themselves must be invertible. □

15.4 Gaussian Elimination

A matrix $B \in F^{m\times n}$ is said to be in **reduced row echelon form** if there exists a sequence of integers $(p_1, \ldots, p_r)$, with $0 \le r \le m$ and $1 \le p_1 < p_2 < \cdots < p_r \le n$, such that the following holds:

- for $1 \le i \le r$, all of the entries in row i of B to the left of entry (i, p_i) are zero, i.e., $B(i, j) = 0$ for $1 \le j < p_i$;
- for $1 \le i \le r$, all of the entries in B in column p_i of B above entry (i, p_i) are zero, i.e., $B(i', p_i) = 0$ for $1 \le i' < i$;
- $B(i, p_i) = 1$;
- all entries in rows $r + 1, \ldots, m$ of B are zero, i.e., $B(i, j) = 0$ for $r < i \le m$ and $1 \le j \le n$.

It is easy to see that if B is in reduced row echelon form, the sequence $(p_1, \ldots, p_r)$ above is uniquely determined, and we call it the **pivot sequence** of B. Several further remarks are in order:

- All of the entries of B are completely determined by the pivot sequence, except for the entries (i, j) with $1 \le i \le r$ and $j > i$ with $j \notin \{p_{i+1}, \ldots, p_r\}$, which may be arbitrary.
- If B is an $n \times n$ matrix in reduced row echelon form whose pivot sequence is of length n, then B must be the $n \times n$ identity matrix.

- We allow for an empty pivot sequence, i.e., $r = 0$, which will be the case precisely when $B = 0^{m \times n}$.

Example 15.7 The following 4×6 matrix matrix B over the rational numbers is in reduced row echelon form:

$$B = \begin{pmatrix} 0 & 1 & -2 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The pivot sequence of B is $(2, 4, 5)$. Notice that the first three rows of B are linearly independent, that columns 2, 4, and 5 are linearly independent, and that all of other columns of B are linear combinations of columns 2, 4, and 5. Indeed, if we truncate the pivot columns to their first three rows, we get the 3×3 identity matrix. □

Generalizing the previous example, if a matrix is in reduced row echelon form, it is easy to deduce the following properties, which turn out to be quite useful:

Theorem 15.8 *If B is a matrix in reduced row echelon form with pivot sequence $(p_1, \ldots, p_r)$, then*

1. *rows $1, 2, \ldots, r$ of B are linearly independent;*
2. *columns $p_1, \ldots, p_r$ of B are linearly independent, and all other columns of B can be expressed as linear combinations of columns $p_1, \ldots, p_r$.*

Proof. Exercise — just look at the matrix! □

Gaussian elimination is an algorithm that transforms an arbitrary $m \times n$ matrix A into a $m \times n$ matrix B, where B is a matrix in reduced row echelon form obtained from A by a sequence of **elementary row operations**. There are three types of elementary row operations:

Type I: swap two rows,

Type II: multiply a row by a scalar,

Type III: add a scalar multiple of one row to a different row.

The application of any specific elementary row operation to an $m \times n$ matrix C can be affected by multiplying C on the left by a suitable $m \times m$ matrix M. Indeed, the matrix M corresponding to a particular elementary row operation is simply the matrix obtained by applied the same elementary row operation to the $m \times m$ identity matrix. It is easy to see that for any elementary row operation, the corresponding matrix M is invertible.

We now describe the basic version of Gaussian elimination. The input is an $m \times n$ matrix A. The algorithm works with a copy B of A (which we do not need, if the original matrix A is not needed afterwards).

```
1.  B ← A, r ← 0
2.  for j ← 1 to n do
3.      ℓ ← 0, i ← r
4.      while ℓ = 0 and i ≤ m do
5.          i ← i + 1
6.          if B(i, j) ≠ 0 then ℓ ← i
7.      if ℓ ≠ 0 then
8.          r ← r + 1
9.          swap rows B(r) and B(ℓ)
            — B(r, j) is non-zero
            — now make B(r, j) one and clear all entries above and below B(r, j)
10.         B(r) ← B(r, j)^{-1} B(r)
11.         for i ← 1 to m do
12.             if i ≠ r then
13.                 B(i) ← B(i) − B(i, j)B(r)
14. output B
```

Note that the only steps in the algorithm where B is actually modified are at steps 9, 10, and 13, where we perform (respectively) Type I, II, and III elementary row operations. We leave it to the reader to verify that the above algorithm indeed transforms A into a matrix B in reduced row echelon form. To do this, one might make use of the following "loop invariant":

> after the jth iteration of the main loop (for $0 \leq j \leq n$), the first j columns of B are in reduced row echelon form with a pivot sequence whose length is equal to the current value of r.

As for the complexity of the algorithm, it is easy to see that it performs $O(mn)$ elementary row operations, each of which takes $O(n)$ operations in F, so a total of $O(mn^2)$ operations in F.

As discussed above, the application the eth elementary row operation in the above algorithm can be thought of as multiplying the current value of the matrix B by a particular invertible $m \times m$ matrix M_e. If the algorithm performs a total of t such elementary row operations, the final, output value of B satisfies the equation

$$B = MA,$$

where

$$M = \prod_{e=1}^{t} M_e.$$

Since the product of invertible matrices is also invertible, we see that M itself is invertible.

The above algorithm does not compute the matrix M, but it can be easily modified to do so. The resulting algorithm, which we call **extended Gaussian elimination**, is the

same as plain Gaussian elimination, except that we initialize the matrix M to be the $m \times m$ identity matrix, and we add the following steps:

- Just before step 9, we add the step: swap rows $M(r)$ and $M(\ell)$.
- Just before step 10, we add the step: $M(r) \leftarrow B(r,j)^{-1}M(r)$.
- Just before step 13, we add the step: $M(i) \leftarrow M(i) - B(i,j)M(r)$.

At the end of the algorithm we output M in addition to B.

So we simply perform the same elementary row operations on M that we perform on B. The reader may verify that the above algorithm is correct, and that it uses $O(mn(m+n))$ operations in F.

Exercise 15.9 Given a matrix $B \in F^{m \times n}$ in reduced row echelon form, show how to compute its pivot sequence using $O(\min\{m,n\})$ operations in F. □

15.5 Applications of Gaussian Elimination

Throughout this section, A is an arbitrary $m \times n$ matrix over F, and $MA = B$, where M is an invertible $m \times m$ matrix, and B is in reduced row echelon form with pivot sequence $(p_1, \ldots, p_r)$. This is precisely the information produced by the extended Gaussian elimination algorithm, given A as as input (the pivot sequence can easily be "read" directly from B — see Exercise 15.9).

Let $V := F^{1 \times m}$, $W := F^{1 \times n}$, and $\rho : V \to W$ be the F-linear map that sends $v \in V$ to $vA \in W$.

Computing the image and kernel

Consider first the **row space** of A, that is, the vector space spanned by the rows of A, or equivalently, the image of ρ in W.

We claim that the row space of A is the same as the row space of B. To see this, note that for any $v \in V$, since $B = MA$, we have $vB = v(MA) = (vM)A$, and so the row space of B is contained in the row space of A. For the other containment, note that since M is invertible, we can write $A = M^{-1}B$, and apply the same argument.

Further, note that row space of B, and hence that of A, clearly has dimension r. Indeed, as stated in Theorem 15.8, the first r rows of B form a basis for the row space of B.

Consider next the kernel of ρ, or what we might call the **row null space** of A. We claim that the last $m - r$ rows of M form a basis for $\ker(\rho)$. Clearly, just from the fact that $MA = B$ and the fact that that the last $m - r$ rows of B are zero, it follows that the last $m - r$ rows of M are contained in $\ker(\rho)$. Furthermore, as M is invertible, its rows are linearly independent, and so it suffices to show that the last $m - r$ rows of M span the entire kernel. Since M is invertible, its rows are linearly independent, and hence form a basis for V. Now, suppose there were a vector $v \in \ker(\rho)$ which was not in the subspace spanned by

the last $m - r$ rows of M. This means that $v = a_1 M(1) + \cdots + a_m M(m)$, where $a_i \neq 0$ for some $1 \leq i \leq r$. Setting $\tilde{v} = (a_1, \ldots, a_m)$, we see that $v = \tilde{v}M$, and so

$$\rho(v) = vA = (\tilde{v}M)A = \tilde{v}(MA) = \tilde{v}B,$$

and from the fact that the first r rows of B are linearly independent and the last $m-r$ rows of B are zero, we see that wB is not the zero vector (and because $\tilde{v}$ has a nonzero entry in one its first r positions). We have derived a contradiction, and hence may conclude that the last $m - r$ rows of M span $\ker(\rho)$.

Finally, note that if $m = n$, then A is invertible if and only if its row space has dimension m, which holds if and only if $r = m$, and in the latter case, B will be the identity matrix, and hence M is the inverse of A.

Let us summarize the above discussion:

- *The first r rows of B form a basis for the row space of A (i.e., the image of ρ).*
- *The last $m - r$ rows of M form a basis for the row null space of A (i.e., the kernel of ρ).*
- *If $m = n$, then A is invertible (i.e., ρ is an isomorphism) if and only if $r = m$, in which case M is the inverse of A (i.e., the matrix representing ρ^{-1}).*

So we see that from the output of the extended Gaussian elimination algorithm, we can simply "read off" bases for both the image and the kernel, as well as the inverse (if it exists), of a linear map represented as a matrix with respect to some ordered bases. Also note that this procedure provides a more concrete version of the statement of Theorem 14.31.

Solving linear systems of equations

Suppose that in addition to the matrix A, we are given $w \in W$, and want to find a solution v (or perhaps describe all solutions v), to the equation

$$vA = w. \tag{15.1}$$

Equivalently, we can phrase the problem as finding an element (or describing all elements) of the set $\rho^{-1}(w)$.

Now, if there exists a solution at all, say $v \in V$, then since $\rho(v) = \rho(\tilde{v})$ if and only if $v \equiv \tilde{v} \pmod{\ker(\rho)}$, it follows that the set of all solutions to (15.1) is equal to the coset $v + \ker(\rho)$. Thus, given a basis for $\ker(\rho)$ and any solution v to (15.1), we have a complete and concise description of the set of solutions to (15.1).

As we have discussed above, the last $m - r$ rows of M give us a basis for $\ker(\rho)$, so it suffices to determine if $w \in \mathrm{im}(\rho)$, and if so, determine a single pre-image v of w.

Also as we discussed, $\mathrm{im}(\rho)$, i.e., the row space of A, is equal to the row space of B, and because of the special form of B, we can quickly and easily determine if the given w is in the row space of B, as follows. Now, w is in the row space of B iff there exists a vector $\bar{v} \in V$

such that $\bar{v}B = w$. We may as well assume that all but the first r entries of $\bar{v}$ are zero. Moreover, $\bar{v}B = w$ implies that for $1 \leq i \leq r$, the ith entry if $\bar{v}$ is equal to p_ith entry of w. Thus, the vector $\bar{v}$, if it exists, is completely determined by the entries of w at positions $p_1, \ldots, p_r$. We can construct $\bar{v}$ satisfying these conditions, and then test if $\bar{v}B = w$. If not, then we may conclude that (15.1) has no solutions; otherwise, setting $v := \bar{v}M$, we see that $vA = (\bar{v}M)A = \bar{v}(MA) = \bar{v}B = w$, and so v is a solution to (15.1).

One easily verifies that if we implement the above procedure as an algorithm, the work done in addition to running the extended Gaussian elimination algorithm amounts to $O(m(n+m))$ operations in F.

A special case of the above procedure is when $m = n$ and A is invertible, in which case (15.1) has a unique solution, namely, $v := wM$, since in this case, $M = A^{-1}$.

The rank of a matrix

Define the **row rank** of A to be the dimension of its row space, i.e., $\dim_F(\mathrm{im}(\rho))$, and define the **column rank** of A to be the dimension of its **column space**, i.e., the space spanned by the column of A.

Now, the column space A may not be the same as the column space of B, but from the relation $B = MA$, and the fact that M is invertible, it easily follows that these two subspaces are isomorphic, and hence have the same dimension. Moreover, by Theorem 15.8, the column rank of B is r, which is the same as the row rank of A.

So we may conclude: *The column rank and row rank of A are the same.*

Because of this, we define the **rank** of a matrix to be the common value of its row and column rank.

The orthogonal compliment of a subspace

So as to give equal treatment to rows and columns, one can also define the **column null space** of A to be the kernel of the linear map defined by multiplication on the left by A. By applying results above to the transpose of A, we see that the column null space of A has dimension $n - r$, where r is the rank of A.

Let $U \subset W$ denote the row space of A, and let $\bar{U} \subset W$ denote the set of all vectors $\bar{u} \in W$ whose transpose $\bar{u}^\top$ belong to the column null space of A. Now, U is a subspace of W of dimension r and $\bar{U}$ is a subspace of W of dimension $n - r$.

Moreover, if $U \cap \bar{U} = \{0_V\}$, then by Theorem 14.17 we have an isomorphism of $U \times \bar{U}$ with $U + \bar{U}$, and since $U \times \bar{U}$ has dimension n, it must be the case that $U + \bar{U} = W$. It follows that every element of W can be expressed uniquely as $u + \bar{u}$, where $u \in U$ and $\bar{u} \in \bar{U}$.

Now, all of the conclusions in the previous paragraph hinged on the assumption that $U \cap \bar{U} = \{0_V\}$. The space $\bar{U}$ consists precisely of all vectors $\bar{u} \in W$ which are "orthogonal" to all vectors $u \in U$, in the sense that the "inner product" $u\bar{u}^\top$ is zero.. For this reason, $\bar{U}$ is sometimes called the "orthogonal compliment of U." The condition $U \cap \bar{U} = \{0_V\}$

is equivalent to saying that U contains no non-zero "self-orthogonal vectors" u such that $uu^\top = 0_F$. If F is the field of real numbers, then of course there are no self-orthogonal vectors, since $uu^\top$ is the sum of the squares of the entries of u. However, for other fields, there may very well be self-orthogonal vectors. As an example, if $F = \mathbb{Z}_2$, then any vector u with an even number of 1-entries is self orthogonal.

So we see that while much of the theory of vector spaces and matrices carries over without change from familiar ground fields, like the real numbers, to arbitrary ground fields F, not everything does. In particular, the usual decomposition of a vector space into a subspace and its orthogonal compliment breaks down, as does any other procedure that relies on properties specific to "inner product spaces."

Exercise 15.10 With A and B as above, show that the column null space of A is the same as the column null space of B. □

Exercise 15.11 Show how to compute a basis for the column null space of A using $O(r(n-r))$ operations in F, given A and B as above. □

Exercise 15.12 With A and B as above, show that the matrix B is uniquely determined by A; more precisely, show that if $M'A = B'$, where M' is an invertible $m \times m$ matrix, and B' is in reduced row echelon form, then $B' = B$. □

15.6 Notes

While a trivial application of the defining formulas yields a simple algorithm for multiplying two $m \times m$ matrices over a field F that uses $O(m^3)$ operations in F, this algorithm is not the best asymptotically speaking. The currently fastest algorithm for this problem, due to Coppersmith and Winograd [22], uses $O(m^\omega)$ operations in F, where $\omega < 2.376$. We note, however, that the good old $O(m^3)$ algorithm is still the only one anyone uses in any practical setting.

Chapter 16

Subexponential-time Algorithms for Discrete Logarithms and Factoring

16.1 Smooth Numbers

The key concept in all subexponential-time algorithms for discrete logarithms and factoring is that of a **smooth number**. If y is a non-negative real number, and m is a positive integer, then we say that m is y**-smooth** if all prime divisors of m are at most y.

For $0 \leq y \leq x$, let us define $\Psi(y, x)$ to be the number of y-smooth integers up to x. The following theorem gives us a lower bound on $\Psi(y, x)$, which will be crucial in the analysis of our discrete logarithm and factoring algorithms.

Theorem 16.1 *Let y be a function of x such that*

$$\frac{y}{\log x} \to \infty \quad \textit{and} \quad u := \frac{\log x}{\log y} \to \infty$$

as $x \to \infty$. Then

$$\Psi(y, x) \geq x \cdot \exp[(-1 + o(1))u \log\log x].$$

Proof. Let us write $u = \lfloor u \rfloor + \delta$, where $0 \leq \delta < 1$. Let us split the primes up to y into two sets: the set V "very small" primes that are at most $y^\delta/2$, and the other primes W that are greater than $y^\delta/2$ but at most y. To simplify matters, let us also include the integer 1 in the set V.

By Theorem 5.11 (Bertrand's Postulate), there exists a constant $C > 0$ such that $|W| \geq Cy/\log y$ for sufficiently large y. By the assumption that $y/\log x \to \infty$ as $x \to \infty$, it follows that that $|W| \geq 2\lfloor u \rfloor$ for sufficiently large x.

To derive the lower bound, we shall count those integers that can be built up by multiplying together $\lfloor u \rfloor$ distinct elements of W, together with one element of V. These products

are clearly distinct, y-smooth numbers, and each is bounded by x, since each is at most $y^{\lfloor u \rfloor} y^{\delta} = y^u = x$.

If S denotes the set of all of these products, then for x sufficiently large, we have

$$\begin{aligned} |S| &= \binom{|W|}{\lfloor u \rfloor} \cdot |V| \\ &= \frac{|W|(|W|-1)\cdots(|W|-\lfloor u \rfloor+1)}{\lfloor u \rfloor!} \cdot |V| \\ &\geq \left(\frac{|W|}{2u}\right)^{\lfloor u \rfloor} \cdot |V| \\ &\geq \left(\frac{Cy}{2u\log y}\right)^{\lfloor u \rfloor} \cdot |V| \\ &= \left(\frac{Cy}{2\log x}\right)^{u-\delta} \cdot |V|. \end{aligned}$$

Taking logarithms, we have

$$\begin{aligned} \log|S| &\geq (u-\delta)(\log y - \log\log x + \log(C/2)) + \log|V| \\ &= \log x - u\log\log x + (\log|V| - \delta\log y) + O(u + \log\log x). \end{aligned} \tag{16.1}$$

To prove the theorem, it suffices to show that

$$\log|S| \geq \log x - (1+o(1))u\log\log x.$$

Under our assumption that $u \to \infty$, the term $O(u + \log\log x)$ in (16.1) is $o(u\log\log x)$, and so it will suffice to show that the term $\log|V| - \delta\log y$ is also $o(u\log\log x)$. But by Theorem 5.2 (Chebyshev's Theorem), for some positive constant D, we have

$$Dy^{\delta}/\log y \leq |V| \leq y^{\delta},$$

and taking logarithms, and again using the fact that $u \to \infty$, we have

$$\log|V| - \delta\log y = O(\log\log y) = o(u\log\log x).$$

□

16.2 An Algorithm for Discrete Logarithms

We now present a probabilistic, subexponential-time algorithm for computing discrete logarithms. The input to the algorithm is p, q, γ, α, where p and q are primes, with $q \mid (p-1)$, γ is an element of $\mathbb{Z}_p^*$ generating a subgroup G of order q, and $\alpha \in G$.

We shall make the simplifying assumption that $q^2 \nmid (p-1)$, which is equivalent to saying that $q \nmid m := (p-1)/q$. This assumption greatly simplifies the design and analysis of the

algorithm, and moreover, for cryptographic applications, this assumption is almost always satisfied. We note, however, that this assumption may be lifted, but the algorithms in this case are significantly more complicated.

Let G' be the subgroup of $\mathbb{Z}_p^*$ of order m. Our assumption that $q \nmid m$ implies that $G \cap G' = \{1\}$, since the order of any element in the intersection must divide both q and m, and so the only possibility is that the order is 1. Therefore, the map $\rho : G \times G' \to \mathbb{Z}_p^*$ that sends (β, δ) to $\beta\delta$ is injective (Theorem 8.55), and since $|\mathbb{Z}_p^*| = qm$, it must be surjective as well.

We shall use this fact in the following way: if β is chosen uniformly at random from G, and δ is chosen uniformly at random from G' (and independent of β), then $\beta\delta$ is uniformly distributed over $\mathbb{Z}_p^*$. Furthermore, since G' is the image of the q-power map on $\mathbb{Z}_p^*$, we may generate a random $\delta \in G'$ simply by choosing $\hat{\delta} \in \mathbb{Z}_p^*$ at random, and setting $\delta := \hat{\delta}^q$.

The discrete logarithm algorithm uses a "smoothness parameter" y, whose choice will be discussed below when we analyze the running time of the algorithm; for now, we only assume that $y < p$. Let $p_1, \ldots, p_k$ be an enumeration of the primes up to y. Let $\pi_i := [p_i \bmod p] \in \mathbb{Z}_p^*$ for $i = 1, \ldots, k$. Let us write $\bar{a}$ to denote the image of an integer a in $\mathbb{Z}_q$, and similarly, for a vector v with integer entries, $\bar{v}$ denotes its image as a vector with entries in $\mathbb{Z}_q$.

The algorithm has two stages.

In the first stage, we find relations of the form

$$\gamma^{r_i} \alpha^{s_i} \delta_i = \pi_1^{e_{i1}} \cdots \pi_k^{e_{ik}}, \tag{16.2}$$

for integers $r_i, s_i, e_{i1}, \ldots, e_{ik}$, and $\delta_i \in G'$, and $i = 1, \ldots, k+1$.

We obtain one such relation by a randomized search, as follows: we choose $r_i, s_i \in \{0, \ldots, q-1\}$ at random, as well as $\hat{\delta}_i \in \mathbb{Z}_p^*$ at random; we then compute $\delta_i := \hat{\delta}_i^q$, $\beta_i := \gamma^{r_i} \alpha^{s_i}$, and $m_i := \mathrm{rep}(\beta_i \delta_i)$. Now, the value β_i is uniformly distributed over G, while δ_i is uniformly distributed over G'; therefore, the product $\beta_i \delta_i$ is uniformly distributed over $\mathbb{Z}_p^*$, and hence m_i is uniformly distributed over $\{1, \ldots, p-1\}$. Next, we simply try to factor m_i by trial division, trying all the primes $p_1, \ldots, p_k$ up to y. If we are lucky, we completely factor m_i in this way, obtaining a factorization

$$m_i = p_1^{e_{i1}} \cdots p_k^{e_{ik}},$$

for some exponents $e_{i1}, \ldots, e_{ik}$, and we get the relation (16.2). If we are unlucky, then we simply try (and try again) until we are lucky.

For $i = 1, \ldots, k+1$, let $v_i := (e_{i1}, \ldots, e_{ik}) \in \mathbb{Z}^{\times k}$. The vectors $\bar{v}_1, \ldots, \bar{v}_{k+1} \in \mathbb{Z}_q^{\times k}$ must be linearly dependent, and the second stage uses Gaussian elimination over the field $\mathbb{Z}_q$ (see §15.4) to find integers $c_1, \ldots, c_k \in \{0, \ldots, q-1\}$, not all zero, such that $\bar{c}_1 \bar{v}_1 + \cdots + \bar{c}_{k+1} \bar{v}_{k+1} = 0$. Let

$$(e_1, \ldots, e_k) := c_1 v_1 + \cdots c_{k+1} v_{k+1} \in \mathbb{Z}^{\times k}.$$

Raising each equation (16.2) to the power c_i, and multiplying them all together, we obtain

$$\gamma^r \alpha^s \delta^t = \pi_1^{e_1} \cdots \pi_k^{e_k},$$

where

$$r := \sum_{i=1}^{k+1} c_i r_i, \;\; s := \sum_{i=1}^{k+1} c_i s_i, \;\; t := \sum_{i=1}^{k+1} c_i.$$

Now, $\delta \in G'$, and since each e_i is a multiple of q, we also have $\pi_i^{e_i} \in G'$ for $i = 1, \ldots, k$. It follows that $\gamma^r \alpha^s \in G'$. But since $\gamma^r \alpha^s \in G$ as well, and $G \cap G' = \{1\}$, it follows that $\gamma^r \alpha^s = 1$. If we are lucky (and we will be with overwhelming probability, as we discuss below), we will have $s \not\equiv 0 \pmod{q}$, in which case, we can compute a multiplicative inverse s' of s modulo q, obtaining

$$\alpha = \gamma^{-rs'},$$

and hence $-rs' \bmod q$ is the discrete logarithm of α to the base γ. If we are very unlucky, we will have $s \equiv 0 \pmod{q}$, at which point the algorithm simply quits, reporting "failure."

Here is the entire algorithm:

Algorithm SEDL

$i \leftarrow 0$
repeat
    $i \leftarrow i + 1$
    repeat
        choose $r_i, s_i \in \{0, \ldots, q-1\}$ at random
        choose $\hat{\delta}_i \in \mathbb{Z}_p^*$ at random
        $\beta_i \leftarrow \gamma^{r_i} \alpha^{s_i}, \quad \delta_i \leftarrow \hat{\delta}_i^q, \quad m_i \leftarrow \mathrm{rep}(\beta_i \delta_i)$
        test if m_i is y-smooth (trial division)
    until $m_i = p_1^{e_{i1}} \cdots p_k^{e_{ik}}$ for some integers $e_{i1}, \ldots, e_{ik}$
until $i = k + 1$

set $v_i \leftarrow (e_{i1}, \ldots, e_{ik}) \in \mathbb{Z}^{\times k}$ for $i = 1, \ldots, k+1$

apply Gaussian elimination to find integers $c_1, \ldots, c_{k+1} \in \{0, \ldots, q-1\}$, not all zero,
    such that $\bar{c}_1 \bar{v}_1 + \cdots + \bar{c}_{k+1} \bar{v}_{k+1} = 0$.

$r \leftarrow \sum_{i=1}^{k+1} c_i r_i, \quad s \leftarrow \sum_{i=1}^{k+1} c_i s_i$

if $s \equiv 0 \pmod{q}$ then
    output "failure"
else
    compute a multiplicative inverse s' of s modulo q
    output $-rs' \bmod q$

As already argued above, if algorithm SEDL does not output "failure," then its output is indeed the discrete logarithm of α to the base γ. There remain three questions to answer:

1. What is the expected running time of algorithm SEDL?
2. How should the smoothness parameter y be chosen so as to minimize the expected running time?
3. What is the probability that algorithm SEDL outputs "failure"?

Let us address these questions in turn. As for the expected running time, let σ be the probability that a random element of $\{1, \ldots, p-1\}$ is y-smooth. Then the expected number of attempts needed to produce a single relation is σ^{-1}, and so the expected number of attempts to produce $k+1$ relations is $(k+1)\sigma^{-1}$. In each attempt, we perform trial division using $p_1, \ldots, p_k$, along with a few other minor computations, leading to a total expected running time in stage 1 of $k^2\sigma^{-1} \cdot (\log p)^{O(1)}$. The running time in stage 2 is dominated by that of the Gaussian elimination step, which takes time $k^3 \cdot (\log p)^{O(1)}$. Thus, if T is the total running time of the algorithm, then we have

$$\mathsf{E}[T] \leq (k^2\sigma^{-1} + k^3) \cdot (\log p)^{O(1)}. \tag{16.3}$$

Let us assume for the moment that

$$y = \exp[(\log p)^{\lambda + o(1)}] \tag{16.4}$$

for some constant λ with $0 < \lambda < 1$. Our final choice of y will indeed satisfy this assumption. Consider the probability σ. We have

$$\sigma = \Psi(y, p-1)/(p-1) = \Psi(y, p)/(p-1) \geq \Psi(y, p)/p,$$

where for the second equality we use the assumption that $y < p$, so p is not y-smooth. With our assumption (16.4), we may apply Theorem 16.1 (with the given value of y and $x := p$), obtaining

$$\sigma \geq \exp[(-1 + o(1))(\log p / \log y) \log\log p].$$

By Theorem 5.2 (Chebyshev's Theorem), we know that $k = \Theta(y/\log y)$, and so $\log k = (1 + o(1)) \log y$. Moreover, assumption (16.4) implies that the factor $(\log p)^{O(1)}$ in (16.3) is of the form $\exp[o(\log y + \log p/\log y)]$, and so we have

$$\mathsf{E}[T] \leq \exp[(1 + o(1)) \max\{(\log p/\log y)\log\log p + 2\log y,\ 3\log y\}]. \tag{16.5}$$

Let us find the value of y that minimizes the right-hand side of (16.5), ignoring the "o(1)" terms. Let $\mu := \log y$, $A := \log p \log\log p$, $S_1 := A/\mu + 2\mu$, and $S_2 := 3\mu$. We want to find μ that minimizes $\max\{S_1, S_2\}$. Using a little calculus, one sees that S_1 is minimized at $\mu = (A/2)^{1/2}$. With this choice of μ, we have $S_1 = (2\sqrt{2})A^{1/2}$ and $S_2 = (3/\sqrt{2})A^{1/2} < S_1$. Thus, choosing

$$y = \exp[(1/\sqrt{2})(\log p \log\log p)^{1/2}],$$

we obtain

$$\mathsf{E}[T] \leq \exp[(2\sqrt{2} + o(1))(\log p \log\log p)^{1/2}].$$

That takes care of the first two questions, although strictly speaking, we have only obtained an upper bound for the expected running time, and we have not shown that the choice of y is actually optimal, but we shall nevertheless content ourselves (for now) with these results. Finally, we deal with the third question, on the probability that the algorithm outputs "failure."

Lemma 16.2 *The probability that the algorithm outputs "failure" is* $1/q$.

Proof. Consider the values r_i, s_i, and β_i generated in the inner loop in stage 1. It is easy to see that, as random variables, the values s_i and β_i are independent, since conditioned on any fixed choice of s_i, the value r_i is uniformly distributed over $\{0, \ldots, q-1\}$, and hence β_i is uniformly distributed over G. Turning this around, we see that conditioned on any fixed choice of β_i, the value s_i is uniformly distributed over $\{0, \ldots, q-1\}$.

So now let us condition on any fixed choice of values β_i and δ_i, for $i = 1, \ldots, k+1$, that give rise to y-smooth integers. By the remarks in the previous paragraph, and by the independence of the δ_i's, we see that in this conditional probability distribution, the variables $\bar{s}_i$ are mutually independent and uniformly distributed over $\mathbb{Z}_q$, and moreover, the behavior of the algorithm is completely determined, and in particular, the values $\bar{c}_1, \ldots, \bar{c}_{k+1}$ are fixed. Therefore, in this conditional probability distribution, the probability that the algorithm outputs failure is just the probability that $\sum_i \bar{s}_i \bar{c}_i = 0$, which is $1/q$, since not all the $\bar{c}_i$'s are zero. Since this equality holds for every choice of β_i and δ_i, the lemma follows. □

Let us summarize the above discussion in the following theorem.

Theorem 16.3 *With the smoothness parameter y set to* $\exp[(1/\sqrt{2})(\log p \log\log p)^{1/2}]$*, the expected running time of algorithm SEDL is*

$$\exp[(2\sqrt{2} + o(1))(\log p \log\log p)^{1/2}].$$

The probability that algorithm SEDL outputs "failure" is $1/q$.

Note that in the description and analysis of algorithm SEDL, we have assumed that the primes $p_1, \ldots, p_k$ were pre-computed. Of course, we can construct this list of primes using, for example, the Sieve of Eratosthenes (see §5.4), and the running time of this pre-computation will be dominated by the running time of algorithm SEDL.

Exercise 16.4 This exercise has implications in cryptography. Let $n = pq$, where p and q are distinct, large primes. Let e and x be positive integers, both of which are strictly less than $\min\{p, q\}$. Suppose you are given n (but not its factorization!) along with the integers e and x. In addition, you are given access to two "oracles," which you may invoke as often as you like.

The first oracle is a "challenge oracle": each invocation of the oracle produces a "challenge" $a \in \{1, \ldots, x\}$ — distributed uniformly and independently of all other challenges.

The second oracle is a "solution oracle": you invoke this oracle with the index of a previous challenge oracle; if the corresponding challenge was a, the solution oracle returns the eth root of a modulo n, i.e., $b \in \{1, \ldots, n-1\}$ such that $b^e \equiv a \pmod{n}$ — note that b always exists and is uniquely determined.

Let us say that you "win" if you are able to compute the eth root modulo n of any challenge, but *without* invoking the solution oracle with the corresponding index of the challenge (otherwise, winning would be trivial, of course).

(a) Design a probabilistic algorithm that wins the above game, using an expected number of $\exp[(c + o(1))(\log x \log\log x)^{1/2}]$ steps, for some constant c, where a "step" is either a computation step or an oracle invocation (either challenge or solution). To simplify matters, you may count as one computation step any arithmetic operation on integers of length $O(\log n)$.

(b) Suppose invocations of the challenge oracle are "cheap," while invocations of the solution oracle are relatively "expensive." How would you modify your strategy in part (a)?

□

16.3 An Algorithm for Factoring Integers

We now present a probabilistic, subexponential-time algorithm for factoring integers. The algorithm uses techniques very similar to those used in algorithm SEDL in §16.2.

Let $n > 1$ be the integer we want to factor. We make a few simplifying assumptions. First, we assume that n is odd — this is not a real restriction, since we can always pull out any factors of 2 in a pre-processing step. Second, we assume that n is not a perfect power, i.e., not of the form a^b for integers $a > 1$ and $b > 1$ — this is also not a real restriction, since we can always partially factor n using the algorithm in §10.5 if n is a perfect power. Third, we assume that n is not prime — this may be efficiently checked using, say, the Miller-Rabin test (see §10.3). Fourth, we assume that n is not divisible by any primes up to a "smoothness parameter" y — we can ensure this using trial division, and it will be clear that the running time of this pre-computation is dominated by that of the algorithm itself.

With these assumptions, the prime factorization of n is of the form

$$n = q_1^{f_1} \cdots q_w^{f_w},$$

where the q_i's are distinct, odd primes, all greater than y, the f_i's are positive integers, and $w > 1$.

The main goal of our factoring algorithm is to find a random square root of 1 in $\mathbb{Z}_n$. Let

$$\rho : \mathbb{Z}_{q_1^{f_1}} \times \cdots \times \mathbb{Z}_{q_w^{f_w}} \to \mathbb{Z}_n$$

be the ring isomorphism of the Chinese Remainder Theorem. The square roots of 1 in $\mathbb{Z}_n$ are precisely those elements of the form $\rho(\pm 1, \ldots, \pm 1)$, and if β is a random square root of 1, then with probability $1 - 2^{-w+1} \geq 1/2$, it will be of the form $\beta = \rho(\beta_1, \ldots, \beta_w)$, where the β_i's are neither all 1 nor all -1 (i.e., $\beta \neq \pm 1$). If this happens, then $\beta - 1 = \rho(\beta_1 - 1, \ldots, \beta_w - 1)$, and so we see that some, but not all, of the values components $\beta_i - 1$ will be zero. The value of $\gcd(\mathrm{rep}(\beta - 1), n)$ is precisely the product of the prime powers $q_i^{f_i}$ such that $\beta_i - 1 = 0$, and hence this gcd will yield a non-trivial factorization of n, unless $\beta = \pm 1$.

Let $p_1, \ldots, p_k$ be the primes up to the smoothness parameter y mentioned above. Let $\pi_i := [p_i \pmod n] \in \mathbb{Z}_n^*$ for $i = 1, \ldots, k$. Let us write $\bar{a}$ to denote the image of an integer a in $\mathbb{Z}_2$, and likewise, for a vector v with integer entries, $\bar{v}$ denotes its image as a vector with entries in $\mathbb{Z}_2$.

We first describe a simplified version of the algorithm, after which we modify the algorithm slightly to deal with a technical problem. Like algorithm SEDL, this algorithm proceeds in two stages. In the first stage, we find relations of the form

$$\alpha_i^2 = \pi_1^{e_{i1}} \cdots \pi_k^{e_{ik}}, \tag{16.6}$$

for $\alpha_i \in \mathbb{Z}_n^*$, and $i = 1, \ldots, k+1$.

We can obtain such a relation by randomized search, as follows: we select $\alpha_i \in \mathbb{Z}_n^*$ at random, square it, and try to factor $m_i := \mathrm{rep}(\alpha_i^2)$ by trial division, trying all the primes $p_1, \ldots, p_k$ up to y. If we are lucky, we obtain a factorization

$$m_i = p_1^{e_{i1}} \cdots p_k^{e_{ik}},$$

for some exponents $e_{i1}, \ldots, e_{ik}$, yielding the relation (16.6).

For $i = 1, \ldots, k+1$, let $v_i := (e_{i1}, \ldots, e_{ik}) \in \mathbb{Z}^{\times k}$. The vectors $\bar{v}_1, \ldots, \bar{v}_{k+1} \in \mathbb{Z}_2^{\times k}$ must be linearly independent, and the second stage uses Gaussian elimination to find integers $c_1, \ldots, c_{k+1} \in \{0, 1\}$, not all zero, such that $\bar{c}_1 \bar{v}_1 + \cdots + \bar{c}_{k+1} \bar{v}_{k+1} = 0$. Let

$$(e_1, \ldots, e_k) := c_1 v_1 + \cdots c_{k+1} v_{k+1} \in \mathbb{Z}^{\times k}.$$

Raising each equation (16.6) to the power c_i, and multiplying them all together, we obtain

$$\alpha^2 = \pi_1^{e_1} \cdots \pi_k^{e_k},$$

where

$$\alpha := \prod_{i=1}^{k+1} \alpha_i^{c_i}.$$

Since each e_i is even, we can compute

$$\beta := \pi_1^{e_1/2} \cdots \pi_k^{e_k/2} \alpha^{-1},$$

and we see that β is a square root of 1 in $\mathbb{Z}_n$. A more careful analysis (see below) shows that in fact, β is uniformly distributed over all square roots of 1, and hence, with probability at least $1/2$, if we compute $\gcd(\mathrm{rep}(\beta - 1), n)$, we get a non-trivial factor of n.

That is the basic idea of the algorithm. There is, however, a technical problem. Namely, in the method outlined above for generating a relation, we attempt to factor $m_i := \mathrm{rep}(\alpha_i^2)$. Thus, the running time of the algorithm will depend in a crucial way on the probability that a random square modulo n is y-smooth. Unfortunately for us, our Theorem 16.1 does not say anything about this situation — it only applies to the situation where a number is chosen at random from an interval $[1, x]$. There are (at least) three different ways to address this problem:

1. Ignore it, and just assume that the bounds in Theorem 16.1 apply to random squares modulo n (taking $x := n$ in the theorem).

2. Prove a version of Theorem 16.1 that applies to random squares modulo n.

3. Modify the factoring algorithm, so that Theorem 16.1 applies.

The first choice, while not completely unreasonable, is not very mathematically satisfying. It turns out that the second choice is a indeed a viable option (i.e., the theorem is true and is not so difficult to prove), but we opt for the third choice, as it is somewhat easier to carry out, and illustrates a probabilistic technique that is more generally useful.

So here is how we modify the basic algorithm. Instead of generating relations of the form (16.6), we generate relations of the form

$$\alpha_i^2 \delta = \pi_1^{e_{i1}} \cdots \pi_k^{e_{ik}}, \tag{16.7}$$

for $\delta \in \mathbb{Z}_n^*$, $\alpha_i \in \mathbb{Z}_n^*$, and $i = 1, \ldots, k+2$. Note that the value δ is the same in all relations.

We generate these relations as follows. For the very first relation (i.e., $i = 1$), we repeatedly choose α_1 and δ in $\mathbb{Z}_n^*$ at random, until $\mathrm{rep}(\alpha_1^2\delta)$ is y-smooth. Then, after having found the first relation, we find subsequent relations (i.e., for $i > 1$) by repeatedly choosing α_i in $\mathbb{Z}_n^*$ at random until $\mathrm{rep}(\alpha_i^2\delta)$ is random, where δ is the same value that was used in the first relation. Now, Theorem 16.1 will apply directly to determine the success probability of each attempt to generate the first relation. Having found this relation, the value $\alpha_1^2\delta$ will be uniformly distributed over all y-smooth elements of $\mathbb{Z}_n^*$ (i.e., elements whose integer representations are y-smooth). Consider the various cosets of $(\mathbb{Z}_n^*)^2$ in $\mathbb{Z}_n^*$. Intuitively, it is much more likely that a random y-smooth element of $\mathbb{Z}_n^*$ lies in a coset that contains many y-smooth elements, rather than a coset with very few, and indeed, it is reasonably likely that the fraction of y-smooth elements in the coset containing δ is not much less than the overall fraction of y-smooth elements in $\mathbb{Z}_n^*$. Therefore, for $i > 1$, each attempt to find a relation should succeed with reasonably high probability. This intuitive argument will be made rigorous in the analysis to follow.

The second stage is then modified as follows. For $i = 1, \ldots, k+2$, let $v_i := (e_{i1}, \ldots, e_{ik}, 1) \in \mathbb{Z}^{\times(k+1)}$. The vectors $\bar{v}_1, \ldots, \bar{v}_{k+2} \in \mathbb{Z}_2^{\times(k+1)}$ must be linearly independent, and we use Gaussian elimination to find integers $c_1, \ldots, c_{k+2} \in \{0, 1\}$, not all zero, such that $\bar{c}_1\bar{v}_1 + \cdots + \bar{c}_{k+2}\bar{v}_{k+2} = 0$. Let

$$(e_1, \ldots, e_{k+1}) := c_1 v_1 + \cdots + c_{k+2} v_{k+2} \in \mathbb{Z}^{\times(k+1)}.$$

Raising each equation (16.7) to the power c_i, and multiplying them all together, we obtain

$$\alpha^2 \delta^{e_{k+1}} = \pi_1^{e_1} \cdots \pi_k^{e_k},$$

where

$$\alpha := \prod_{i=1}^{k+2} \alpha_i^{c_i}.$$

Since each e_i is even, we can compute

$$\beta := \pi_1^{e_1/2} \cdots \pi_k^{e_k/2} \delta^{-e_{k+1}/2} \alpha^{-1},$$

which is a square root of 1 in $\mathbb{Z}_n$.

Here is the entire algorithm:

Algorithm SEF

$i \leftarrow 0$
repeat
    $i \leftarrow i + 1$
    repeat
        choose $\alpha_i \in \mathbb{Z}_n^*$ at random
        if $i = 1$ then choose $\delta \in \mathbb{Z}_n^*$ at random
        $m_i \leftarrow \mathrm{rep}(\alpha_i^2 \delta)$
        test if m_i is y-smooth (trial division)
    until $m_i = p_1^{e_{i1}} \cdots p_k^{e_{ik}}$ for some integers $e_{i1}, \ldots, e_{ik}$
until $i = k + 2$

set $v_i \leftarrow (e_{i1}, \ldots, e_{ik}, 1) \in \mathbb{Z}^{\times(k+1)}$ for $i = 1, \ldots, k + 2$

apply Gaussian elimination to find integers $c_1, \ldots, c_{k+2} \in \{0, 1\}$, not all zero,
    such that $\bar{c}_1 \bar{v}_1 + \cdots + \bar{c}_{k+2} \bar{v}_{k+2} = 0$.
set $(e_1, \ldots, e_{k+1}) \leftarrow c_1 v_1 + \cdots + c_{k+2} v_{k+2}$

$\alpha \leftarrow \prod_{i=1}^{k+2} \alpha_i^{c_i}, \quad \beta \leftarrow \pi_1^{e_1/2} \cdots \pi_k^{e_k/2} \delta^{-e_{k+1}/2} \alpha^{-1}$

if $\beta = \pm 1$ then
    output "failure"
else
    output $\gcd(\mathrm{rep}(\beta - 1), n)$

Now the analysis. From the discussion above, it is clear that algorithm SEF either outputs "failure," or outputs a non-trivial factor of n. So we have the same three questions to answer as we did in the analysis of algorithm SEDL:

1. What is the expected running time of algorithm SEF?
2. How should the smoothness parameter y be chosen so as to minimize the expected running time?
3. What is the probability that algorithm SEF outputs "failure"?

To answer the first question, let σ denote the probability that a random element of $\mathbb{Z}_n^*$ is y-smooth. For $i = 1, \ldots, k+2$, let X_i denote the number iterations of the inner loop of stage 1 in the ith iteration of the main loop, i.e., X_i is the number of attempts made in finding the ith relation.

Lemma 16.5 *We have*

$$\mathsf{E}[X_i] = \sigma^{-1}$$

for $i = 1, \ldots, k+2$.

Proof. We first compute $\mathsf{E}[X_1]$. As δ is chosen uniformly from $\mathbb{Z}_n^*$ and independent of α_1, at each attempt to find a relation, $\alpha_1^2\delta$ is uniformly distributed over $\mathbb{Z}_n^*$, and hence the probability that the attempt succeeds is precisely σ. This means $\mathsf{E}[X_1] = \sigma^{-1}$.

We next compute $\mathsf{E}[X_i]$ for $i > 1$. To this end, let us denote the cosets of $(\mathbb{Z}_n^*)^2$ in $\mathbb{Z}_n^*$ as $C_1, \ldots, C_t$. As it happens, $t = 2^w$, but this fact plays no role in the analysis. For $j = 1, \ldots, t$, let σ_j denote the probability that a random element of C_j is y-smooth, and let τ_j denote the probability that the value of δ determined in finding the first relation belongs to C_j.

We claim that for $j = 1, \ldots, t$, we have $\tau_j = \sigma_j \sigma^{-1} t^{-1}$. To see this, note that each coset C_j has the same number of elements, namely, $|\mathbb{Z}_n^*|t^{-1}$, and so the number of y-smooth elements in C_j is equal to $\sigma_j |\mathbb{Z}_n^*| t^{-1}$. Moreover, the value $\alpha_1^2\delta$ is uniformly distributed over all $\sigma|\mathbb{Z}_n^*|$ of the y-smooth numbers in $\mathbb{Z}_n^*$, and hence

$$\tau_j = \frac{\sigma_j |\mathbb{Z}_n^*| t^{-1}}{\sigma |\mathbb{Z}_n^*|} = \sigma_j \sigma^{-1} t^{-1},$$

which proves the claim.

Now, for a fixed value of δ and a random choice of $\alpha_i \in \mathbb{Z}_n^*$, one sees that $\alpha_i^2\delta$ is uniformly distributed over the coset containing δ. Therefore, for $j = 1, \ldots, t$, we have

$$\mathsf{E}[X_i \mid \delta \in C_j] = \sigma_j^{-1}.$$

It follows that

$$\begin{aligned} \mathsf{E}[X_i] &= \sum_{j=1}^{t} \mathsf{E}[X_i \mid \delta \in C_j] \cdot \mathsf{P}[\delta \in C_j] \\ &= \sum_{j=1}^{t} \sigma_j^{-1} \cdot \tau_j \end{aligned}$$

$$\begin{aligned} &= \sum_{j=1}^{t} \sigma_j^{-1} \cdot \sigma_j \sigma^{-1} t^{-1} \\ &= \sigma^{-1}, \end{aligned}$$

which proves the lemma. □

So in stage 1, the expected number of attempts made in generating a single relation is σ^{-1}, each such attempt takes time $k \cdot (\log n)^{O(1)}$, and we have to generate $k+2$ relations, leading to a total expected running time in stage 1 of $\sigma^{-1}k^2 \cdot (\log n)^{O(1)}$. Stage 2 is dominated by the cost of Gaussian elimination, which takes time $k^3 \cdot (\log n)^{O(1)}$. Thus, if T is the total running time of the algorithm, we have

$$\mathsf{E}[T] \leq (\sigma^{-1}k^2 + k^3) \cdot (\log n)^{O(1)}.$$

By our assumption that n is not divisible by any primes up to y, all y-smooth integers up to $n-1$ are in fact relatively prime to n. Therefore, the number of y-smooth elements of $\mathbb{Z}_n^*$ is equal to $\Psi(y, n-1)$, and since n itself is not y-smooth, this is equal to $\Psi(y,n)$. From this, it follows that

$$\sigma = \Psi(y,n)/|\mathbb{Z}_n^*| \geq \Psi(y,n)/n.$$

The rest of the running time analysis is essentially the same as in the analysis of algorithm SEDL; that is, assuming $y = \exp[(\log n)^{\lambda + o(1)}]$ for some constant $0 < \lambda < 1$, we obtain

$$\mathsf{E}[T] \leq \exp[(1+o(1)) \max\{(\log n/\log y)\log\log n + 2\log y,\ 3\log y\}]. \tag{16.8}$$

Setting $y = \exp[(1/\sqrt{2})(\log n \log\log n)^{1/2}]$, we obtain

$$\mathsf{E}[T] \leq \exp[(2\sqrt{2} + o(1))(\log n \log\log n)^{1/2}].$$

That basically takes care of the first two questions. As for the third, we have:

Lemma 16.6 *The probability that the algorithm outputs "failure" is* $2^{-w+1} \leq 1/2$.

Proof. Let θ be the squaring map on $\mathbb{Z}_n^*$. By part (b) of Exercise 8.65, if we condition on any fixed values of $\delta, \alpha_1^2, \ldots, \alpha_{k+2}^2$ that give rise to y-smooth integers, then in the resulting conditional probability distribution, the values $\alpha_1, \ldots, \alpha_{k+2}$ are mutually independent, with each α_i uniformly distributed over $\theta^{-1}(\alpha_i^2)$. Moreover, these fixed values of $\delta, \alpha_1^2, \ldots, \alpha_{k+2}^2$ completely determine the behavior of the algorithm, and in particular, the values of $c_1, \ldots, c_{k+2}$, α^2, and $e_1, \ldots, e_{k+1}$. By part (d) of Exercise 8.65, it follows that α is uniformly distributed over $\theta^{-1}(\alpha^2)$, and also that β is uniformly distributed over $\theta^{-1}(1)$. Thus, in this conditional probability distribution, β is a random square root of 1, and so $\beta = \pm 1$ with probability 2^{-w+1}. Since this holds for all relevant choices of $\delta, \alpha_1^2, \ldots, \alpha_{k+2}^2$, it also holds unconditionally. Finally, since we are assuming that $w > 1$, we have $2^{-w+1} \leq 1/2$. □

Let us summarize the above discussion in the following theorem.

Theorem 16.7 *With the smoothness parameter y set to $\exp[(1/\sqrt{2})(\log n \log\log n)^{1/2}]$, the expected running time of algorithm SEF is*

$$\exp[(2\sqrt{2} + o(1))(\log n \log\log n)^{1/2}].$$

The probability that algorithm SEF outputs "failure" is at most $1/2$.

Exercise 16.8 It is perhaps a bit depressing that after all that work, algorithm SEF only succeeds (in the worst case) with probability $1/2$. Of course, to reduce the failure probability, we can simply repeat the entire computation — with ℓ repetitions, the failure probability drops to $2^{-\ell}$. However, there is a better way to reduce the failure probability. Suppose that in stage 1, instead of collecting $k+2$ relations, we collect $k+1+\ell$ relations, where $\ell \geq 1$ is an integer parameter.

(a) Show that in stage 2, we can use Gaussian elimination to find integer vectors

$$c^{(j)} := (c_1^{(j)}, \ldots, c_{k+1+\ell}^{(j)}) \in \{0,1\}^{\times(k+1+\ell)} \quad (j = 1, \ldots, \ell)$$

such that the vectors $\bar{c}^{(1)}, \ldots, \bar{c}^{(\ell)} \in \mathbb{Z}_2^{\times(k+1+\ell)}$ are linearly independent and satisfy

$$\bar{c}_1^{(j)}\bar{v}_1 + \ldots + \bar{c}_{k+1+\ell}^{(j)}\bar{v}_{k+1+\ell} = 0 \quad (j = 1, \ldots, \ell).$$

(b) Show that given vectors $c^{(1)}, \ldots, c^{(\ell)}$ as in part (a), if for $j = 1, \ldots, \ell$, we set

$$(e_1^{(j)}, \ldots, e_{k+1}^{(j)}) \leftarrow c_1^{(j)}v_1 + \ldots + c_{k+1+\ell}^{(j)}v_{k+1+\ell},$$

$$\alpha^{(j)} \leftarrow \prod_{i=1}^{k+1+\ell} \alpha_i^{c_i^{(j)}},$$

and

$$\beta^{(j)} \leftarrow \pi_1^{e_1^{(j)}/2} \cdots \pi_k^{e_k^{(j)}/2} \delta^{-e_{k+1}^{(j)}/2} (\alpha^{(j)})^{-1},$$

then the values $\beta^{(1)}, \ldots, \beta^{(\ell)}$ are independent and uniformly distributed over the set of all square roots of 1 in $\mathbb{Z}_n$, and hence at least one of $\gcd(\mathrm{rep}(\beta^{(j)} - 1), n)$ splits n with probability at least $1 - 2^{-\ell}$.

So, for example, if we set $\ell = 20$, then the failure probability is reduced to less than one in a million, while the increase in running time over algorithm SEF will hardly be noticeable. □

16.4 Practical Improvements

Our presentation and analysis of algorithms for discrete logarithms and factoring was geared towards simplicity and mathematical rigor. However, if one really wants to compute discrete logarithms or factor numbers, then a number of important practical improvements should be considered. In this section, we sketch some of these improvements, focusing our attention on algorithms for factoring numbers (although some of the techniques apply to discrete logarithms as well). Unlike the other sections in this chapter, this chapter is more of a survey of results and techniques.

16.4.1 Better smoothness density estimates

From an algorithmic point of view, the simplest way to improve the running times of both algorithms SEDL and SEF is to use a more accurate smoothness density estimate, which dictates a different choice of the smoothness bound y in those algorithms, speeding them up significantly. While our Theorem 16.1 is a valid *lower bound* on the density of smooth numbers, it is not "tight," in the sense that the actual density of smooth numbers is somewhat higher. We quote from the literature the following result:

Theorem 16.9 *Let y be a function of x such that for some $\epsilon > 0$, we have*

$$y = \Omega((\log x)^{1+\epsilon}) \quad \textit{and} \quad u := \frac{\log x}{\log y} \to \infty$$

as $x \to \infty$. Then

$$\Psi(y, x) = x \cdot \exp[(-1 + o(1))u \log u].$$

Proof. See §16.5. □

Let us apply this result to the analysis of algorithm SEF. Assume that $y = \exp[(\log n)^{1/2+o(1)}]$ — our choice of y will in fact be of this form. With this assumption, we have $\log\log y = (1/2 + o(1))\log\log n$, and using Theorem 16.9, we can improve the inequality (16.8), obtaining instead (verify)

$$\mathsf{E}[T] \leq \exp[(1 + o(1)) \max\{(1/2)(\log n/\log y)\log\log n + 2\log y, 3\log y\}].$$

From this, if we set

$$y := \exp[(1/2)(\log n \log\log n)^{1/2})],$$

we obtain

$$\mathsf{E}[T] \leq \exp[(2 + o(1))(\log n \log\log n)^{1/2}].$$

An analogous improvement can be obtained for algorithm SEDL.

Although this improvement reduces the constant $2\sqrt{2} \approx 2.828$ to 2, the constant is in the exponent, and so this improvement is not to be scoffed at!

16.4.2 The Quadratic Sieve Algorithm

We now describe a practical improvement to algorithm SEF. This algorithm, known as the Quadratic Sieve, is faster in practice than algorithm SEF; however, the analysis of its running time is somewhat heuristic.

First, let us return to the simplified version of algorithm SEF, where we collect relations of the form (16.6). Furthermore, instead of choosing the values α_i at random, we will choose them in a special way, as we now describe. Let

$$\tilde{n} := \lfloor \sqrt{n} \rfloor,$$

and define the polynomial

$$F := (\mathtt{X} + \tilde{n})^2 - n \in \mathbb{Z}[\mathtt{X}].$$

In addition to the usual "smoothness parameter" y, we need a "sieving parameter" z, whose choice will be discussed below. We shall assume that both y and z are of the form $\exp[(\log n)^{1/2+o(1)}]$, and our ultimate choices of y and z will indeed satisfy this assumption.

For all integers $s = 1, 2, \ldots, \lfloor z \rfloor$, we shall determine for which values of s the corresponding value $F(s)$ is y-smooth — note that for $s > 0$, we have $F(s) > 0$. For each such s, since we have $F(s) \equiv (s + \tilde{n})^2 \pmod{n}$, this gives us one relation of the form (16.6), with $\alpha_i := [(s + \tilde{n}) \bmod n]$. If this procedure yields at least $k + 1$ values of s such that $F(s)$ is smooth, then we can apply Gaussian elimination as usual to find a square root β of 1 in $\mathbb{Z}_n$. Hopefully, we will have $\beta \neq \pm 1$, allowing us to split n.

Observe that for $1 \leq s \leq z$, we have

$$F(s) = (s + \tilde{n})^2 - n = s^2 + 2s\tilde{n} + \tilde{n}^2 - n \leq z^2 + 2zn^{1/2},$$

and so we have

$$F(s) \leq n^{1/2+o(1)}.$$

Now, although the values $F(s)$ are not at all random, we might expect heuristically that the number of integers s up to z such that $F(s)$ is y-smooth is roughly equal to $\hat{\sigma}z$, where $\hat{\sigma}$ is the probability that a random integer in the interval $[1, n^{1/2}]$ is y-smooth, i.e.,

$$\hat{\sigma} = \exp[(-1/4 + o(1))(\log n / \log y) \log \log n].$$

This already gives us an improvement over algorithm SEF, since now we are looking for y-smooth numbers of size around $n^{1/2}$, rather than of size around n. But there is another improvement possible; namely, instead of testing each individual number $F(s)$ for smoothness using trial division, we can test them all at once using the following "sieving procedure":

> Create a vector $v[1 \ldots \lfloor z \rfloor]$, and initialize $v[s]$ to $F(s)$, for $1 \leq s \leq z$. For each prime p up to y, do the following:
>
> 1. Compute the roots of the polynomial F modulo p.
>
>   *This can be done quite efficiently, as follows. For $p = 2$, F has exactly one root mod p, which is determined by the parity of $\tilde{n}$. For $p > 2$, we may use the familiar quadratic formula together with an algorithm for computing square roots modulo p, as discussed in Example 13.2. A quick calculation shows that the discriminant of F is n, and thus, F has a root modulo p if and only if n is a quadratic residue modulo p, in which case it will have two roots (under our usual assumptions, we cannot have $p \mid n$).*
>
> 2. Assume that the distinct roots of F modulo p lying in the interval $[1, p]$ are r_i, for $i = 1, \ldots, v_p$.
>
>   *Note that $v_p = 1$ for $p = 2$ and $v_p \in \{0, 2\}$ for $p > 2$. Also note that $F(s) \equiv 0 \pmod{p}$ if and only if $s \equiv r_i \pmod{p}$ for some $i = 1, \ldots, v_p$.*

For $i = 1, \ldots, v_p$, do the following:

$s \leftarrow r_i$
while $s \leq z$ do
    repeat $v[s] \leftarrow v[s]/p$ until $p \nmid v[s]$
    $s \leftarrow s + p$

At the end of this sieving procedure, the values of s that are y-smooth may be identified as precisely those such that $v[s] = 1$. The running time of this sieving procedure is at most $(\log n)^{O(1)}$ times

$$\sum_{p \leq y} \frac{z}{p} = z \sum_{p \leq y} \frac{1}{p} = O(z \log\log y) = z^{1+o(1)}$$

Here, we have made use of Theorem 5.14, although this is not really necessary — for our purposes, the bound $\sum_{p \leq y}(1/p) = O(\log y)$ would suffice. Note that this sieving procedure is a factor of $k^{1+o(1)}$ faster than the method for finding smooth numbers based on trial division. With just a little extra book-keeping, we can not only identify the values s such that $F(s)$ is smooth, but we can also compute the factorization of $F(s)$ into primes.

Now, let us put together all the pieces. We have to choose z just large enough so as to find at least $k+1$ values of s up to z such that $F(s)$ is y-smooth. So we should choose z so that $z \approx k/\hat{\sigma}$ — in practice, we could choose an initial estimate for z, and if this choice of z does not yield enough relations, we could keep doubling z until we do get enough relations. Assuming that $z \approx k/\hat{\sigma}$, the cost of sieving is $(k/\hat{\sigma})^{1+o(1)}$, or

$$\exp[(1 + o(1))(1/4)(\log n/\log y)\log\log n + \log y].$$

The cost of Gaussian elimination is still $O(k^3)$, or

$$\exp[(3 + o(1))\log y].$$

Thus, if T is the running time of the entire algorithm, we have

$$T \leq \exp[(1 + o(1))\max\{(1/4)(\log n/\log y)\log\log n + \log y,\ 3\log y\}].$$

Let $\mu := \log y$, $A := (1/4)\log n \log\log n$, $S_1 := A/\mu + \mu$ and $S_2 := 3\mu$, and let us find the value of μ that minimizes $\max\{S_1, S_2\}$. Using a little calculus, one finds that S_1 is minimized at $\mu = A^{1/2}$. For this value of μ, we have $S_1 = 2A^{1/2}$ and $S_2 = 3A^{1/2} > S_1$, and so this choice of μ is a bit larger than optimal. For $\mu < A^{1/2}$, S_1 is decreasing (as a function of μ), while S_2 is always increasing. It follows that the optimal value of μ is obtained by setting

$$A/\mu + \mu = 3\mu$$

and solving for μ. This yields $\mu = (A/2)^{1/2}$. So setting

$$y = \exp[(1/(2\sqrt{2}))(\log n \log\log n)^{1/2}],$$

we have

$$T \leq \exp[(3/(2\sqrt{2}) + o(1))(\log n \log\log n)^{1/2}].$$

Thus, we have reduced the constant in the exponent from 2, for algorithm SEF (using the more accurate smoothness density estimates), to $3/(2\sqrt{2}) \approx 1.061$.

We mention one final improvement. The matrix to which we apply Gaussian elimination in stage 2 is "sparse"; indeed, since any integer less than n has $O(\log n)$ prime factors, the total number of non-zero entries in the matrix is $k^{1+o(1)}$. In this case, there are special algorithms (which we do not discuss in this text, but see §16.5) for working with such sparse matrices, which allow us to perform stage 2 of the factoring algorithm in time $k^{2+o(1)}$, or

$$\exp[(2 + o(1)) \log y].$$

This gives us

$$T \leq \exp[(1 + o(1)) \max\{(1/4)(\log n/\log y)\log\log n + \log y,\ 2\log y\}],$$

and setting

$$y = \exp[(1/2)(\log n \log\log n)^{1/2}]$$

yields

$$T \leq \exp[(1 + o(1))(\log n \log\log n)^{1/2}].$$

Thus, this improvement reduces the constant in the exponent from $3/(2\sqrt{2}) \approx 1.061$ to 1. Moreover, the special algorithms designed to work with sparse matrices typically use much less space than ordinary Gaussian elimination — even if the input to Gaussian elimination is sparse, the intermediate matrices will not be.

The Quadratic Sieve may fail to factor n, for one of two reasons: first, it mail fail to find $k+1$ relations; second, it may find these relations, but in stage 2, it only finds a trivial square root of 1. There is no rigorous theory to say why the algorithm should not fail for one of these two reasons, but experience shows that the algorithm does indeed work as expected.

16.5 Notes

Many of the algorithmic ideas in this chapter were first developed for the problem of factoring integers, and then later adapted to the discrete logarithm problem. The first (heuristic) subexponential-time algorithm for factoring integers, called the *Continued Fraction Method* (not discussed here), was introduced by Lehmer and Powers [42], and later refined and implemented by Morrison and Brillhart [50]. The first rigorously analyzed subexponential-time algorithm for factoring integers was introduced by Dixon [26]. Our algorithm SEF is a variation of Dixon's algorithm, which works the same way as algorithm SEF, except that it generates relations of the form (16.6) directly (and indeed, it is possible to prove a variant

of Theorem 16.1, and for that matter, Theorem 16.9, for random squares modulo n). Our algorithm SEF uses an idea suggested by Rackoff (personal communication).

Theorem 16.9 was proved by Canfield, Erdős, and Pomerance [17]. The Quadratic Sieve was introduced by Pomerance [57]. Recall that the Quadratic Sieve has a heuristic running time of

$$\exp[(1 + o(1))(\log n \log\log n)^{1/2}].$$

This running time bound can also be achieved *rigorously* by a probabilistic algorithm [44], and to date, this is the fastest rigorously analyzed factoring algorithm. We should stress, however, that most practitioners in this field are not so much interested in rigorous running time analyses as they are in actually factoring integers, and for such purposes, heuristic running time estimates are quite acceptable. Indeed, the Quadratic Sieve is much more practical than the algorithm in [44], which is mainly of theoretical interest.

There are two other factoring algorithms not discussed here, but that should anyway at least be mentioned. The first is the *Elliptic Curve Method*, introduced by Lenstra [43]. Unlike all of the the other known subexponential-time algorithms, the running time of this algorithm is sensitive to the sizes of the factors of n; in particular, if p is the smallest prime dividing n, the algorithm will find p (heuristically) in expected time

$$\exp[(\sqrt{2} + o(1))(\log p \log\log p)^{1/2}].$$

This algorithm is quite practical, and is the method of choice when it is known (or suspected) that n has some small factors. It also has the advantage that it uses only polynomial space (unlike all of the other known subexponential-time factoring algorithms).

The second is the *Number Field Sieve*, the basic idea of which was introduced by Pollard [56], and later generalized and refined by Buhler, Lenstra, and Pomerance [16], as well as by others. The Number Field Sieve will split n (heuristically) in expected time

$$\exp[(c + o(1))(\log n)^{1/3}(\log\log n)^{2/3}],$$

where c is a constant (currently, the smallest value of c is 1.902 [21]). The Number Field Sieve is currently the asymptotically fastest known factoring algorithm (at least, heuristically), and it is also practical, having been used to set the latest factoring record — the factorization of a 512-bit integer that is the product of two primes of about the same size (see Cavallar, *et al.* [19]).

As for subexponential-time algorithms for discrete logarithms, Adleman [1] adapted the ideas used for factoring to the discrete logarithm problem, although it seems that some of the basic ideas were known much earlier. Our algorithm SEDL is a variation on this algorithm, and the basic technique is usually referred to as the *Index Calculus Method*. Note that our restriction to subgroups of prime order q such that $q^2 \nmid (p-1)$ greatly simplifies the linear algebra; otherwise, things can get a bit tricky. The basic idea of the number field sieve was adapted to the discrete logarithm problem by Gordon [29]; see also Adleman [2] and Schirokauer, Weber, and Denny [62].

For many more details and references for subexponential-time algorithms for factoring and discrete logarithms, see Chapter 6 of Crandall and Pomerance [23].

Last, but not least, we should mention the fact that there are in fact *polynomial time* algorithms for factoring and discrete logarithms; however, these algorithms require special hardware, namely, a *quantum computer*. Shor [66, 67] showed that these problems could be solved in polynomial time on such a device; however, at the present time, it is unclear when and if such machines will ever be built. Much, indeed most, of modern-day cryptography will crumble if this happens, or if efficient "classical" algorithms for these problems are discovered (which is still a real possibility).

Chapter 17

More Rings

This chapter develops a number of other concepts concerning rings.

17.1 The Field of Fractions of an Integral Domain

Let D be any integral domain. Just as we can form the field of rational numbers by forming fractions involving integers, we can construct a field consisting of fractions whose numerators and denominators are elements of D. This construction is quite straightforward, but to do it carefully is a bit tedious.

First, we define an auxiliary ring R as follows. R consists of all pairs $(a, b) \in D^{\times 2}$, with $b \neq 0_D$. Addition and multiplication in R are defined as follows:

$$(a, b) + (c, d) := (ad + bc, bd), \qquad (a, b) \cdot (c, d) := (ac, bd).$$

The fact that D is an integral domain ensures that if $b \neq 0_D$ and $d \neq 0_D$, then $bd \neq 0_D$, so these rules for addition and multiplication are well-defined binary operations on R. We leave it to the reader to verify that R is a ring, and in fact, an integral domain.

Next, we define an ideal M in R as follows. M consists of all pairs of the form $(0_D, b)$, with $b \neq 0_D$. We leave it to the reader to verify that M is an ideal, and that $R^* = R \setminus M$, i.e., the set of invertible elements in R consists precisely of those elements of R that lie outside M.

Finally, we define the quotient ring $K := R/M$. This is the **field of fractions of** F.

We next state and prove some basic properties about K.

First, we claim that K is a field — this follows immediately from the observation in the above paragraph that $R^* = R \setminus M$.

Second, we claim that the map $\rho : D \to K$ that sends $a \in D$ to $[(a, 1_D) \bmod M] \in K$ is an embedding. To see this, one verifies (1) that the map σ that sends $a \in D$ to $(a, 1_D) \in R$ is a ring homomorphism, (2) that the kernel of σ is trivial, so σ is injective, and (3) that if $a \neq 0_D$, then $\sigma(a) \notin M$. Since ρ is the composition of σ with the natural map from R to R/M, the claim that ρ is an embedding follows immediately from the above three observations.

So starting from D, we can synthesize "out of thin air" its field of fractions K, which essentially contains D as a subring, via the canonical embedding $\rho : D \to K$.

Now suppose that we are given a field L that contains D as a subring. Consider the set K' consisting of all elements in L of the form ab^{-1}, where $a, b \in D$ and $b \neq 0$ — note that here, the arithmetic operations are performed using the rules for arithmetic in L. One may easily verify that K' is a subfield of L that contains D, and it is easy to see that this is the smallest subfield of L that contains D. One may also easily verify that the map $\tau : K \to K'$ that sends $[(a, b) \bmod M] \in K$ to $ab^{-1} \in K'$ is an isomorphism. A fine point: in so far as we view D to be a subring of K, via the canonical embedding ρ above, the map τ from K to K' acts as the identity on D. The subfield K' of L may be referred to as the **field of fractions of D within L**.

From now on, we shall simply write elements of the field of fractions K of D as fractions a/b, where $a, b \in D$ and $b \neq 0_D$. One can check that all of the usual rules for fractions learnt in elementary school carry over to this more general setting; in particular,

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}, \quad \frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}, \quad \text{and} \quad \frac{a}{b} = \frac{c}{d} \text{ iff } ad = bc.$$

Note that because of the fact that every integral domain can be embedded in a field, it would have been sufficient to state and prove Theorem 9.41 for fields rather than for integral domains — the statement of this theorem in terms of the more general notion of an integral domain is really not any more general than the corresponding statement for fields.

Function fields. An important special case of the above construction for the field of fractions of D is when $D = F[\mathtt{X}]$, where F is a field. In this case, the field of fractions is denoted $F(\mathtt{X})$, and is called the **field of rational functions (over F)**. This terminology is a bit unfortunate, since just as with polynomials, although the elements of $F(\mathtt{X})$ define functions, they are not (in general) in one-to-one correspondence with these functions.

Since $F[\mathtt{X}]$ is a subring of $F(\mathtt{X})$, and since F is a subring of $F[\mathtt{X}]$, we see that F is a subfield of $F(\mathtt{X})$.

More generally, we may apply the above construction to the ring $D = F[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ of multi-variate polynomials over a field F, in which case the field of fractions is denoted $F(\mathtt{X}_1, \ldots, \mathtt{X}_n)$, and is also called the field of rational functions (over F, in the variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$).

Exercise 17.1 Let F be a field of characteristic zero. Show that F contains an isomorphic copy of $\mathbb{Q}$. □

Exercise 17.2 Show that the field of fractions of $\mathbb{Z}[i]$ within $\mathbb{C}$ is $\mathbb{Q}[i]$. (See Example 9.27 and Exercise 9.32.) □

17.2 Unique Factorization of Polynomials

Throughout this section, F denotes a field.

Like the ring $\mathbb{Z}$, the ring $F[\mathtt{X}]$ of polynomials is an integral domain, and as we shall see, because of the division with remainder property for polynomials, $F[\mathtt{X}]$ has many other properties in common with $\mathbb{Z}$. Indeed, essentially all the ideas and results from Chapters 1 and 2 can be carried over almost immediately from $\mathbb{Z}$ to $F[\mathtt{X}]$, and in this section and the next, we shall do just that.

Recall that for $a, b \in F[\mathtt{X}]$, we write $b \mid a$ if $a = bc$ for some $c \in F[\mathtt{X}]$; note that $\deg(a) = \deg(b) + \deg(c)$. Also, recall that because of the cancellation law for an integral domain, if $b \mid a$ and $b \neq 0$, then the choice of c above is unique, and may be denoted a/b.

The units of $F[\mathtt{X}]$ are precisely the units F^* of F; i.e., the non-zero constants. We call two polynomials $a, b \in F[\mathtt{X}]$ **associates** if $a = bu$ for $u \in F^*$. Clearly, any non-zero polynomial a is associate to a unique monic polynomial (i.e., with leading coefficient 1), called the **monic associate** of a. Note that a polynomial a is a unit if and only if it is associate to 1. Let us call a polynomial **normalized** if it is either zero or monic.

We call a polynomial p **irreducible** if it is non-constant and all divisors of p are associate to 1 or p. Conversely, we call a polynomial n **reducible** if it is non-constant and is not irreducible. Equivalently, non-constant n is reducible if and only if there exist polynomials $a, b \in F[\mathtt{X}]$ of degree strictly less that n such that $n = ab$.

Clearly, if a and b are associate polynomials, then a is irreducible if and only if b is irreducible.

The irreducible polynomials play a role similar to that of the prime numbers. Just as it is convenient to work with only positive prime numbers, it is also convenient to restrict attention to monic irreducible polynomials.

Corresponding to Theorem 1.2, every non-zero polynomial can be expressed as a unit times a product of monic irreducibles in an essentially unique way:

Theorem 17.3 *Every non-zero polynomial $n \in F[\mathtt{X}]$ can be expressed as*

$$n = u \cdot \prod_p p^{\nu_p(n)},$$

where u is a unit, and the product is over all monic irreducible polynomials, with all but a finite number of the exponents zero. Moreover, the exponents and the unit are uniquely determined by n.

To prove this theorem, we may assume that n is monic, since the non-monic case trivially reduces to the monic case.

The proof of the existence part of Theorem 17.3 is just as for Theorem 1.2. If n is 1 or a monic irreducible, we are done. Otherwise, there exist $a, b \in F[\mathtt{X}]$ of degree strictly less than n such that $n = ab$, and again, we may assume that a and b are monic. By induction on degree, both a and b can be expressed as a product of monic irreducible polynomials, and hence, so can n.

The proof of the uniqueness part of Theorem 17.3 is almost identical to that of Theorem 1.2. Analogous to Theorem 1.10, we have:

Theorem 17.4 *For any ideal $I \subset F[\mathtt{X}]$, there exists a unique normalized polynomial d such that $I = (d)$.*

Proof. We first prove the existence part of the theorem. If $I = \{0\}$, then $d = 0$ does the job, so let us assume that $I \neq \{0\}$. Let d be a monic polynomial of minimal degree in I. We want to show that $I = (d)$.

We first show that $I \subset (d)$. To this end, let c be any element in I. It suffices to show that $d \mid c$. Using the Division with Remainder Property, write $c = qd + r$, where $\deg(r) < \deg(d)$. Then by the closure properties of ideals, one sees that $r = c - qd$ is also an element of I, and by the minimality of the choice of d, we must have $r = 0$. Thus, $d \mid c$.

We next show that $(d) \subset I$. This follows immediately from the fact that $d \in I$ and the closure properties of ideals.

That proves the existence part of the theorem. As for uniqueness, note that if $(d) = (d')$, we have $d \mid d'$ and $d' \mid d$, from which it follows that $d' = ud$ for a unit u. □

For $a, b \in F[\mathtt{X}]$, we call $d \in F[\mathtt{X}]$ a **common divisor** of a and b if $d \mid a$ and $d \mid b$; moreover, we call such a d the **greatest common divisor** of a and b if d is normalized, and all other common divisors of a and b divide d. It is immediate from the definition of a greatest common divisor that it is unique if it exists at all.

Analogous to Theorem 1.11, we have:

Theorem 17.5 *For any $a, b \in F[\mathtt{X}]$, there exists a greatest common divisor d of a and b, and moreover, $(a, b) = (d)$; in particular, $as + bt = d$ for some $s, t \in F[\mathtt{X}]$.*

Proof. We apply the previous theorem to the ideal $I = (a, b)$. Let $d \in F[\mathtt{X}]$ with $I = (d)$, as in that theorem. Note that $a, b, d \in I$.

It is clear that d is a common divisor of a and b. Moreover, there exist $s, t \in F[\mathtt{X}]$ such that $as + bt = d$. If $d' \mid a$ and $d' \mid b$, then clearly $d' \mid (as + bt)$, and hence $d' \mid d$. □

For $a, b \in F[\mathtt{X}]$, we denote by $\gcd(a, b)$ the greatest common divisor of a and b.

We say that a and b are **relatively prime** if $\gcd(a, b) = 1$. Notice that a and b are relatively prime if and only if $(a, b) = F[\mathtt{X}]$, i.e., if and only if there exist $s, t \in F[\mathtt{X}]$ such that $as + bt = 1$.

Analogous to Theorem 1.12, we have:

Theorem 17.6 *For $a, b, c \in F[\mathtt{X}]$ such that $c \mid ab$ and $\gcd(a, c) = 1$, we have $c \mid b$.*

Proof. Suppose that $c \mid ab$ and $\gcd(a, c) = 1$. Then since $\gcd(a, c) = 1$, by Theorem 17.5 we have $as + ct = 1$ for some $s, t \in F[\mathtt{X}]$. Multiplying this equation by b, we obtain $abs + cbt = b$. Since d divides ab by hypothesis, it follows that $c \mid (abs + cbt)$, and hence $c \mid b$. □

Analogous to Theorem 1.13, we have:

Theorem 17.7 *Let $p \in F[\mathtt{X}]$ be irreducible, and let $a, b \in F[\mathtt{X}]$. Then $p \mid ab$ implies that $p \mid a$ or $p \mid b$.*

Proof. Assume that $p \mid ab$. The only divisors of p are associate to 1 or p. Thus, $\gcd(p, a)$ is either 1 or the monic associate of p. If $p \mid a$, we are done; otherwise, if $p \nmid a$, we must have $\gcd(p, a) = 1$, and by the previous theorem, we conclude that $p \mid b$. □

Now to prove the uniqueness part of Theorem 17.3. Clearly, the choice of the unit u is uniquely determined: $u = \mathrm{lc}(n)$. Suppose we have

$$p_1 \cdots p_r = p'_1 \cdots p'_s,$$

where the p_i and p'_i are monic irreducible polynomials (duplicates are allowed among the p_i and among the p'_i). If $r = 0$, we must have $s = 0$ and we are done. Otherwise, as p_1 divides the right-hand side, by inductively applying Theorem 17.7, one sees that p_1 is equal to some p'_i. We can cancel these terms and proceed inductively (on r).

That completes the proof of Theorem 17.3.

Because of the unique factorization property of $F[\mathtt{X}]$, any rational function $a/b \in F(\mathtt{X})$ can be expressed as a fraction a'/b' in "lowest terms," that is, $a/b = a'/b'$ and $\gcd(a', b') = 1$, and this representation is unique up to multiplication by units.

For all polynomials a and b, it is easy to see that

$$\gcd(a, b) = \prod_p p^{\min(\nu_p(a), \nu_p(b))},$$

where the function $\nu_p(\cdot)$ is as implicitly defined in Theorem 17.3.

For $a, b \in F[\mathtt{X}]$ a **common multiple** of a and b is a polynomial m such that $a \mid m$ and $b \mid m$; moreover, such an m is the **least common multiple** of a and b if m is normalized, and m divides all common multiples of a and b. In light of Theorem 17.3, it is clear that the least common multiple exists and is unique; indeed, if we denote the least common multiple of a and b as $\mathrm{lcm}(a, b)$, then for all polynomials a and b, we have

$$\mathrm{lcm}(a, b) = \prod_p p^{\max(\nu_p(a), \nu_p(b))}.$$

Moreover, for all $a, b \in F[\mathtt{X}]$, we have

$$\gcd(a, b) \cdot \mathrm{lcm}(a, b) = ab.$$

Just as in §1.3, the notions of greatest common divisor and least common multiple generalize from two to any number of polynomials.

17.3 Polynomial Congruences

Throughout this section, F denotes a field.

Recall that for polynomials $a, b, n \in F[\mathtt{X}]$, we write $a \equiv b \pmod{n}$ when $n \mid (a - b)$. For a non-zero polynomial n, and $a \in F[\mathtt{X}]$, we say that a is a **unit modulo** n if there exists

$a' \in F[\mathtt{X}]$ such that $aa' \equiv 1 \pmod{n}$, in which case we say that a' is a **multiplicative inverse of a modulo n**.

All of the results we proved in Chapter 2 for integer congruences carry over almost identically to polynomials. As such, we do not give proofs of any of the results here. The reader may simply check that the proofs of the corresponding results translate almost directly.

Theorem 17.8 *An polynomial a is a unit modulo n if and only if a and n are relatively prime.*

Theorem 17.9 *If a is relatively prime to n, then $az \equiv az' \pmod{n}$ if and only if $z \equiv z' \pmod{n}$. More generally, if $d = \gcd(a, n)$, then $az \equiv az' \pmod{n}$ if and only if $z \equiv z' \pmod{n/d}$.*

Theorem 17.10 *Let n be a non-zero polynomial and let $a, b \in F[\mathtt{X}]$. If a is relatively prime to n, then the congruence $az \equiv b \pmod{n}$ has a solution z; moreover, any polynomial z' is a solution if and only if $z \equiv z' \pmod{n}$.*

Theorem 17.11 *Let n be a non-zero polynomial and let $a, b \in F[\mathtt{X}]$. Let $d = \gcd(a, n)$. If $d \mid b$, then the congruence $az \equiv b \pmod{n}$ has a solution z, and any polynomial z' is also a solution if and only if $z \equiv z' \pmod{n/d}$. If $d \nmid b$, then the congruence $az \equiv b \pmod{n}$ has no solution z.*

Theorem 17.12 (Chinese Remainder Theorem) *Let $k > 0$, and let $a_1, \ldots, a_k \in F[\mathtt{X}]$, and let $n_1, \ldots, n_k$ be non-zero polynomials such that $\gcd(n_i, n_j) = 1$ for all $1 \leq i < j \leq k$. Then there exists a polynomial z such that*

$$z \equiv a_i \pmod{n_i} \quad (i = 1, \ldots, k).$$

Moreover, any other polynomial z' is also a solution of these congruences if and only if $z \equiv z' \pmod{n}$, where $n := \prod_{i=1}^{k} n_i$.

The Chinese Remainder Theorem also has a more algebraic interpretation. Define the F-algebras $A_i := F[\mathtt{X}]/(n_i)$ for $1 \leq i \leq k$, along with the product F-algebra $A_1 \times \cdots \times A_k$. The map ρ from $F[\mathtt{X}]$ to $A_1 \times \cdots \times A_k$ that sends $z \in F[\mathtt{X}]$ to $([z \bmod n_1], \ldots, [z \bmod n_k])$ is an F-algebra homomorphism. The Chinese Remainder Theorem says that ρ is surjective with kernel (n), giving rise to an F-algebra isomorphism between $F[\mathtt{X}]/(n)$ and $A_1 \times \cdots \times A_k$.

Let us recall the formula for the solution z (see proof of Theorem 2.7). We have

$$z := \sum_{i=1}^{k} w_i a_i,$$

where

$$w_i := n_i' m_i, \quad n_i' := n/n_i, \quad m_i n_i' \equiv 1 \pmod{n_i} \quad (i = 1, \ldots, k).$$

Now, let us consider the special case of the Chinese Remainder Theorem where $a_i \in F$ and and $n_i = (\mathtt{X} - b_i)$ with $b_i \in F$, for $1 \leq i \leq k$. The condition that $\gcd(n_i, n_j) = 1$ for all $i \neq j$ is equivalent to the condition that $b_i \neq b_j$ for all $i \neq j$. A polynomial z satisfies the system of congruences if and only if $z(b_i) = a_i$ for $1 \leq i \leq k$. Moreover, we have $n_i' = \prod_{j \neq i}(\mathtt{X} - b_j)$, and $m_i := 1/\prod_{j \neq i}(b_i - b_j)$ is a multiplicative inverse of n_i' modulo n_i. So we get

$$z = \sum_{i=1}^{k} a_i \frac{\prod_{j \neq i}(\mathtt{X} - b_j)}{\prod_{j \neq i}(b_i - b_j)}.$$

The reader will recognize this as the LaGrange Interpolation Formula. Thus, the Chinese Remainder Theorem for polynomials includes LaGrange Interpolation as a special case.

We can now bring to bear the theory of vector spaces. Consider again the F-algebra homomorphism $\rho : F[\mathtt{X}] \to A_1 \times \cdots \times A_k$ discussed above. If $n_i = (\mathtt{X} - b_i)$ for $1 \leq i \leq k$, then each A_i is just an isomorphic copy of F, and the map ρ sends $z \in F[\mathtt{X}]$ to $(z(b_1), \ldots, z(b_k))$ in $F^{\times k}$. Both $F[\mathtt{X}]$ and $F^{\times k}$ are F-vector spaces, and the map ρ is an F-linear map. Moreover, the restriction $\tilde{\rho}$ of ρ to the k-dimensional subspace $F[\mathtt{X}]_{<k}$ of $F[\mathtt{X}]$, consisting of all polynomials of degree strictly less than k, is also an F-linear map, and by the Chinese Remainder Theorem, the image of $\tilde{\rho}$ is still all of $F^{\times k}$. Thus, $\tilde{\rho}$ is an F-vector space *isomorphism* of $F[\mathtt{X}]_{<k}$ with $F^{\times k}$.

We may encode elements of $F[\mathtt{X}]_{<k}$ as row vectors in a natural way, encoding the polynomial $z = \sum_{i=0}^{k-1} z_i \mathtt{X}^i$ as the row vector $(z_0, \ldots, z_{k-1}) \in F^{1 \times k}$. With this encoding, we have

$$\tilde{\rho}(z) = (z_0, \ldots, z_{k-1})V,$$

where V is the $k \times k$ matrix

$$V := \begin{pmatrix} 1 & 1 & & 1 \\ b_1 & b_2 & & b_k \\ \vdots & \vdots & \cdots & \vdots \\ b_1^{k-1} & b_2^{k-1} & \cdots & b_k^{k-1} \end{pmatrix}.$$

The matrix V (well, actually its transpose) is known as a **Vandermonde matrix**. Because $\tilde{\rho}$ is an isomorphism, it follows that the matrix V is invertible.

More generally, for $\ell \leq k$, one might also consider linear transformations $\sigma : F[\mathtt{X}]_{<k} \to F^{\times \ell}$ that send $z \in F[\mathtt{X}]_{<k}$ to $(z(b_1), \ldots, z(b_\ell))$, for fixed $b_1, \ldots, b_\ell \in F$. If $z = \sum_{i=0}^{k-1} z_i \mathtt{X}^i$, then

$$\sigma(z) = (z_0, \ldots, z_{k-1})W,$$

where W is the $k \times \ell$ matrix

$$W := \begin{pmatrix} 1 & 1 & & 1 \\ b_1 & b_2 & & b_\ell \\ \vdots & \vdots & \cdots & \vdots \\ b_1^{k-1} & b_2^{k-1} & \cdots & b_\ell^{k-1} \end{pmatrix}.$$

Now, if $b_i = b_j$ for some $i \neq j$, then the columns of W are linearly dependent, and hence the column rank of W is less than ℓ. Since the column rank of W is equal to its row rank, the dimension of the row space of W is less than ℓ, and hence, σ is not surjective. Conversely, if the b_i's are pair-wise distinct, then since the submatrix of W consisting of its first ℓ rows is an invertible Vandermonde matrix, we see that the rank of W is equal to ℓ, and hence σ is surjective.

17.4 Polynomial Quotient Algebras

Throughout this section, F denotes a field.

Let $f \in F[\mathtt{X}]$ be a monic polynomial, and consider the quotient algebra $A := F[\mathtt{X}]/(f)$. Let $\eta := [\mathtt{X} \bmod f]$, so that $A = F[\eta]$.

If $f = 1$, then A is just the trivial algebra, and there is not much more to say, so assume $\ell := \deg(f) > 0$. As A contains an isomorphic copy of F via the canonical embedding that sends $c \in F$ to $[c \bmod F]$, we may simply view F as a subring of A. Also, A has dimension ℓ over F, with $1, \eta, \ldots, \eta^{\ell-1}$ being a basis. That is, every element of A can be expressed uniquely as $g(\eta)$ for $g \in F[\mathtt{X}]$ of degree less than ℓ.

Now, if f is irreducible, then since every polynomial $a \not\equiv 0 \pmod f$ is invertible modulo f, it follows that A is a field. Conversely, if f is not irreducible, then A cannot be a field — indeed, if g is a non-trivial factor of f, then $g(\eta)$ is a zero divisor.

If $F = \mathbb{Z}_p$ for a prime number p, and f is irreducible, then we see that E is a finite field of cardinality p^ℓ. As we shall see later, for any prime p and any positive integer ℓ, there exists an irreducible polynomial of degree ℓ over $\mathbb{Z}_p$, and so there exists a finite field of cardinality p^ℓ. In the next chapter, we shall see how one can perform arithmetic in such extension fields efficiently, and later, we shall also see how to efficiently construct irreducible polynomials of any given degree over a finite field. Although different irreducible polynomials give rise to finite fields that superficially look very different, we shall also see that all finite fields of the same cardinality are isomorphic.

Minimal polynomials. Now suppose that A is any F-algebra, and that $\alpha \in A$. Consider the polynomial evaluation map $\rho : F[\mathtt{X}] \to A$ that sends $g \in F[\mathtt{X}]$ to $g(\alpha)$. The kernel of ρ is an ideal in $F[\mathtt{X}]$, and since every ideal in $F[\mathtt{X}]$ is principal, it follows that there exists a normalized polynomial ϕ such that $\ker(\rho) = (\phi)$. The polynomial ϕ is called the **minimal polynomial of α (over F)**. If $\phi = 0$, then ρ is injective, and hence the image $A[\alpha]$ of ρ is isomorphic to $F[\mathtt{X}]$. Otherwise, $A[\alpha]$ is isomorphic to $F[\mathtt{X}]/(\phi)$; moreover, since any

polynomial that is zero at α is a polynomial multiple of ϕ, we see that ϕ is the unique monic polynomial of smallest degree that is zero at α.

If A is a finite dimensional F-algebra, with say dimension n, then any $\alpha \in A$ has a non-zero minimal polynomial. Indeed, the elements $1, \alpha, \ldots, \alpha^n$ must be linearly dependent, and hence there exist $c_0, \ldots, c_n \in F$, not all zero, such that $c_0 + c_1\alpha + \cdots + c_n\alpha^n = 0$, and therefore, the non-zero polynomial $g := \sum_i c_i \mathtt{X}^i$ is zero at α.

Example 17.13 The polynomial $\mathtt{X}^2 + 1$ is irreducible over $\mathbb{R}$, since if it were not, it would have a root in $\mathbb{R}$, which is clearly impossible, since -1 is not the square of any real number. It follows immediately that $\mathbb{C} := \mathbb{R}[\mathtt{X}]/(\mathtt{X}^2 + 1)$ is a field, without having to explicitly calculate a formula for the inverse of a non-zero complex number. □

Example 17.14 Consider the polynomial $f := \mathtt{X}^4 + \mathtt{X}^3 + 1$ over $\mathbb{Z}_2$. We claim that f is irreducible. It suffices to show that f has no irreducible factors of degree 1 or 2.

If f had a factor of degree 1, then it would have a root; however, $f(0) = 0 + 0 + 1 = 1$ and $f(1) = 1 + 1 + 1 = 1$. So f has no factors of degree 1.

Does f have a factor of degree 2? The polynomials of degree 2 are $\mathtt{X}^2$, $\mathtt{X}^2 + \mathtt{X}$, $\mathtt{X}^2 + 1$, and $\mathtt{X}^2 + \mathtt{X} + 1$. The first and second of these polynomials are divisible by $\mathtt{X}$, and hence not irreducible, while the third has a 1 as a root, and hence is also not irreducible. The last polynomial, $\mathtt{X}^2 + \mathtt{X} + 1$, has no roots, and hence is the only irreducible polynomial of degree 2 over $\mathbb{Z}_2$. So now we may conclude that if f were not irreducible, it would have to be equal to

$$(\mathtt{X}^2 + \mathtt{X} + 1)^2 = \mathtt{X}^4 + 2\mathtt{X}^3 + 3\mathtt{X}^2 + 2\mathtt{X} + 1 = \mathtt{X}^4 + \mathtt{X}^2 + 1,$$

which it is not.

Thus, $E := \mathbb{Z}_2[\mathtt{X}]/(f)$ is a field with $2^4 = 16$ elements. We may think of elements E as bit strings of length 4, where the rule for addition is bit-wise "exclusive-or." The rule for multiplication is more complicated: to multiply two given bit strings, we interpret the bits as coefficients of polynomials (with the left-most bit the coefficient of $\mathtt{X}^3$), multiply the polynomials, reduce the product modulo f, and write down the bit string corresponding to the reduced product polynomial. For example, to multiply `1001` and `0011`, we compute

$$(\mathtt{X}^3 + 1)(\mathtt{X} + 1) = \mathtt{X}^4 + \mathtt{X}^3 + \mathtt{X} + 1,$$

and

$$(\mathtt{X}^4 + \mathtt{X}^3 + \mathtt{X} + 1) \text{ rem } (\mathtt{X}^4 + \mathtt{X}^3 + 1) = \mathtt{X}.$$

Hence, the product of `1001` and `0011` is `0010`.

Theorem 10.2 says that E^* is a cyclic group. Indeed, the element $\eta :=$ `0010` (i.e., $\eta = [\mathtt{X} \bmod f]$) is a generator for E^*, as the following table of powers shows:

i	η^i	i	η^i
1	0010	8	1110
2	0100	9	0101
3	1000	10	1010
4	1001	11	1101
5	1011	12	0011
6	1111	13	0110
7	0111	14	1100
		15	0001

Such a table of powers is sometimes useful for computations in small finite fields such as this one. Given $\alpha, \beta \in E^*$, we can compute $\alpha\beta$ by obtaining (by table lookup) i, j such that $\alpha = \eta^i$ and $\beta = \eta^j$, computing $k := (i+j) \bmod 15$, and then obtaining $\alpha\beta = \eta^k$ (again by table lookup).

□

Example 17.15 In the field E is Example 17.14, what is the minimal polynomial of 1011 over $\mathbb{Z}_2$? □

Exercise 17.16 Show that if the factorization of f over $F[\mathtt{X}]$ into irreducibles is as $f = f_1^{e_1} \cdots f_r^{e_r}$, and if $\alpha = [h \bmod f] \in F[\mathtt{X}]/(f)$, then the minimal polynomial ϕ of α is $\mathrm{lcm}(\phi_1, \ldots, \phi_r)$, where ϕ_i is the minimal polynomial of $[h \bmod f_i^{e_i}] \in F[\mathtt{X}]/(f_i^{e_i})$. □

17.5 General Properties of Extension Fields

We now discuss a few general notions related to extension fields. These are all quite simple applications of the theory developed so far.

Let E be an extension field of a field F. Then E is an F-algebra, and in particular, an F-vector space. If E is a finite dimensional F-vector space, then we say that E is a **finite extension of** F, and $\dim_F(E)$ is called the **degree** of the extension, and is denoted $[E : F]$; otherwise, we say that E is an **infinite extension of** F.

An element $\alpha \in E$ is called **algebraic over** F if there exists a non-zero polynomial $f \in F[\mathtt{X}]$ such that $f(\alpha) = 0$; otherwise, α is called **transcendental over** F. If all elements of E are algebraic over F, then we call E an **algebraic extension of** F. From the discussion on minimal polynomials in §17.4, we may immediately state:

Theorem 17.17 *If E is a finite extension of F, then E is also an algebraic extension of F.*

Suppose $\alpha \in E$ is algebraic over F. Let ϕ be its minimal polynomial, so that $F[\mathtt{X}]/(\phi)$ is isomorphic (as an F-algebra) to $F[\alpha]$. Since $F[\alpha]$ is a subring of a field, it must be an integral domain, which implies that ϕ is irreducible, which in turn implies that $F[\alpha]$ is a a subfield of E. Moreover, the degree $[F[\alpha] : F]$ is equal to the degree of ϕ, and this number is called the **degree of α (over F)**. It is clear that if E is finite dimensional, then the degree of α is at most $[E : F]$.

Suppose that $\alpha \in E$ is transcendental over F. Consider the "rational function evaluation map" that sends $f/g \in F(\mathtt{X})$ to $f(\alpha)/g(\alpha) \in E$. Since no non-zero polynomial over F vanishes at α, it is easy to see that this map is well defined, and is in fact an injective F-algebra homomorphism from $F(\mathtt{X})$ into E. The image is denoted $F(\alpha)$, and this is clearly a subfield of E containing F and α, and it is plain to see that it is the smallest such subfield. It is also clear that $F(\alpha)$ has infinite dimension over F, since it contains an isomorphic copy of the infinite dimensional vector space $F[\mathtt{X}]$.

More generally, for any $\alpha \in E$, algebraic or transcendental, we can define $F(\alpha)$ to be the set consisting of all elements of the form $f(\alpha)/g(\alpha) \in E$, where $f, g \in F[\mathtt{X}]$ and $g(\alpha) \neq 0$. It is clear that $F(\alpha)$ is a field, and indeed, it is the smallest subfield of E containing F and α. If α is algebraic, then $F(\alpha) = F[\alpha]$, and is isomorphic (as an F-algebra) to $F[\mathtt{X}]/(\phi)$, where ϕ is the minimal polynomial of α over F; otherwise, if α is transcendental, then $F(\alpha)$ is isomorphic (as an F-algebra) to the rational function field $F(\mathtt{X})$.

Example 17.18 In the field E is Example 17.14, find all the elements of degree 2 over $\mathbb{Z}_2$. □

Example 17.19 If $f \in F[\mathtt{X}]$ is monic and irreducible, $E = F[\mathtt{X}]/(f)$, and $\eta := [\mathtt{X} \bmod f] \in E$, then η is algebraic over F, its minimal polynomial over F is f, and its degree over F is equal to $\deg(f)$. Also, we have $E = F[\eta]$, and any element $\alpha \in E$ is algebraic of degree at most $\deg(f)$. □

Exercise 17.20 Show that if E is a finite extension of F, with a basis $\alpha_1, \ldots, \alpha_n$ over F, and K is a finite extension of E, with a basis $\beta_1, \ldots, \beta_m$ over E, then

$$\alpha_i \beta_j \quad (i = 1, \ldots, n;\ j = 1, \ldots, m)$$

is a basis for K over F, and hence K is a finite extension of F and $[K : F] = [K : E][E : F]$. □

Exercise 17.21 Show that if E is an algebraic extension of F, and K is an algebraic extension of E, then K is an algebraic extension of F. □

Exercise 17.22 Let E be an extension of F. Show that the set of all elements in E that are algebraic over F is a subfield of E containing F. □

We close this section with a discussion of a **splitting field** — a finite extension of the coefficient field in which a given polynomial splits completely into linear factors. As the next theorem shows, splitting fields always exist.

Theorem 17.23 *Let F be a field, and $f \in F[\mathtt{X}]$ a monic polynomial of degree ℓ. Then there exists a finite extension E of F in which f factors as*

$$f = (\mathtt{X} - \alpha_1)(\mathtt{X} - \alpha_2) \cdots (\mathtt{X} - \alpha_\ell),$$

with $\alpha_1, \ldots, \alpha_\ell \in E$.

Proof. We prove the existence of E by induction on the degree ℓ of f. If $\ell = 0$, then the theorem is clearly true. Otherwise, let g be an irreducible factor of f, and set $K := F[\mathtt{X}]/(g)$, so that $\alpha := [\mathtt{X} \bmod g]$ is a root of g, and hence of f, in K. So over the field K, f factors as

$$f = (\mathtt{X} - \alpha)h,$$

where $h \in K[\mathtt{X}]$ is a polynomial of degree $\ell - 1$. Applying the induction hypothesis, there exists a finite extension E of K such that h splits into linear factors over K. Thus, over E, f into linear factors, and by Exercise 17.20, E is a finite extension of F. □

17.6 Formal Derivatives

Let R be any ring, and let $f \in R[\mathtt{X}]$ be a polynomial. If $f = \sum_{i=0}^{\ell} f_i \mathtt{X}^i$, we define the **formal derivative** of f as

$$\mathbf{D}(f) := \sum_{i=1}^{\ell} i f_i \mathtt{X}^{i-1}.$$

We stress that unlike the "analytical" notion of derivative from calculus, which is defined in terms of limits, this definition is purely "symbolic." Nevertheless, some of the usual rules for derivatives still hold:

Theorem 17.24 *For all $f, g \in R[\mathtt{X}]$ and $c \in R$, we have*

1. $\mathbf{D}(f + g) = \mathbf{D}(f) + \mathbf{D}(g)$;
2. $\mathbf{D}(cf) = c\mathbf{D}(f)$;
3. $\mathbf{D}(fg) = \mathbf{D}(f)g + f\mathbf{D}(g)$.

Proof. Parts (1) and (2) follow immediately by inspection, but part (3) requires some proof. First, note that part (3) holds trivially if either f or g are zero, so let us assume that neither are zero.

We first prove part (3) for monomials, i.e., polynomials of the form $c\mathtt{X}^i$ for non-zero $c \in R$ and $i \geq 0$. Suppose $f = c\mathtt{X}^i$ and $g = d\mathtt{X}^j$. If $i = 0$, so $f = c$, then the result follows from part (2) and the fact that $\mathbf{D}(c) = 0$; when $j = 0$, the result holds by a symmetric argument. So assume that $i > 0$ and $j > 0$. Now, $\mathbf{D}(f) = ic\mathtt{X}^{i-1}$ and $\mathbf{D}(g) = jd\mathtt{X}^{j-1}$, and $\mathbf{D}(fg) = \mathbf{D}(cd\mathtt{X}^{i+j}) = (i + j)cd\mathtt{X}^{i+j-1}$. The result follows from a simple calculation.

Having proved part (3) for monomials, we now prove it in general on induction on the total number of monomials appearing in f and g. If the total number is 2, then both f and g are monomials, and we are in the base case; otherwise, one of f and g must consist of at least 2 monomials, and for concreteness, say it is g that has this property. So we can write $g = g_1 + g_2$, where both g_1 and g_2 have fewer monomials than does g. Applying part (1)

and the induction hypothesis for part (3), we have

$$\begin{aligned}\mathbf{D}(fg) &= \mathbf{D}(fg_1 + fg_2)\\ &= \mathbf{D}(fg_1) + \mathbf{D}(fg_2)\\ &= \mathbf{D}(f)g_1 + f\mathbf{D}(g_1) + \mathbf{D}(f)g_2 + f\mathbf{D}(g_2)\\ &= \mathbf{D}(f)(g_1 + g_2) + f(\mathbf{D}(g_1) + \mathbf{D}(g_2))\\ &= \mathbf{D}(f)(g_1 + g_2) + f\mathbf{D}(g_1 + g_2)\\ &= \mathbf{D}(f)g + f\mathbf{D}(g).\end{aligned}$$

□

17.7 Formal Power Series and Laurent Series

We discuss generalizations of polynomials that allow an infinite number of non-zero coefficients. Although we are mainly interested in the case where the coefficients come from a field F, we develop the basic theory for general rings R.

17.7.1 Formal power series

The ring $R[\![\mathtt{X}]\!]$ of **formal power series over** R consists of all formal expressions of the form

$$a = a_0 + a_1\mathtt{X} + a_2\mathtt{X}^2 + \cdots,$$

where $a_0, a_1, a_2, \ldots \in R$. Unlike ordinary polynomials, we allow an infinite number of non-zero coefficients. We may write such a formal power series as

$$a = \sum_{i=0}^{\infty} a_i\mathtt{X}^i.$$

The rules for addition and multiplication of formal power series are *exactly* the same as for polynomials — indeed, the formulas (9.1) and (9.2) in §9.2 for addition and multiplication may be applied directly, with the observation that when applied to formal power series, the inner sum in (9.2) contains only finitely many non-zero terms, and so is well defined.

We shall not attempt to interpret a formal power series as a function, and therefore, "convergence" issues shall simply not arise.

Clearly, $R[\![\mathtt{X}]\!]$ contains $R[\mathtt{X}]$ as a subring. Let us consider the group of units of $R[\![\mathtt{X}]\!]$.

Theorem 17.25 *Let $a = \sum_{i=0}^{\infty} a_i\mathtt{X}^i \in R[\![\mathtt{X}]\!]$. Then $a \in (R[\![\mathtt{X}]\!])^*$ if and only if $a_0 \in R^*$.*

Proof. If a_0 is not a unit, then it is clear that a is not a unit, since the constant term of a product formal power series is equal to the product of the constant terms.

Conversely, if a_0 is a unit, we show how to define the coefficients of the inverse $b = \sum_{i=0}^{\infty} b_i\mathtt{X}^i$ of a. Let $ab = c = \sum_{i=1}^{\infty} c_i\mathtt{X}^i$. We want $c = 1$, i.e., $c_0 = 1$ and $c_i = 0$ for all

$i > 0$. Now, $c_0 = a_0 b_0$, so we set $b_0 := a_0^{-1}$. Next, we have $c_1 = a_0 b_1 + a_1 b_0$, so we set $b_1 := -a_1 b_0 \cdot a_0^{-1}$. Next, we have $c_2 = a_0 b_2 + a_1 b_1 + a_2 b_0$, so we set $b_2 := -(a_1 b_1 + a_2 b_0) \cdot a_0^{-1}$. Continuing in this way, we see that if we define $b_i := -(a_1 b_{i-1} + \cdots + a_i b_0) \cdot a_0^{-1}$ for $i \geq 1$, then $ab = 1$. □

Example 17.26 In the ring $R[\![\mathtt{X}]\!]$, the multiplicative inverse of $1 - \mathtt{X}$ is $\sum_{i=0}^{\infty} \mathtt{X}^i$. □

Exercise 17.27 For a field F, show that any non-zero ideal in $F[\![\mathtt{X}]\!]$ is of the form $(\mathtt{X}^m)$ for some $m \geq 0$. □

17.7.2 Formal Laurent series

One may generalize formal power series to allow a finite number of negative powers of $\mathtt{X}$. The ring $R(\!(\mathtt{X})\!)$ of **formal Laurent series over** R consists of all formal expressions of the form

$$a = a_m \mathtt{X}^m + a_{m+1} \mathtt{X}^{m+1} + \cdots,$$

where m is allowed to be any integer (possibly negative), and $a_m, a_{m+1}, \ldots \in R$. Thus, elements of $R(\!(\mathtt{X})\!)$ may have an infinite number of terms involving positive powers of $\mathtt{X}$, but only a finite number of terms involving negative powers of $\mathtt{X}$. We may write such a formal Laurent series as

$$a = \sum_{i=m}^{\infty} a_i \mathtt{X}^i.$$

Again, the rules for addition and multiplication of formal Laurent series are the same as for polynomials, using the formulas the formulas (9.1) and (9.2) in §9.2; and again, we observe that when applied to formal Laurent series, the inner sum in (9.2) contains only finitely many non-zero terms. Note that the fact that we do not allow an infinite number of terms involving *both* positive and negative powers of $\mathtt{X}$ is critical — without this restriction, the inner sum in (9.2) may have an infinite number of non-zero terms, and so multiplication would not be well defined.

We leave it to the reader to verify that $R(\!(\mathtt{X})\!)$ is a ring containing $R[\![\mathtt{X}]\!]$.

Theorem 17.28 *If D is an integral domain, then $D(\!(\mathtt{X})\!)$ is an integral domain.*

Proof. Let $a = \sum_{i=m}^{\infty} a_i \mathtt{X}^i$ and $b = \sum_{i=n}^{\infty} b_i \mathtt{X}^i$, where $a_m \neq 0$ and $b_n \neq 0$. Then $ab = \sum_{i=m+n}^{\infty} c_i$, where $c_{m+n} = a_m b_n \neq 0$. □

Theorem 17.29 *If F is a field, then $F(\!(\mathtt{X})\!)$ is a field.*

Proof. Consider any non-zero element $a = \sum_{i=m}^{\infty} a_i \mathtt{X}^i \in F(\!(\mathtt{X})\!)$, where $a_m \neq 0$. Then we can write $a = \mathtt{X}^m b$, where b is a formal power series with non-zero constant term, and hence there is a formal power series c such that $bc = 1$. Thus, $\mathtt{X}^{-m} c$ is a multiplicative inverse of a in $F(\!(\mathtt{X})\!)$. □

Exercise 17.30 Show that for a field F, $F(\!(\mathtt{X})\!)$ is the field of fractions of $F[\![\mathtt{X}]\!]$, i.e., there is no proper subfield of $F(\!(\mathtt{X})\!)$ that contains $F[\![\mathtt{X}]\!]$. □

17.7.3 Reversed formal Laurent series

While formal Laurent series are useful in some situations, in many others, it is more useful and natural to consider **reversed formal Laurent series over** R. These are formal expressions of the form

$$a = \sum_{i=-\infty}^{m} a_i \mathtt{X}^i,$$

where $a_m, a_{m-1}, \ldots \in R$. Thus, in a reversed formal Laurent series, we allow an infinite number of terms involving negative powers of $\mathtt{X}$, but only a finite number of terms involving positive powers of $\mathtt{X}$.

Again, the rules for addition and multiplication of reversed formal Laurent series are the same as for polynomials, using the formulas the formulas (9.1) and (9.2) in §9.2. The ring of all reversed formal Laurent series is denoted $R(\!(\mathtt{X}^{-1})\!)$, and as the notation suggests, the map that sends $\mathtt{X}$ to $\mathtt{X}^{-1}$ (and acts as the identity on R) is an isomorphism of $R(\!(\mathtt{X}^{-1})\!)$ with $R(\!(\mathtt{X})\!)$.

Now, for any $a = \sum_{i=-\infty}^{m} a_i \mathtt{X}^i \in R(\!(\mathtt{X}^{-1})\!)$ with $a_m \neq 0$, let us define the **degree of** a, denoted $\deg(a)$, to be the value m, and the **leading coefficient of** a, denoted $\mathrm{lc}(a)$, to be the value a_m. As for ordinary polynomials, we define the degree of 0 to be $-\infty$, and the leading coefficient of 0 to be 0. Note that if a happens to be a polynomial, then these definitions of degree and leading coefficient agree with that for ordinary polynomials.

Theorem 17.31 *For $a, b \in R(\!(\mathtt{X}^{-1})\!)$, we have $\deg(ab) \leq \deg(a) + \deg(b)$, where equality holds unless both $\mathrm{lc}(a)$ and $\mathrm{lc}(b)$ are zero divisors. Furthermore, if $b \neq 0$ and $\mathrm{lc}(b)$ is a unit, then b is a unit, and we have $\deg(a/b) = \deg(a) - \deg(b)$.*

Proof. Exercise. □

It is also natural to define a "floor function" for reversed formal Laurent series: for $a \in R(\!(\mathtt{X}^{-1})\!)$ with $a = \sum_{i=-\infty}^{m} a_i \mathtt{X}^i$, we define

$$\lfloor a \rfloor := \sum_{i=0}^{m} a_i \mathtt{X}^i \in R[\mathtt{X}];$$

that is, we compute the floor function by simply throwing away all terms involving negative powers of $\mathtt{X}$.

Now, let $a, b \in R[\mathtt{X}]$ with $b \neq 0$ and $\mathrm{lc}(b)$ a unit, and write $a = bq + r$, where $q, r \in R[\mathtt{X}]$ with $\deg(r) < \deg(b)$. We can form the quotient $a/b \in R(\!(\mathtt{X}^{-1})\!)$ and apply the floor function to obtain $\lfloor a/b \rfloor \in R[\mathtt{X}]$. It is not too hard to see that $\lfloor a/b \rfloor = q$; indeed, dividing the equation $a = bq + r$ by q inside the field $R(\!(\mathtt{X}^{-1})\!)$, we obtain $a/b = q + r/b$, and $\deg(r/b) < 0$, from which it follows that $\lfloor a/b \rfloor = q$.

Let F be a field. Since $F(\!(\mathtt{X}^{-1})\!)$ is isomorphic to $F(\!(\mathtt{X})\!)$, and the latter is a field, it follows that $F(\!(\mathtt{X}^{-1})\!)$ is a field. Now, $F(\!(\mathtt{X}^{-1})\!)$ contains $F[\mathtt{X}]$ as a subring, and hence contains

(an isomorphic copy) of $F(\mathtt{X})$. Just as $F(\mathtt{X})$ corresponds to the field of rational numbers, $F(\!(\mathtt{X}^{-1})\!)$ corresponds to the field real numbers. Indeed, we can think of real numbers as decimal numbers with a finite number of digits to the left of the decimal point and an infinite number to the right, and reversed formal Laurent series have a similar "syntactic" structure. In many ways, this syntactic similarity between the real numbers and reversed formal Laurent series is more than just superficial.

Exercise 17.32 Write down the rule for determining the multiplicative inverse of an element of $R(\!(\mathtt{X}^{-1})\!)$ whose leading coefficient is a unit in R. □

Exercise 17.33 Let F be a field of characteristic other than 2. Show that a non-zero $z \in F(\!(\mathtt{X}^{-1})\!)$ has a square-root in $z \in F(\!(\mathtt{X}^{-1})\!)$ if and only if $\deg(z)$ is even and $\mathrm{lc}(z)$ has a square-root in F. □

Exercise 17.34 Let R be a ring, and let $\alpha \in R$. Show that the multiplicative inverse of $\mathtt{X} - \alpha$ in $R(\!(\mathtt{X}^{-1})\!)$ is $\sum_{j=1}^{\infty} \alpha^{j-1}\mathtt{X}^{-j}$. □

Exercise 17.35 Let R be an arbitrary ring, let $\alpha_1, \ldots, \alpha_\ell \in R$, and

$$f := (\mathtt{X} - \alpha_1)(\mathtt{X} - \alpha_2) \cdots (\mathtt{X} - \alpha_\ell).$$

For $j \geq 0$, define the "power sum"

$$s_j := \sum_{i=1}^{\ell} \alpha_i^j.$$

Show that in the ring $R(\!(\mathtt{X}^{-1})\!)$, we have

$$\frac{\mathbf{D}(f)}{f} = \sum_{i=1}^{\ell} \frac{1}{(\mathtt{X} - \alpha_i)} = \sum_{j=1}^{\infty} s_{j-1}\mathtt{X}^{-j}.$$

□

Exercise 17.36 Continuing with the previous exercise, derive **Newton's identities**, which state that if $f = \mathtt{X}^\ell + f_1\mathtt{X}^{\ell-1} + \cdots + f_\ell$, with $f_1, \ldots, f_\ell \in R$, then then

$$\begin{aligned} s_1 + f_1 &= 0 \\ s_2 + f_1 s_1 + 2f_2 &= 0 \\ s_3 + f_1 s_2 + f_2 s_1 + 3f_3 &= 0 \\ &\vdots \\ s_\ell + f_1 s_{\ell-1} + \cdots + f_{\ell-1} s_1 + \ell f_\ell &= 0. \end{aligned}$$

□

17.8 ♣ Unique Factorization Domains

As we have seen, both the integers and the ring $F[\mathtt{X}]$ of polynomials over a field enjoy a unique factorization property. These are special cases of a more general phenomenon, which we explore here.

Throughout this section, D denotes an integral domain.

We call $a, b \in D$ **associates** if $a = bu$ for some $u \in D^*$. A non-zero element $p \in D$ is called **irreducible** if it is not a unit, and all divisors of p are associate to 1 or p.

Definition 17.37 *We call D a **unique factorization domain (UFD)** if*

1. *every non-zero element of D that is not a unit can be written as a product of irreducibles in D, and*
2. *the factorization into irreducibles is unique up to associates and the order in which the factors appear.*

Another way to state part (2) of the above definition is that if $p_1 \cdots p_r$ and $p'_1 \cdots p'_s$ are two factorizations of some element as a product of irreducibles, then $r = s$, and there exists a permutation π on the indices $\{1, \ldots, r\}$ such that p_i and $p'_{\pi(i)}$ are associate.

As we have seen, both $\mathbb{Z}$ and $F[\mathtt{X}]$ are UFDs. In both of those cases, we chose to single out a special irreducible element among all those associate to any given irreducible: for $\mathbb{Z}$, we always chose p to be positive, and for $F[\mathtt{X}]$, we chose p to be monic. For any specific unique factorization domain D, there may be such a natural choice, but in the general case, there will not be.

Example 17.38 Having already seen two examples of UFDs, it is perhaps a good idea to look at an example of an integral domain that is not a UFD. Consider the subring $\mathbb{Z}[\sqrt{-5}]$ of the complex numbers, which consists an all complex numbers of the form $a + b\sqrt{-5}$, where $a, b \in \mathbb{Z}$. As this is a subring of the field $\mathbb{C}$, it is an integral domain (one may also view $\mathbb{Z}[\sqrt{-5}]$ as the quotient ring $\mathbb{Z}[\mathtt{X}]/(\mathtt{X}^2 + 5)$).

Let us first determine the units in $\mathbb{Z}[\sqrt{-5}]$. For $a, b \in \mathbb{Z}$, we have $N(a+b\sqrt{-5}) = a^2+5b^2$, where N is the usual norm map on $\mathbb{C}$. If $z \in \mathbb{Z}[\sqrt{-5}]$ is a unit, then there exists $z' \in \mathbb{Z}[\sqrt{-5}]$ such that $zz' = 1$. Taking norms, we obtain

$$1 = N(1) = N(zz') = N(z)N(z').$$

Since the norm of an element of $\mathbb{Z}[\sqrt{-5}]$ is a non-negative integer, this implies that $N(z) = 1$. If $z = a + b\sqrt{-5}$, then $N(z) = a^2 + 5b^2$, and it is clear that $N(z) = 1$ if and only if $z = \pm 1$. We conclude that the only units in $\mathbb{Z}[\sqrt{-5}]$ are ± 1.

Now consider the following factorizations:

$$\begin{aligned} 46 &= 2 \cdot 23, \\ 46 &= (1 + 3\sqrt{-5})(1 - 3\sqrt{-5}). \end{aligned}$$

We claim that each of these four factors are irreducibles in $\mathbb{Z}[\sqrt{-5}]$. For suppose, say, that $2 = zz'$, for $z, z' \in \mathbb{Z}[\sqrt{-5}]$, with neither a unit. Taking norms, we have $4 = N(2) = N(z)N(z')$, and therefore, $N(z) = N(z') = 2$ — but this is impossible, since there are no integers a and b such that $a^2 + 5b^2 = 2$. Analogous arguments apply to the other three factors, which we leave to the reader. Since the only units in $\mathbb{Z}[\sqrt{-5}]$ are ± 1, it is clear that these four irreducibles are non-associate. □

For $a, b \in D$, we call $d \in D$ a **common divisor** of a and b if $d \mid a$ and $d \mid b$; moreover, we call such a d a **greatest common divisor** of a and b if all other common divisors of a and b divide d. We say that a and b are **relatively prime** if the only common divisors of a and b are units. It is immediate from the definition of a greatest common divisor that it is unique, up to multiplication by units, if it exists at all. For general integral domains D, greatest common divisors need not exist. Unlike in the case of $\mathbb{Z}$ and $F[\mathtt{X}]$, in the general setting, we shall not attempt to "normalize" greatest common divisors, and we speak only of "a" greatest common divisor, rather than "the" greatest common divisor.

Just as for integers and polynomials, we can generalize the notion of a greatest common divisor in an arbitrary integral domain D from two to any number of elements of D, and we can also define a least common multiple of any number of elements as well.

These greatest common divisors and least common multiples need not exist, but if they do, they are unique up to associates. If D is a UFD, then they will always exist. The existence question easily reduces to the question of the existence of a greatest common divisor and least common multiple of a and b, where a and b are non-zero elements of D. If we write

$$a = u \prod_{i=1}^{r} p_i^{e_i} \quad \text{and} \quad b = v \prod_{i=1}^{r} p_i^{f_i},$$

where u and v are units, $p_1, \ldots, p_r$ are non-associate irreducibles, and the e_i's and f_i's are non-negative integers, then

$$\prod_{i=1}^{r} p^{\min(e_i, f_i)}$$

is a greatest common divisor of a and b, while

$$\prod_{i=1}^{r} p^{\max(e_i, f_i)}$$

is a least common multiple of a and b.

It is also evident that in a UFD D, if $c \mid ab$ and c and a are relatively prime, then $c \mid b$. In particular, if p is irreducible and $p \mid ab$, then $p \mid a$ or $p \mid b$. From this, we see that if p is irreducible, then the quotient ring $D/(p)$ is an integral domain, and so the ideal (p) is a prime ideal (see Exercise 9.53).

In a general integral domain D, we say that an element $p \in D$ is **prime** if for all $a, b \in D$, $p \mid ab$ implies $p \mid a$ or $p \mid b$. Thus, if D is a UFD, then all irreducibles are primes; however, in a general integral domain, this may not be the case. Here are a couple of simple but useful facts whose proofs we leave to the reader.

Theorem 17.39 *Any prime element in D is irreducible.*

Proof. Exercise. □

Theorem 17.40 *Suppose D satisfies part (1) of Definition 17.37. Also, suppose that all irreducibles in D are prime. Then D is a UFD.*

Proof. Exercise. □

Exercise 17.41 Let D be a UFD and F its field of fractions. Show that

(a) every element $x \in F$ can be expressed as $x = a/b$, where $a, b \in D$ are relatively prime, and

(b) that if $x = a/b$ for $a, b \in D$ relatively prime, then for any other $a', b' \in D$ with $x = a'/b'$, we have $a' = ca$ and $b' = cb$ for some $c \in D$.

□

17.8.1 Unique factorization in Euclidean and Principal Ideal Domains

Our proofs of the unique factorization property in both $\mathbb{Z}$ and $F[\mathtt{X}]$ hinged on the division with remainder property for these rings. This notion can be generalized, as follows.

Definition 17.42 *D is said to be a **Euclidean domain** if there is a function λ mapping the non-zero elements of D to the set of non-negative integers, such that for $a, b \in D$ with $b \neq 0$, there exist $q, r \in D$, with the property that $a = bq + r$ and either $r = 0$ or $\lambda(r) < \lambda(b)$.*

Example 17.43 Both $\mathbb{Z}$ and $F[\mathtt{X}]$ are Euclidean domains. In $\mathbb{Z}$, we can take the ordinary absolute value function $|\cdot|$ as λ, and for $F[\mathtt{X}]$, the function $\deg(\cdot)$ will do. □

Example 17.44 Recall again the ring

$$\mathbb{Z}[i] = \{a + bi : a, b \in \mathbb{Z}\}$$

of Gaussian integers from Example 9.27. This is a Euclidean domain, using the usual norm map N on complex numbers for the function λ. Let $z, w \in \mathbb{Z}[i]$, with $w \neq 0$. We want to show the existence of $u, v \in \mathbb{Z}[i]$ such that $z = uw + v$, where $N(v) < N(w)$. Suppose that in the field $\mathbb{C}$, we compute $zw^{-1} = r + si$, where $r, s \in \mathbb{Q}$. Let m, n be integers such that $|m - r| \leq 1/2$ and $|n - s| \leq 1/2$ — such integers m and n always exist, but may not be uniquely determined. Set $u := m + ni \in \mathbb{Z}[i]$ and $v := z - uw$. Then we have

$$zw^{-1} = u + \delta,$$

where $\delta \in \mathbb{C}$ with $N(\delta) \leq 1/4$, and

$$v = z - uw = z - (zw^{-1} - \delta)w = \delta w,$$

and hence

$$N(v) = N(\delta w) = N(\delta)N(w) \leq \frac{1}{4}N(w).$$

□

Theorem 17.45 *If D is a Euclidean domain and I is an ideal in D, then there exists $d \in D$ such that $I = (d)$.*

Proof. If $I = \{0\}$, then $d = 0$ does the job, so let us assume that $I \neq \{0\}$. Let d be an non-zero element of I such that $\lambda(d)$ is minimal. We claim that $I = (d)$.

It will suffice to show that for all $c \in I$, we have $d \mid c$. Now, we know that there exists $q, r \in D$ such that $c = qd + r$, where either $r = 0$ or $\lambda(r) < \lambda(d)$. If $r = 0$, we are done; otherwise, r is a non-zero element of I with $\lambda(r) < \lambda(d)$, contradicting the minimality of $\lambda(d)$. □

Recall that an ideal of the form $I = (d)$ is called a principal ideal. If all ideals in D are principal, then D is called a **principal ideal domain (PID)**. Theorem 17.45 says that any Euclidean domain is a PID.

PIDs enjoy many nice properties, including:

Theorem 17.46 *If D is a PID, then D is a UFD.*

For the rings $\mathbb{Z}$ and $F[\mathtt{X}]$, the proof of part (1) of Definition 17.37 was quite straightforward (as it also would be for any Euclidean domain). For a general PID, however, this requires a different sort of argument. We begin with the following fact:

Theorem 17.47 *If D is a PID, and $I_1 \subset I_2 \subset \cdots$ is an ascending chain of ideals in D, then there exists an integer k such that $I_k = I_{k+1} = \cdots$.*

Proof. Let $I := \cup_{i=1}^{\infty} I_i$. It is easy to see that I is an ideal. Thus, $I = (d)$ for some $d \in D$. But $d \in \cup_{i=1}^{\infty} I_i$ implies that $d \in I_k$ for some k, which shows that $I = (d) \subset I_k$. It follows that $I = I_k = I_{k+1} = \cdots$. □

We can now prove the existence part of Theorem 17.46:

Theorem 17.48 *If D is a PID, then every non-zero, non-unit element of D can be expressed as a product of irreducibles in D.*

Proof. Let $n \in D$, $n \neq 0$, and n not a unit. If n is irreducible, we are done. Otherwise, we can write $n = ab$, where neither a nor b are units. As ideals, we have $(n) \subsetneq (a)$ and $(n) \subsetneq (b)$. If we continue this process recursively, building up a "factorization tree" where n is at the root, a and b are the children of n, and so on, then the recursion must stop, since any infinite path in the tree would give rise to a chain of ideals

$$(n) = I_1 \subsetneq I_2 \subsetneq \cdots,$$

contradicting Theorem 17.47. □

The proof of the uniqueness part of Theorem 17.46 is essentially the same as for proofs we gave for $\mathbb{Z}$ and $F[\mathtt{X}]$.

Analogous to Theorems 1.11 and 17.5, we have:

Theorem 17.49 *Let D be a PID. For any $a, b \in D$, there exists a greatest common divisor d of a and b, and moreover, $(a, b) = (d)$; in particular, $as + bt = d$ for some $s, t \in D$.*

Proof. Exercise. □

The previous theorem says that in a PID, a and b are relatively prime if and only if there exist $s, t \in D$ such that $as + bt = 1$.

Analogous to Theorems 1.12 and 17.6, we have:

Theorem 17.50 *Let D be a PID. For $a, b, c \in D$ such that $c \mid ab$ and a and c are relatively prime, we have $c \mid b$.*

Proof. Exercise. □

Analogous to Theorems 1.13 and 17.51, we have:

Theorem 17.51 *Let D be a PID. Let $p \in D$ be irreducible, and let $a, b \in D$. Then $p \mid ab$ implies that $p \mid a$ or $p \mid b$. That is, all irreducibles in D are prime.*

Proof. Exercise. □

Theorem 17.46 now follows immediately from Theorems 17.48, 17.51, and 17.40.

Exercise 17.52 Consider the polynomial

$$\mathtt{X}^3 - 1 = (\mathtt{X} - 1)(\mathtt{X}^2 + \mathtt{X} + 1).$$

Over $\mathbb{C}$, the roots of $\mathtt{X}^3 - 1$ are $1, (-1 \pm \sqrt{-3})/2$. Let $\omega = (-1 + \sqrt{-3})/2$, and note that $\omega^2 = (-1 - \sqrt{-3})/2$, and $\omega^3 = 1$.

(a) Show that the ring $\mathbb{Z}[\omega]$ consists of all elements of the form $a + b\omega$, where $a, b \in \mathbb{Z}$, and is an integral domain.

(b) Determine the units in $\mathbb{Z}[\omega]$.

(c) Show that $\mathbb{Z}[\omega]$ is a Euclidean domain.

□

Exercise 17.53 Design and analyze an efficient algorithm to compute a greatest common divisor of two Gaussian integers. □

17.8.2 Unique factorization in $D[\mathtt{X}]$

In this section, we prove the following:

Theorem 17.54 *If D is a UFD, then so is $D[\mathtt{X}]$.*

This theorem implies, for example, that $\mathbb{Z}[\mathtt{X}]$ is a UFD. Applying the theorem inductively, one also sees that for any field F, the ring $F[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ of multi-variate polynomials over F is also a UFD.

We begin with some simple observations. First, recall that for an integral domain D, $D[\mathtt{X}]$ is an integral domain, and the units in $D[\mathtt{X}]$ are precisely the units in D. Second, it is easy to see that an element of D is irreducible in D if and only if it is irreducible in $D[\mathtt{X}]$. Third, for $c \in D$ and $f = \sum_i a_i \mathtt{X}^i \in D[\mathtt{X}]$, we have $c \mid f$ if and only if $c \mid a_i$ for all i.

We call a non-zero polynomial $f \in F[\mathtt{X}]$ **primitive** if the only elements in D that divide f are units. A couple of simple observations, which hold for any integral domain D, are the following:

- a primitive polynomial that is not a unit has degree greater than zero, i.e., it is not a constant;
- any non-constant irreducible polynomial must be primitive.

If D is a UFD, then given any non-zero polynomial $f \in D[\mathtt{X}]$, we can partially factor it as $f = cf'$, where $c \in D$ and f' is a primitive polynomial — just take c to be a greatest common divisor of all the coefficients of f. These values c and f' are uniquely determined, up to associates, and are called, respectively, a **content of f** and a **primitive part of f**.

It is easy to prove the existence part of Theorem 17.54:

Theorem 17.55 *Let D be a UFD. Any non-zero, non-unit element of $D[\mathtt{X}]$ can be expressed as a product of irreducibles in $D[\mathtt{X}]$.*

Proof. For a non-zero, non-unit polynomial $f \in D[\mathtt{X}]$, write $f = cf'$ for $c \in D$ and f' primitive. Since D is a UFD, we know that c factors into irreducibles, and so it suffices to consider only primitive polynomials. So assume that f is a primitive polynomial. If f is irreducible, we are done. Otherwise, we can write $f = gh$, where neither g nor h are units. Since f is primitive and not a unit, it must not be a constant. It must also be the case that g and h are primitive, non-constant polynomials, both of degree strictly less than that of f. By induction on degree, both g and h can be expressed as the product of irreducible, non-constant polynomials, and hence, so can f. □

The uniqueness part of Theorem 17.54 is (as usual) more difficult. We begin with the following fact:

Theorem 17.56 *Let D be a UFD. The product of two primitive polynomials in $D[\mathtt{X}]$ is also primitive.*

Proof. Let $f, g \in D[\mathtt{X}]$ be primitive polynomials, and let $h := fg$. If h is not primitive, then $m \mid h$ for some non-zero, non-unit $m \in D$, and as D is a UFD, there is some irreducible element $p \in D$ that divides m, and therefore, divides h as well. Consider the quotient ring $D/(p)$, which is an integral domain (because D is a UFD), and the corresponding ring of polynomials $D/(p)[\mathtt{X}]$, which is also an integral domain. Consider the natural homomorphism from $D[\mathtt{X}]$ to $D/(p)[\mathtt{X}]$ that sends $a \in D[\mathtt{X}]$ to the polynomial $\bar{a} \in D/(p)[\mathtt{X}]$ obtained by mapping each coefficient of a to its residue class mod p. Then we have

$$0 = \bar{h} = \overline{fg} = \bar{f}\bar{g},$$

and since $D/(p)[\mathtt{X}]$ is an integral domain, it follows that $\bar{f} = 0$ or $\bar{g} = 0$, which means that $p \mid f$ or $p \mid g$. This contradicts the assumption that f and g are primitive. □

As a generalization of the previous theorem, we have:

Theorem 17.57 *Let D be a UFD. If f and g are non-zero polynomials in $D[\mathtt{X}]$, and a is a content of f and b is a content of g, then ab is a content of fg.*

Proof. Write $f = af'$ and $g = bg'$, where f' and g' are primitive polynomials. Then we have $fg = (ab)(f'g')$. By the previous theorem, $f'g'$ is primitive, and so ab is a content of fg. □

Theorem 17.58 *Let D be a UFD and F its field of fractions. If $h \in D[\mathtt{X}]$ with $\deg(h) > 0$ is irreducible, then h is also irreducible in $F[\mathtt{X}]$.*

Proof. Suppose that h is irreducible in $D[\mathtt{X}]$, but not in in $F[\mathtt{X}]$, so that $h = fg$ for non-constant polynomials $f, g \in F[\mathtt{X}]$, both of degree strictly less than that of h. Each coefficient of f is a fraction, with numerator and denominator in D. Let a be the product of all these denominators, so that $f_0 := af \in D[\mathtt{X}]$. Likewise, let b be the product of all the denominators of the coefficients of g, so that $g_0 := bg \in D[\mathtt{X}]$. Then we have $abh = f_0 g_0$. Let us write $f_0 = cf_1$ and $g_0 = dg_1$, where $c, d \in D$ and f_1 and g_1 are primitive. Then we have $(ab)h = (cd)(f_1 g_1)$. Now, since h is irreducible and non-constant, it must be primitive, and therefore ab is a content of $(ab)h$. Also, by the Theorem 17.56, the polynomial $f_1 g_1$ is primitive, and so cd is a content of $(cd)(f_1 g_1)$. Since ab and cd are contents of the same polynomial, it follows that ab and cd are associate, and hence $cd = uab$ for some unit u, from which it follows that $h = uf_1 g_1$. This contradicts the assumption that h is irreducible in $D[\mathtt{X}]$. □

Theorem 17.59 *Let D be a UFD. If $h \in D[\mathtt{X}]$ is irreducible, and $f, g \in D[\mathtt{X}]$, then $h \mid fg$ implies $h \mid f$ or $h \mid g$. That is, every irreducible in $D[\mathtt{X}]$ is prime.*

Proof. We may assume that neither f nor g are zero, as otherwise, the theorem is trivial.

Let us consider two cases. In the first case, suppose that $\deg(h) = 0$, i.e, $h \in D$. If a is a content of f and b is a content of g, then by Theorem 17.57, ab is a content of fg. Now,

if $h \mid fg$, then h divides the content of fg, so $h \mid ab$, and since h is irreducible, $h \mid a$ or $h \mid b$, which implies that $h \mid f$ or $h \mid g$.

In the second case, suppose that $\deg(h) > 0$. By the previous theorem, h is irreducible in $F[\mathtt{X}]$, and by unique factorization in $F[\mathtt{X}]$, we have $f = hf'$ for some $f' \in F[\mathtt{X}]$, or $g = hg'$ for some $g' \in F[\mathtt{X}]$. Suppose that $f = hf'$ for $f' \in F[\mathtt{X}]$ — the proof is analogous in the other situation. Let us choose $c \in D$ to clear the denominators of the coefficients of f', so that $cf = hf'_0$, where $f'_0 \in D[\mathtt{X}]$. Now, if d is a content of f'_0, then since h must be primitive, it follows from Theorem 17.57 that d is a content of hf'_0, and hence a content of cf. As any content of cf is a multiple of c, it follows that $c \mid d$. Canceling c, we obtain $f = h(f'_0/c)$, where $f'_0/c \in D[\mathtt{X}]$, which proves the theorem. □

Theorem 17.54 now follows immediately from Theorems 17.55, 17.59, and 17.40.

In the proof of Theorem 17.54, there is clearly a connection between factorization in $D[\mathtt{X}]$ and $F[\mathtt{X}]$, where F is the field of fractions of D. We should perhaps make this connection more explicit. Suppose $f \in D[\mathtt{X}]$ factors into irreducibles in $D[\mathtt{X}]$ as

$$f = c_1^{a_1} \cdots c_r^{a_r} h_1^{b_1} \cdots h_s^{b_s}.$$

where the c_i's are non-associate, irreducible constants, and the h_i's are non-associate, irreducible, non-constant polynomials. By Theorem 17.58, the h_i's are irreducible in $F[\mathtt{X}]$. Moreover, the h_i's are not associate in $F[\mathtt{X}]$ (see Exercise 17.63 below), and thus in $F[\mathtt{X}]$, f factors as

$$f = ch_1^{b_1} \cdots h_s^{b_s},$$

where $c := c_1^{a_1} \cdots c_r^{a_r}$ is a unit in F, and the h_i's are non-associate irreducible polynomials in $F[\mathtt{X}]$.

Example 17.60 It is important to keep in mind the distinction between factorization in $D[\mathtt{X}]$ and $F[\mathtt{X}]$. Consider the polynomial $2\mathtt{X}^2 - 2 \in \mathbb{Z}[\mathtt{X}]$. Over $\mathbb{Z}[\mathtt{X}]$, this polynomial factors as $2(\mathtt{X}-1)(\mathtt{X}-1)$, where each of these three factors are irreducible in $\mathbb{Z}[\mathtt{X}]$. Over $\mathbb{Q}[\mathtt{X}]$, this polynomials has two irreducible factors, namely, $\mathtt{X}-1$ and $\mathtt{X}+1$. □

The following theorem provides a useful criterion for establishing that a polynomial is irreducible.

Theorem 17.61 (Eisenstein's Criterion) *Let D be a UFD and F its field of fractions. Let $f = a_n\mathtt{X}^n + a_{n-1}\mathtt{X}^{n-1} + \cdots + a_0 \in D[\mathtt{X}]$. If there exists an irreducible $p \in D$ such that*

$$p \nmid a_n,\ p \mid a_{n-1},\ \cdots,\ p \mid a_0,\ p^2 \nmid a_0,$$

then f is irreducible over F.

Proof. Suppose $f = gh$, where $r := \deg(g) < n$ and $s := \deg(h) < n$. Let us write

$$g = \sum_{i=0}^{r} b_i\mathtt{X}^i \quad \text{and} \quad h = \sum_{i=0}^{s} c_i\mathtt{X}^i.$$

Since $p \mid a_0 = b_0 c_0$, but $p^2 \nmid a_0$, it follows that p divides one of b_0 or c_0, but not both. Let us say $p \mid b_0$ and $p \nmid c_0$. Also, since $p \nmid a_n = b_r c_s$, we know that $p \nmid b_r$. So there is a least non-negative integer t such that $p \nmid b_t$, and this t satisfies $0 < t \leq r < n$. Now consider $a_t = b_t c_0 + b_{t-1} c_1 + \cdots + b_0 c_t$. By assumption, $p \mid a_t$ and by the choice of t, every term on the right after the first one is also divisible by p, which forces p to divide $b_t c_0$ as well. But this is impossible, since p divides neither b_t nor c_0. □

As an application of the previous theorem, we have:

Theorem 17.62 *For any prime number q, the qth cyclotomic polynomial*

$$\Phi_q = \frac{\mathtt{X}^q - 1}{\mathtt{X} - 1} = \mathtt{X}^{q-1} + \mathtt{X}^{q-2} + \cdots + 1$$

is irreducible over $\mathbb{Q}$.

Proof. Let

$$f = \Phi_q(\mathtt{X} + 1) = \frac{(\mathtt{X} + 1)^q - 1}{(\mathtt{X} + 1) - 1}.$$

It is easy to see that

$$f = \sum_{i=0}^{q-1} a_i \mathtt{X}_i, \quad \text{where} \quad a_i = \binom{q}{i+1} \quad (i = 0, \ldots, q-1).$$

Thus, $a_{q-1} = 1$, $a_0 = q$, and for $0 < i < q - 1$, we have $q \mid a_i$ (see Exercise 1.16). Theorem 17.61 therefore applies, and we conclude that f is irreducible over $\mathbb{Q}$. It follows that Φ_q is irreducible over $\mathbb{Q}$, since if $\Phi_q = gh$ were a non-trivial factorization of Φ_q, then $f = \Phi_q(\mathtt{X} + 1) = g(\mathtt{X} + 1)h(\mathtt{X} + 1)$ would be a non-trivial factorization of f. □

Exercise 17.63 Suppose that D is a UFD, F its field of fractions, and $f, g \in D[\mathtt{X}]$ are primitive polynomials $f = cg$ for some $c \in F$. Show that $c \in D^*$. □

Exercise 17.64 Show that neither $\mathbb{Z}[\mathtt{X}]$ nor $F[\mathtt{X}, \mathtt{Y}]$ (where F is a field) are PIDs (even though they are UFDs). □

Exercise 17.65 Show that the polynomial $\mathtt{X}^4 + 1$ is irreducible in $\mathbb{Q}[\mathtt{X}]$. □

Exercise 17.66 Design and analyze an efficient algorithm for the following problem. The input is a pair of polynomials $a, b \in \mathbb{Z}[\mathtt{X}]$, along with their greatest common divisor d *in the ring* $\mathbb{Q}[\mathtt{X}]$ — d is a normalized polynomial (i.e., zero or monic) with rational coefficients represented as fractions in lowest terms. The output is the greatest common divisor of a and b the the ring $\mathbb{Z}[\mathtt{X}]$. □

17.9 ♣ Constructing the Real Numbers

It is instructive to see how the language and techniques of the theory of rings can be used to define the real numbers $\mathbb{R}$, starting from the rational numbers $\mathbb{Q}$. The purpose of this section is mainly to illustrate concepts from algebra, and not to do any serious analysis; moreover, we leave most of the technical details as exercises to the reader.

First, let us define the ring S of all infinite sequences $(a_1, a_2, \ldots)$ of rational numbers, where addition and multiplication are defined component-wise.

Exercise 17.67 Show that S is indeed a ring, where the additive identity 0_S is the "all zero" sequence, and the multiplicative identity 1_S is the "all one" sequence. □

We now make some "analytical" definitions. Let $x = (a_1, a_2, \ldots) \in S$.

- We say that x is **bounded** if there exists a rational number b such that $|a_i| \leq b$ for all $i \geq 1$.
- We say that x is **null** if for all rational $\epsilon > 0$, there exists integer $n \geq 1$, such that for all $i \geq n$, we have $|a_i| < \epsilon$.
- We say that x is **positive** if there exists a rational number $\delta > 0$ and an integer $n \geq 1$, such that for all $i \geq n$, we have $a_i > \delta$.
- We say that x is **negative** if there exists a rational number $\delta < 0$ and an integer $n \geq 1$, such that for all $i \geq n$, we have $a_i < \delta$.
- We say that x is **Cauchy** if for all rational $\epsilon > 0$, there exists an integer $n \geq 1$, such that for all $i, j \geq n$, we have $|a_i - a_j| < \epsilon$.

Exercise 17.68 Show that

(a) every null sequence is Cauchy,

(b) every Cauchy sequence is bounded,

(c) every Cauchy sequence is either positive, negative, or null,

(d) the sum of two positive (resp., negative) sequences is positive (resp., negative), the product of two positive (resp., negative) sequences is positive, and the product of a positive and a negative sequence is negative.

□

Let C be the subset of C consisting of all Cauchy sequences, and let N be the subset of S consisting of all null sequences. By the previous exercise, we have $N \subset C$.

Exercise 17.69 Show that

(a) C is a subring of S, and

(b) N is an ideal in C.

□

Because N is an ideal in S, we can form the quotient ring C/N. This will be our definition of the real numbers. We first need to show that C/N is a field.

Exercise 17.70 Let $x = (a_1, a_2, \ldots)$ be a non-null Cauchy sequence. Define $x' := (a'_1, a'_2, \ldots)$, where $a'_i := a_i^{-1}$, if $a_i \neq 0$, and $a'_i := 0$, otherwise. Show that x' is Cauchy and that $xx' \equiv 1_S \pmod{N}$. Conclude that C/N is a field. □

Of course, we want to view the rationals as a subfield of the reals:

Exercise 17.71 Show that the map $\rho : \mathbb{Q} \to S$ that sends $a \in \mathbb{Q}$ to the sequence $(a, a, \ldots)$ is a homomorphism. Also, show that ρ is injective, and that its image is contained in C. Further, show that $\rho^{-1}(N) = \{0\}$, and from this, conclude that the composition of ρ with the natural map from C to C/N is an embedding of $\mathbb{Q}$ into C/N. □

Of course, the real numbers are a special type of field in that they come equipped with a total order "$<$." We can define a total order on C/N, in terms of the usual "$<$" relation on $\mathbb{Q}$.

Exercise 17.72 Show that if x is a positive (resp., negative) Cauchy sequence, then every element of the coset $x + N$ is also positive (resp., negative). □

We can define an element $x + N$ of C/N to be **positive** if x is positive, and to be **negative** if x is negative. Because of the previous exercise, this definition does not depend on the choice of x, and so is unambiguous. Because of part (c) of Exercise 17.68, every element of C/N is either positive, negative, or zero. For $\alpha, \beta \in C/N$, we say "$\alpha < \beta$" if $\alpha - \beta$ is negative. Of course, from this definition, we define the relations "$>$," "$\leq$," and "$\geq$," along with the absolute value function "$|\cdot|$," in the obvious way. This gives us a total order on C/N, extending that on $\mathbb{Q}$.

One can derive all of the usual properties of inequalities from these definitions, for example:

Exercise 17.73 Let $\alpha, \beta, \gamma \in C/N$. Show that

(a) exactly one of $\alpha < \beta$, $\alpha = \beta$, or $\alpha > \beta$ holds,

(b) $\alpha < \beta$ and $\beta < \gamma$ implies $\alpha < \gamma$,

(c) $\alpha < \beta$ implies $\alpha + \gamma < \beta + \gamma$,

(d) $\alpha < \beta$ and $\gamma > 0$ implies $\alpha\gamma < \beta\gamma$,

(e) $\alpha < \beta$ implies $-\alpha > -\beta$,

(f) $|\alpha\beta| = |\alpha||\beta|$,

(g) $|\alpha + \beta| \leq |\alpha| + |\beta|$.

□

Indeed, all of the familiar properties of the reals may be derived from these definitions. However, this is not a course in analysis, and so we will not pursue this matter any further, except to ask the interested reader to derive the following standard results from real analysis from the definition of the reals as C/N:

Exercise 17.74 Show that the rationals are dense in the reals, i.e., between any two distinct real numbers, there lies a rational number. □

Exercise 17.75 Any Cauchy sequence of real numbers converges to a real number. □

We have given one specific construction of the real numbers. There are other constructions (e.g., "Dedekind cuts"). However, all these constructions yield isomorphic fields.

Chapter 18

Polynomial Arithmetic and Applications

In this chapter, we study algorithms for performing arithmetic in $F[\mathtt{X}]$, where F is a field. There are many similarities between arithmetic in $\mathbb{Z}$ and in $F[\mathtt{X}]$, and many of the algorithms we discuss in this chapter will be quite similar to the corresponding algorithms for integers. There are differences, however, and one has to be aware of these.

All of the algorithms in this chapter work over any field F, and as we did in §15 in the context of algorithms for matrices, we analyze the running-times of these algorithms by counting the number of arithmetic operations in F that the algorithm performs. An arithmetic operation is an addition, subtraction, multiplication, division, or comparison.

The actual running-time of an algorithm for a particular field will depend on the time to perform arithmetic operations in the field. For example, if $F = \mathbb{Z}_p$, and we have an algorithm that uses $O(m)$ arithmetic operations in F, then since each arithmetic operation takes time $O(\mathrm{len}(p)^2)$, the total running-time would be $O(m\,\mathrm{len}(p)^2)$.

If the field F is the field of rational numbers, then the situation is somewhat more complicated. In this case, to determine an upper bound on the running-time of an algorithm, we must not only determine an upper bound on the number of arithmetic operations in F, but also an upper bound on the binary lengths of the numerators and denominators of the rational numbers (assuming these numbers are represented as fractions in lowest terms) appearing as intermediate values in the computation. Thus, if our algorithm uses $O(m)$ arithmetic operations, and all numerators and denominators have binary length $O(\ell)$, then the total running-time is $O(m\ell^2)$. We shall not address these issues here, but only remark that t all of the algorithms discussed in this chapter can indeed be shown to run in polynomial time in this setting.

18.1 Basic Arithmetic

In this section, we temporarily take a more general point of view, and consider arithmetic of polynomials over an arbitrary ring R.

We assume that a polynomial $a = a_0 + a_1\mathtt{X} + \cdots + a_n\mathtt{X}^n \in R[\mathtt{X}]$ is represented as a coefficient vector $(a_0, a_1, \ldots, a_n)$. Further, we assume that $a_n \neq 0$ if $a \neq 0$, and that $n = 1$ if $a = 0$. For a polynomial $a \in R[\mathtt{X}]$, we define its **length**, denoted $\mathrm{len}(a)$, to be the length of its coefficient vector. Thus, $\mathrm{len}(a) = \max\{\deg(a)+1, 1\}$. It is sometimes more convenient to state the running times of algorithms in terms of $\mathrm{len}(a)$, rather than $\deg(a)$ (the latter has the inconvenient habit of taking on the value 0, or worse, $-\infty$).

The following theorem is the analog of Theorem 3.16.

Theorem 18.1 *Let a and b be arbitrary polynomials in $R[\mathtt{X}]$.*

(i) We can compute $a \pm b$ with $O(\mathrm{len}(a) + \mathrm{len}(b))$ operations in R.

(ii) We can compute $a \cdot b$ with $O(\mathrm{len}(a)\,\mathrm{len}(b))$ operations in R.

(iii) If $b \neq 0$ and $\mathrm{lc}(b)$ is a unit in R, we can compute q and r such that $a = bq + r$ and $\deg(r) < \deg(b)$ with $O(\mathrm{len}(b)\,\mathrm{len}(q))$ operations in R.

Proof. All of these operations can be performed using the standard "paper-and-pencil" method method. Indeed, the basic arithmetic algorithms for polynomials are significantly simpler than the corresponding algorithms for integers, since in the case of polynomials, we do not have to worry about "carries." We leave the verification of the operations counts to the reader. □

Analogous to algorithms for modular integer arithmetic, we can also do arithmetic in the residue class ring $R[\mathtt{X}]/(f)$, where $f \in R[\mathtt{X}]$ is a monic polynomial of degree $\ell > 0$ whose leading coefficient $\mathrm{lc}(f)$ is a unit.

For computational purposes, elements of $R[\mathtt{X}]/(f)$ are represented as polynomials of degree less than ℓ, which in turn are represented as coefficient vectors of length at most ℓ. With this representation, addition and subtraction in $R[\mathtt{X}]/(f)$ can be performed using $O(\ell)$ operations in R, while multiplication takes $O(\ell^2)$ operations in R.

As in §3.4, we make a clear distinction between elements of $R[\mathtt{X}]$ and elements of $R[\mathtt{X}]/(f)$. To convert an element of $a \in R[\mathtt{X}]$ to an element $\alpha \in R[\mathtt{X}]/(f)$, we write $\alpha \leftarrow [a \bmod f]$; to convert an element $\alpha \in R[\mathtt{X}]/(f)$ to an element $a \in R[\mathtt{X}]$, we write $a \leftarrow \mathrm{rep}(\alpha)$, where the resulting value a is the unique polynomial of degree less than ℓ such that $\alpha = [a \bmod f]$.

The repeated-squaring algorithm for computing powers works equally well in this setting: given $\alpha \in R[\mathtt{X}]/(f)$ and a non-negative exponent e, we can compute α^e using $O(\mathrm{len}(e))$ multiplications in $R[\mathtt{X}]/(f)$, and so a total of $O(\mathrm{len}(e)\,\ell^2)$ operations in R.

The following exercises deal with arithmetic with polynomials $R[\mathtt{X}]$ over a ring R.

Exercise 18.2 State and re-work the polynomial analog of Exercise 3.17. □

Exercise 18.3 State and re-work the polynomial analog of Exercise 3.18. Assume $n_1, \ldots, n_k$ are monic polynomials. □

Exercise 18.4 State and re-work the polynomial analog of Exercise 3.19. □

In the following exercises, assume that we have an algorithm that multiplies two polynomials of length at most ℓ using at most $M(\ell)$ operations in R, where M is a well-behaved complexity function.

Exercise 18.5 State and re-work the polynomial analog of Exercise 3.20. □

Exercise 18.6 This problem is the polynomial analog of Exercise 3.21. Let us first define the notion of a "floating point" reversed Laurent series $\hat{z}$, which is a pair (a, e), where $a \in R[\mathtt{X}]$ and $e \in \mathbb{Z}$ — the value of $\hat{z}$ is $a\mathtt{X}^e \in R(\!(\mathtt{X}^{-1})\!)$, and we call $\mathrm{len}(a)$ the **precision** of $\hat{z}$. We say that $\hat{z}$ is a **length** k **approximation** of $z \in R(\!(\mathtt{X}^{-1})\!)$ if $\hat{z}$ has precision k and $\hat{z} = z(1+\epsilon)$ for $\epsilon \in R(\!(\mathtt{X}^{-1})\!)$ with $\deg(\epsilon) \leq -k$ — this is the same as saying that the high order k coefficients of $\hat{z}$ and z are equal. Show how to compute — given monic $b \in R[\mathtt{X}]$ and positive integer k — a length k approximation to $1/b \in R(\!(\mathtt{X}^{-1})\!)$ using $O(M(k))$ operations in R. Hint: using Newton iteration, show how to go from a length t approximation to a length $2t$ approximation, making use of just the high order t coefficients of b, using $O(M(t))$ operations in R. □

Exercise 18.7 State and re-work the polynomial analog of Exercise 3.22. Assume that b is a monic polynomial. □

Exercise 18.8 State and re-work the polynomial analog of Exercise 3.23. Conclude that a polynomial of length ℓ can be evaluated at ℓ points using $O(M(\ell)\,\mathrm{len}(\ell))$ operations in R. □

Exercise 18.9 State and re-work the polynomial analog of Exercise 3.24, assuming that R is a field of odd characteristic. □

Exercise 18.10 State and re-work the polynomial analog of Exercise 3.25. Assume that $2 \in R^*$. □

Example 18.11 Let F be a field, $f \in F[\mathtt{X}]$ a monic irreducible polynomial of degree ℓ, and let $E := F[\mathtt{X}]/(f)$. E is a finite extension of degree ℓ over F. Suppose we are given an element $\alpha \in E$, and want to efficiently compute the minimal polynomial of α over F, i.e., the monic polynomial $\phi \in F[\mathtt{X}]$ of least degree such that $\phi(\alpha) = 0$, which we know has degree at most ℓ.

We can solve this problem using Gaussian elimination, as follows. Consider the F-linear map ρ from $F[\mathtt{X}]_{\leq \ell}$ to E that sends a polynomial f of degree at most ℓ to $f(\alpha)$. Let us fix ordered bases for $F[\mathtt{X}]_{\leq \ell}$ and E: for $F[\mathtt{X}]_{\leq \ell}$, let us take $\mathtt{X}^\ell, \mathtt{X}^{\ell-1}, \ldots, 1$, and for E, let us take $1, \eta, \ldots, \eta^{\ell-1}$, where $\eta := [\mathtt{X} \bmod f]$. The matrix A representing the map ρ (via multiplication on the right by A), is the $(\ell+1) \times \ell$ matrix A whose ith row, for $1 \leq i \leq \ell+1$, is the coordinate vector of $\alpha^{\ell+1-i}$.

We apply Gaussian elimination to A to find a set of row vectors $v_1, \ldots, v_s$ which are coordinate vectors for a basis for the kernel of ρ. Now, the coordinate vector of the minimal polynomial of α is a linear combination of $v_1, \ldots, v_s$. To find it, we form the $s \times (\ell + 1)$ matrix B whose rows consist of $v_1, \ldots, v_s$, and apply Gaussian elimination to B, obtaining an $s \times (\ell + 1)$ matrix B' in reduced row echelon form whose row space is the same as that of B. Let g be the polynomial whose coordinate vector is the last row of B'. Because of the choice of ordered basis for $F[\mathtt{X}]_{\leq \ell}$, and because B' is in reduced row echelon form, it is clear that no non-zero polynomial in $\ker(\rho)$ has degree less than that of g. Moreover, as g is already monic (again, by the fact that B' is in reduced row echelon form), it follows that g is in fact the minimal polynomial of α over F.

The total amount of work performed by this algorithm is $O(\ell^3)$ operations in F to build the matrix A (this just amounts to computing ℓ successive powers of α), and $O(\ell^3)$ operations in F to perform both Gaussian elimination steps.

Note that this algorithm works just as well when f is not irreducible. □

18.2 Euclid's Algorithm

Now we return to doing arithmetic with polynomials whose coefficients lie in a field F, and we consider the computation of greatest common divisors in $F[\mathtt{X}]$. The following is the analog of Theorem 4.1.

Theorem 18.12 *Let $a, b \in F[\mathtt{X}]$, with $\deg(a) \geq \deg(b)$ and $a \neq 0$. Define the polynomials $r_0, r_1, \ldots, r_{\ell+1}$, and $q_1, \ldots, q_\ell$, where $\ell \geq 0$, as follows:*

$$\begin{aligned}
r_0 &= a, \\
r_1 &= b, \\
r_0 &= r_1 q_1 + r_2 \quad (-\infty < \deg(r_2) < \deg(r_1)), \\
&\vdots \\
r_{i-1} &= r_i q_i + r_{i+1} \quad (-\infty < \deg(r_{i+1}) < \deg(r_i)), \\
&\vdots \\
r_{\ell-2} &= r_{\ell-1} q_{\ell-1} + r_\ell \quad (-\infty < \deg(r_\ell) < \deg(r_{\ell-1})), \\
r_{\ell-1} &= r_\ell q_\ell \quad (r_{\ell+1} = 0).
\end{aligned}$$

Then $r_\ell / \operatorname{lc}(r_\ell) = \gcd(a, b)$. Moreover, if $b \neq 0$, then $\ell \leq \deg(b) + 1$, and if $b = 0$, then $\ell = 0$.

Proof. Arguing as in the proof of Theorem 4.1, one sees that $\gcd(r_0, r_1) = \gcd(r_\ell, r_{\ell+1}) = r_\ell / \operatorname{lc}(r_\ell)$. That proves the first statement. Also, one easily sees that for $0 \leq i \leq \ell - 1$, $\deg(r_{\ell-i}) \geq i$, from which the second statement follows. □

This gives us the following Euclidean algorithm for polynomials, which takes as input polynomials a, b with $\deg(a) \geq \deg(b)$ and $a \neq 0$:

while $b \neq 0$ do
    $(a, b) \leftarrow (b, a \text{ rem } b)$
output $a/\operatorname{lc}(a)$

By Theorem 18.12, this algorithm outputs the greatest common divisor of a and b.

Theorem 18.13 *Euclid's algorithm for polynomials uses $O(\operatorname{len}(a)\operatorname{len}(b))$ operations in F.*

Proof. The proof is almost identical to that of Theorem 4.3. Details are left to the reader. □

Just as for integers, if $d = \gcd(a, b)$, then $(d) = (a, b)$, and so there exist polynomials s and t such that $as + bt = d$. The procedure to calculate s and t is precisely the same as in the case for integers; however, in the polynomial case, we can be much more precise about the relative sizes of the objects involved in the calculation.

Theorem 18.14 *Let a, b, $r_0, r_1, \ldots, r_{\ell+1}$, and $q_1, \ldots, q_\ell$ be as in Theorem 18.12. Define polynomials $s_0, s_1, \ldots, s_{\ell+1}$ and $t_0, t_1, \ldots, t_{\ell+1}$ as follows:*

$$s_0 := 1, \quad t_0 := 0,$$

$$s_1 := 0, \quad t_1 := 1,$$

and for $1 \leq i \leq \ell$,

$$s_{i+1} := s_{i-1} - s_i q_i, \quad t_{i+1} := t_{i-1} - t_i q_i.$$

Then

(i) for $0 \leq i \leq \ell + 1$, we have $s_i a + t_i b = r_i$;

(ii) for $0 \leq i \leq \ell$, $s_i t_{i+1} - t_i s_{i+1} = (-1)^i$;

(iii) for $0 \leq i \leq \ell + 1$, $\gcd(s_i, t_i) = 1$;

(iv) for $2 \leq i \leq \ell + 1$, we have

$$\deg(s_i) = \deg(b) - \deg(r_{i-1}) \quad \textit{and} \quad \deg(t_i) = \deg(a) - \deg(r_{i-1});$$

moreover, for $i = 1$, we have

$$\deg(s_i) \leq \deg(b) - \deg(r_{i-1}) \quad \textit{and} \quad \deg(t_i) = \deg(a) - \deg(r_{i-1});$$

(v) for $1 \leq i \leq \ell$, we have $\deg(s_{i+1}) > \deg(s_i)$; for $0 \leq i \leq \ell$, we have $\deg(t_{i+1}) \geq \deg(t_i)$, and these inequalities are strict*, except for the case $i = 1$ when $\deg(a) = \deg(b)$;*

(vi) for $1 \leq i \leq \ell + 1$, we have $\deg(s_i) \leq \deg(b)$, and for $0 \leq i \leq \ell + 1$, we have $\deg(t_i) \leq \deg(a)$.

Proof. (i), (ii), and (iii) are proved just as in the corresponding parts of Theorem 4.5.

For (iv), first observe that $\deg(q_1) \geq 0$, and $\deg(q_i) \geq 1$ for $2 \leq i \leq \ell$.

We now prove the first statement of (iv) by induction on i. From the definitions, we see that $s_2 = 1$, and $\deg(b) - \deg(r_1) = 0$. Also, $t_2 = -q_1$, and $\deg(a) - \deg(r_1) = \deg(q_1)$. That proves (iv) for $i = 2$. Now suppose $i > 2$. Consider first the statement involving s_i. By definition, $s_i = s_{i-2} - s_{i-1}q_{i-1}$. We claim that $\deg(s_{i-1}q_{i-1}) > \deg(s_{i-2})$; this follows from the fact that $\deg(q_{i-1}) > 0$ and the fact that $\deg(s_{i-1}) \geq \deg(s_{i-2})$ (the latter fact follows from the induction hypothesis for $i > 3$ and by inspection for $i = 3$). Thus, again applying the induction hypothesis, we see that

$$\deg(s_i) = \deg(s_{i-1}) + \deg(q_{i-1}) = \deg(b) - \deg(r_{i-2}) + \deg(q_{i-1}) = \deg(b) - \deg(r_{i-1}).$$

The induction step for t_i is analogous, and is left to the reader.

The second statement of part (iv) (i.e., the statement for $i = 1$) follows trivially by inspection.

Parts (v) and (vi) follow easily from part (iv); the details are left to the reader. □

We can easily turn the scheme described in Theorem 18.14 into a simple algorithm, taking as input polynomials a, b, such that $\deg(a) \geq \deg(b)$ and $a \neq 0$:

$s \leftarrow 1,\ t \leftarrow 0$
$s' \leftarrow 0,\ t' \leftarrow 1$
while $b \neq 0$ do
    Compute q, r such that $a = bq + r$, with $\deg(r) < \deg(b)$
    $(s, t, s', t') \leftarrow (s', t', s - s'q, t - t'q)$
    $(a, b) \leftarrow (b, r)$
output $a/\operatorname{lc}(a), s/\operatorname{lc}(a), t/\operatorname{lc}(a)$

Theorem 18.15 *The extended Euclidean algorithm for polynomials uses $O(\operatorname{len}(a)\operatorname{len}(b))$ operations in F.*

Proof. Left as an exercise for the reader. □

18.3 Computing Modular Inverses and Chinese Remaindering

In this and the remaining sections of this chapter, we explore various applications of Euclid's algorithm for polynomials. Many of these applications are analogous to their integer counterparts, although there are some differences to watch for.

We begin with the obvious application of the extended Euclidean algorithm for polynomials to the problem of computing multiplicative inverses in $F[\mathtt{X}]/(f)$, where $f \in F[\mathtt{X}]$ with $\deg(f) > 0$.

Given $a \in F[\mathtt{X}]$ with $\deg(a) < \deg(f)$, we can determine if $[a \bmod f]$ has a multiplicative inverse in $F[\mathtt{X}]/(f)$, and if so, determine this inverse, using $O(\mathrm{len}(f)^2)$ operations in F, as follows. We run the extended Euclidean algorithm on input (f, a) to determine polynomials d, s, and t, such that $d = \gcd(f, a)$ and $fs + at = d$. If $d \neq 1$, then $[a \bmod f]$ is not invertible; otherwise, $[a \bmod f]$ is invertible, and $[t \bmod f]$ is its inverse. Moreover, by parts (v) and (vi) of Theorem 18.14, we have $\deg(t) < \deg(f)$ (verify), and so the polynomial t may be used directly to represent the multiplicative inverse of $[a \bmod f]$; i.e., there is no need to reduce t modulo f.

If the polynomial f is irreducible, then $F[\mathtt{X}]/(f)$ is a field, and the extended Euclidean algorithm, together with the basic algorithms for addition, subtraction, and multiplication modulo f, gives us efficient algorithms for performing addition, subtraction, multiplication and division in the extension field $F[\mathtt{X}]/(f)$, assuming of course, that we have efficient algorithms for arithmetic in F.

We also observe that Theorem 17.12 (the Chinese Remainder Theorem for polynomials) can be made computationally effective as well.

Theorem 18.16 *Given polynomials $n_1, \ldots, n_k$ and $a_1, \ldots, a_k$ over a field F, with $\deg(n_i) > 0$, $\gcd(n_i, n_j) = 1$ for $i \neq j$, and $\deg(a_i) < \deg(n_i)$, we can compute $z \in F[\mathtt{X}]$ such that $\deg(z) < \deg(n)$ and $z \equiv a_i \pmod{n_i}$ using $O(\mathrm{len}(n)^2)$ operations in F, where $n = \prod_i n_i$.*

Proof. Exercise (just use the formulas in the proof of Theorem 2.7, which are repeated below the statement of Theorem 17.12). □

18.3.1 Chinese remaindering and polynomial interpolation

We remind the reader of the discussion following Theorem 17.12, where the point was made that when $n_i = (\mathtt{X} - b_i)$ for $1 \leq i \leq k$, then the Chinese Remainder Theorem for polynomials reduces to LaGrange interpolation. Thus, Theorem 18.16 says that given distinct elements $b_1, \ldots, b_k \in F$, along with elements $a_1, \ldots, a_k \in F$, we can compute the unique polynomial $z \in F[\mathtt{X}]$ of degree less than k such that

$$z(b_i) = a_i \quad (i = 1, \ldots, k),$$

using $O(k^2)$ operations in F.

It is perhaps worth noting that we could also solve the polynomial interpolation problem using Gaussian elimination, by inverting the corresponding Vandermonde matrix. However, this algorithm would use $O(k^3)$ operations in F. This is specific instance of a more general phenomenon: there are many computational problems involving polynomials over fields can be solved using Gaussian elimination, but which can be solved more efficiently using more specialized algorithmic techniques.

Exercise 18.17 State and re-work the polynomial analog of Exercises 4.12 and 4.13. In the special case of polynomial interpolation, this algorithm is called **Newton interpolation**. □

18.3.2 Mutual independence and secret sharing

As we also saw in the discussion following Theorem 17.12, for $\ell \leq k$ and fixed and distinct $b_1, \ldots, b_\ell \in F$, the "multi-point evaluation" map σ from $F[\mathtt{X}]_{<k}$ to $F^{\times \ell}$ that sends $z \in F[\mathtt{X}]_{<k}$ to $(z(b_1), \ldots, z(b_\ell)) \in F^{\times \ell}$ is a surjective F-linear map. If F is a finite field, then this has the following probabilistic interpretation: if the coefficient vector $(z_0, \ldots, z_{k-1})$ of z is a random variable, uniformly distributed over $F^{\times k}$, i.e., the z_i's are independently and uniformly distributed over F, then the random variables $z(b_1), \ldots, z(b_\ell)$ are independently and uniformly distributed over F. This is because: (1) σ is surjective, and (2) every element of $F^{\times \ell}$ has the same number of pre-images under σ, namely $|\ker(\sigma)| = |F|^d$, where $d = \dim_F(\ker(\sigma))$; from this, it follows that when $z \in F[\mathtt{X}]_{<k}$ is chosen at random, all possible values are equally likely.

Put another way, the collection $\{z(b) : b \in F\}$ of random variables is ℓ-wise independent, where each individual $z(b)$ uniformly distributed over F. Clearly, given z and b, we can efficiently compute the value of $z(b)$, so this construction gives us a nice way to build effectively constructible, ℓ-wise independent collections of random variables for any ℓ, thus generalizing the constructions in Examples 6.23 and 6.25 of pairwise and 3-wise independent collections.

As a particular application of this idea, we describe a simple **secret sharing scheme**. Suppose Alice wants to share a secret among some number m of parties, call them $P_1, \ldots, P_m$, in such a way that if less than k parties share their individual secret shares with one another, then Alice's secret is still well hidden, while any subset of k parties can reconstruct Alice's secret.

She can do this as follows. Suppose her secret s is (or can be encoded as) an element of a finite field F, and that $b_0, b_1, \ldots, b_m$ are some fixed, distinct elements of F, where $b_0 = 0$. This presumes, of course, that $|F| \geq m + 1$. To share her secret s, Alice chooses $z_1, \ldots, z_{k-1} \in F$ at random, and sets $z_0 := s$. Let $z \in F[\mathtt{X}]$ be the polynomial whose coefficient vector is $(z_0, \ldots, z_{k-1})$, i.e.,

$$z = \sum_{i=0}^{k-1} z_i \mathtt{X}^i.$$

For $1 \leq i \leq m$, Alice gives party P_i its share

$$a_i := z(b_i).$$

For the purposes of analysis, it is convenient to define

$$a_0 := z(b_0) = z(0) = z_0 = s.$$

Clearly, if any k parties pool their shares, they can reconstruct Alice's secret by interpolating a polynomial of degree less than k at k points — the constant term of this polynomial is equal to Alice's secret s.

It remains to show that Alice's secret remains well hidden provided less than k parties pool their shares. To do this, first assume that Alice's secret s is uniformly distributed over F, independently of $z_1, \ldots, z_{k-1}$ (we will relax this assumption below). With this assumption, $z_0, z_1, \ldots, z_{k-1}$ are independently and uniformly distributed over F. Now consider any subset of $k-1$ parties; to simplify notation, assume the parties are $P_1, \ldots, P_{k-1}$. Then the random variables $a_0, a_1, \ldots, a_{k-1}$ are mutually independent. The variables $a_1, \ldots, a_{k-1}$ are of course the shares of $P_1, \ldots, P_{k-1}$, while a_0 is equal to Alice's secret (the fact that a_0 has two interpretations, one as the value of z at a point, and one as a coefficient of z, plays a crucial role in the analysis). Because of mutual independence, the distribution of a_0, conditioned on fixed values of the shares $a_1, \ldots, a_{k-1}$, is still uniform over F, and so even by pooling their shares, these $k-1$ parties would have no better chance of guessing Alice's secret than they would have without pooling their shares.

Continuing the analysis of the previous paragraph, consider the conditional probability distribution in which we condition on the event that $a_0 = s$ for some specific, fixed value of $s \in F$. Because the $z_0, z_1, \ldots, z_{k-1}$ were initially independently and uniformly distributed over F, and because $z_0 = a_0$, in this conditional probability distribution, we have $z_0 = s$ and $z_1, \ldots, z_{k-1}$ are independently and uniformly distributed over F. So this conditional probability distribution perfectly models the secret sharing algorithm performed by Alice for a specific secret s, without presuming that s is drawn from any particular distribution. Moreover, because the $a_0, a_1, \ldots, a_{k-1}$ were initially independently and uniformly distributed over F, in this conditional probability distribution, $a_1, \ldots, a_{k-1}$ are independently and uniformly distributed over F.

The argument in the previous two paragraphs shows that

> *for any fixed secret s, the shares $a_1, \ldots, a_m$ are $(k-1)$-wise independent, with each individual share a_i uniformly distributed over F.*

This property ensures that Alice's secret is *perfectly* hidden, provided that less than k parties pool their shares: for any secret s, these parties just see a bunch of random values in F, with no particular bias that would give any hint whatsoever as to the actual value of s.

Secret sharing has a number of cryptographic applications, but one simple motivation is the following. Alice may have some data that she wants to "back up" on some file servers, who play the role of the parties $P_1, \ldots, P_m$. To do this, Alice gives each server a share of her secret data (if she has a lot of data, she can break it up into many small blocks, and process each block separately). If at a later time, Alice wants to restore her data, she contacts any k servers who will give Alice their shares, from which Alice can reconstruct the original data. In using a secret sharing scheme in this way, Alice trusts that the servers are reliable to the extent that they do not modify the value of their share (as otherwise,

this would cause Alice to reconstruct the wrong data). We shall discuss later in this chapter how one can relax this trust assumption. But even with this trust assumption, Alice does gain something above and beyond the simpler solution of just backing up her data on a single server, namely:

- even if some of the servers crash, or are otherwise unreachable, she can still recover her data, as long as at least k are available at the time she wants to do the recovery;
- even if the data on some (but strictly less than k) of the servers is "leaked" to some attacker, the attacker gains no information about Alice's data.

Exercise 18.18 Consider the data-backup scenario described above. Suppose that Alice wants to back up a large file, which she does by breaking it up into a long sequence of h of "F-sized" blocks. Moreover, Alice does not want to trust that the servers do not maliciously (or accidentally) modify their shares. Show that if Alice has a small amount of secure storage, namely, space for $O(m)$ elements of F, then she can effectively protect herself from malicious servers, so that if any particular server tries to give her a modified share, Alice will fail to detect this with probability at most $(h-1)/|F|$. If $|F|$ is very large (say, $|F| = 2^{128}$), and h is any reasonable value (say, $h \leq 2^{40}$), this failure probability will be acceptably small for all practical purposes. □

18.4 Rational Function Reconstruction and Applications

We next state and prove the polynomial analog of Theorem 4.15. As we are now "reconstituting" a rational function, rather than a rational number, we call this procedure **rational function reconstruction**. Because of the relative simplicity of polynomials compared to integers (in particular, the lack of "carries"), the rational reconstruction theorem for polynomials is a bit "sharper" than the rational reconstruction theorem for integers.

Theorem 18.19 *Let $r^* \geq -1$ and $t^* \geq 0$ be integers, and let $n, y \in F[\mathtt{X}]$ be polynomials such that $r^* + t^* < \deg(n)$ and $\deg(y) < \deg(n)$. Suppose we run the Extended Euclidean Algorithm with inputs $a := n$ and $b := y$. Then, adopting the notation of Theorem 18.14, the following hold:*

1. *There exists a unique index i, with $1 \leq i \leq \ell + 1$, such that $\deg(r_i) \leq r^* < \deg(r_{i-1})$, and for this i, $t_i \neq 0$; let $r' := r_i$, $s' := s_i$, and $t' := t_i$.*
2. *Furthermore, for any polynomials $r, s, t \in F[\mathtt{X}]$ such that*

$$r = sn + ty, \quad \deg(r) \leq r^*, \quad 0 \leq \deg(t) \leq t^*, \tag{18.1}$$

we have

$$r = r'\alpha, \ s = s'\alpha, \ t = t'\alpha,$$

for some non-zero polynomial $\alpha \in F[\mathtt{X}]$.

Proof. By hypothesis, $-1 \leq r^* < \deg(n) = \deg(r_0)$. Moreover, since

$$\deg(r_0), \ldots, \deg(r_\ell), \deg(r_{\ell+1}) = -\infty$$

is a decreasing sequence, and $t_i \neq 0$ for $1 \leq i \leq \ell + 1$, the first statement of the theorem is clear.

Now let i be defined as in the first statement of the theorem. Also, let r, s, t be as in (18.1).

From part (iv) of Theorem 18.14, we have

$$\deg(t_i) \leq \deg(n) - \deg(r_{i-1}) < \deg(n) - r^*.$$

From the equalities $r_i = s_i n + t_i y$ and $r = sn + ty$, we have the two congruences:

$$\begin{aligned} r &\equiv ty \pmod{n}, \\ r_i &\equiv t_i y \pmod{n}. \end{aligned}$$

Subtracting t_i times the first from t times the second, we obtain

$$rt_i \equiv r_i t \pmod{n}.$$

This says that n divides $rt_i - r_i t$; however, using the bounds $\deg(r) \leq r^*$ and $\deg(t_i) < \deg(n) - r^*$, we see that $\deg(rt_i) < \deg(n)$, and using the bounds $\deg(r_i) \leq r^*$, $\deg(t) \leq t^*$, and $r^* + t^* < \deg(n)$, we see that $\deg(r_i t) < \deg(n)$; it immediately follows that

$$\deg(rt_i - r_i t) < \deg(n).$$

Since n divides $rt_i - r_i t$ and $\deg(rt_i - r_i t) < \deg(n)$, the only possibility is that

$$rt_i - r_i t = 0.$$

The rest of the proof runs *exactly* the same as the corresponding part of the proof of Theorem 4.15, as the reader may easily verify. □

Note that when $r^* = -1$, the only possibility for r is the zero polynomial.

18.4.1 Application: Polynomial Interpolation with Errors

We now discuss the polynomial analog of the application in §4.5.1.

If we "encode" a polynomial $z \in F[\mathtt{X}]$, with $\deg(z) < k$, as the sequence $(a_1, \ldots, a_k) \in F^{\times k}$, where $a_i = z(b_i)$, then we can efficiently recover z from this encoding, using an algorithm for polynomial interpolation. Here, of course, the b_i's are distinct elements of F, and F is a finite field (which must have at least k elements, of course).

Now suppose that Alice encodes z as $(a_1, \ldots, a_k)$, and sends this encoding to Bob, but that some, say at most ℓ, of the a_i's may be corrupted during transmission. Let $(\tilde{a}_1, \ldots, \tilde{a}_k)$ denote the vector actually received by Bob.

Here is how we can use Theorem 18.19 to recover the original value of z from $(\tilde{a}_1, \ldots, \tilde{a}_k)$, assuming:

- the original polynomial z has degree at most k',
- at most ℓ errors occur in transmission, and
- $k > 2\ell + k'$.

Let us set $n_i := (\mathtt{X} - b_i)$ for $1 \leq i \leq k$, and $n := n_1 \cdots n_k$. Now, suppose Bob obtains the corrupted encoding $(\tilde{a}_1, \ldots, \tilde{a}_k)$. Here is what Bob does to recover z:

1. Interpolate, obtaining a polynomial y, with $\deg(y) < k$ and $y(b_i) = \tilde{a}_i$ for $1 \leq i \leq k$.
2. Run the Extended Euclidean Algorithm on $a := n$ and $b := y$, and let r', t' be the values obtained from Theorem 18.19 applied with $r^* := k' + \ell$ and $t^* := \ell$.
3. If $t' \mid r'$, output r'/t'; otherwise, output "error."

We claim that the above procedure outputs z, under the assumptions listed above. To see this, let t be the product of the n_i's for those values of i where an error occurred. Now, assuming at most ℓ errors occurred, we have $\deg(t) \leq \ell$. Also, let $r := tz$, and note that $\deg(r) \leq k' + \ell$. We claim that

$$r \equiv ty \pmod{n}. \tag{18.2}$$

To show that (18.2) holds, it suffices to show that

$$tz \equiv ty \pmod{n_i} \tag{18.3}$$

for all $1 \leq i \leq k$. To show this, consider first an index i at which no error occurred, so that $a_i = \tilde{a}_i$. Then $tz \equiv ta_i \pmod{n_i}$ and $ty \equiv t\tilde{a}_i \equiv ta_i \pmod{n_i}$, and so (18.3) holds for this i. Next, consider an index i for which an error occurred. Then by construction, $tz \equiv 0 \pmod{n_i}$ and $ty \equiv 0 \pmod{n_i}$, and so (18.3) holds for this i. Thus, (18.2) holds, from which it follows that the values r', t' obtained from Theorem 18.19 satisfy

$$\frac{r'}{t'} = \frac{r}{t} = \frac{tz}{t} = z.$$

One easily checks that both the procedures to encode and decode a value z run in time $O(k^2)$. The above scheme is an example of an *error correcting code* called a *Reed-Solomon code*. Note that we are completely free to choose the finite field F however we want, just so long as it is big enough. An attractive choice in some settings is to choose $F = \mathbb{Z}_2[\mathtt{Y}]/(f)$, where $f \in \mathbb{Z}_2[\mathtt{Y}]$ is an irreducible polynomial; with this choice, elements of F may be encoded as bit strings of length $\deg(f) - 1$.

One can combine the above error correction technique with the idea of secret sharing (see §18.3.2) to obtain a secret sharing scheme that is robust, even in the presence of erroneous (as opposed to just missing) shares. More precisely, Alice can share a secret $s \in F$ among parties $P_1, \ldots, P_m$, in such a way that (1) if at most k' parties pool their shares, Alice's secret remains well hidden, and (2) from any k shares, we can correctly reconstruct Alice's

secret, provided at most ℓ of the shares are incorrect, and $k > 2\ell + k'$. To do this, Alice chooses $z_1, \ldots, z_{k'} \in F$ at random, sets $z_0 := s$, and $z := \sum_{i=0}^{k'} z_i \mathtt{X}^i \in F[\mathtt{X}]$, and computes the ith share as $a_i := z(b_i)$, for $1 \leq i \leq m$. Here, we assume that the b_i's are distinct, non-zero elements of F. Now, just as in §18.3.2, as long as at most k' parties pool their shares, Alice's secret remains well hidden; however, as long as $k > k' + 2\ell$, we can correctly and efficiently reconstruct Alice's secret given any k values $\tilde{a}_i$, as long as at most ℓ of the $\tilde{a}_i$'s differ from the corresponding value of a_i.

18.4.2 Application: recovering rational functions from their reversed formal Laurent series

We now discuss the polynomial analog of the application in §4.5.2. This is an entirely straightforward translation of the results in §4.5.2, but we shall see below that this problem has its own interesting application.

Suppose Alice knows a rational function $z = s/t \in F(\mathtt{X})$, where s and t are polynomials with $\deg(s) < \deg(t)$, and tells Bob some of the high order coefficients of the reversed formal Laurent series (see §17.7) representing z in $F(\!(\mathtt{X}^{-1})\!)$. We shall show that if $\deg(t) \leq M$ and Bob is given the bound M on $\deg(t)$, along with the high order $2M$ coefficients of z, then Bob can determine z, expressed as a rational function in lowest terms.

So suppose that $z = s/t = \sum_{i=1}^{\infty} z_i \mathtt{X}^{-i}$, and that Alice tells Bob the coefficients $z_1, \ldots, z_{2M}$. Equivalently, Alice gives Bob the polynomial

$$y := \lfloor zn \rfloor,$$

where $n := \mathtt{X}^{2M}$. Here is Bob's algorithm for recovering z:

1. Run the Extended Euclidean Algorithm on inputs $a := n$ and $b := y$, and let let s', t' be as in Theorem 18.19, using $r^* := M - 1$ and $t^* := M$.

2. Output s', t'.

We claim that $z = -s'/t'$.

To prove this, let $z = s/t$ as above, and note that by definition

$$\frac{s}{t} = \frac{y}{n} + w, \tag{18.4}$$

where $w \in F(\!(\mathtt{X}^{-1})\!)$ with $\deg(w) < -2M$. Clearing denominators, we have

$$sn = ty + wnt.$$

Thus we see that $r := wnt$ is an element of $F[\mathtt{X}]$ with $\deg(r) \leq M - 1$, and so we have

$$r = sn - ty, \quad \deg(r) \leq r^*, \quad 0 \leq \deg(t) \leq t^*, \quad \text{and} \quad r^* + t^* < \deg(n).$$

It follows that the polynomials s', t' from Theorem 18.19 satisfy $s = s'\alpha$ and $-t = t'\alpha$ for some non-zero polynomial α. Thus, $s'/t' = -s/t$, which proves the claim.

We may further observe that since the extended Euclidean algorithm guarantees that $\gcd(s', t') = 1$, not only do we obtain z, but we obtain z expressed as a fraction in lowest terms.

It is clear that this algorithm takes $O(M^2)$ operations in F.

18.4.3 Linearly generated sequences

A very useful application of the problem discussed in §18.4.2 is that of determining the minimal polynomial of a linearly generated sequence.

Let us set the stage for this problem. Let V be an F-vector space and consider an infinite sequence

$$S = (z_1, z_2, \ldots),$$

where $z_i \in V$ for $i = 1, 2 \ldots$. We say that S is **linearly generated (over F)** if there exist scalars $c_1, \ldots, c_m \in F$ such that the following recurrence relation holds:

$$z_{m+1+i} = \sum_{j=1}^{m} c_j z_{j+i} \quad (\text{for } i = 0, 1, 2, \ldots).$$

In this case, all of the elements of the sequence S are determined by the initial segment $z_1, \ldots, z_m$, together with the coefficients $c_1, \ldots, c_m$ defining the recurrence relation.

The general problem we consider is this: how to determine the coefficients defining such a recurrence relation, given a sufficiently long initial segment of S. To study this problem, it turns out to be very useful to rephrase the problem just a bit. Let $g \in F[\mathtt{X}]$ be a polynomial of degree, say, m, and write $g = \sum_{j=0}^{m} g_j \mathtt{X}^j$. Next, define

$$g \star S := \sum_{j=0}^{m} g_j z_{j+1}.$$

Then it is clear that S is linearly generated if and only if there exists a non-zero polynomial g such that

$$(\mathtt{X}^i g) \star S = 0 \quad (\text{for } i = 0, 1, 2, \ldots). \tag{18.5}$$

Indeed, if there is such a non-zero polynomial g, then we can take

$$c_1 := -(g_0/g_m),\ c_2 := -(g_1/g_m),\ \ldots,\ c_m := -(g_{m-1}/g_m)$$

as coefficients defining the recurrence relation for S. Different polynomials give rise to different coefficients, so it would be nice to characterize all the polynomials that satisfy (18.5).

Let $I(S)$ be the set of all polynomials g (including the zero polynomial) that satisfy (18.5).

Theorem 18.20 *$I(S)$ is an ideal in $F[\mathtt{X}]$.*

Proof. First, note that for any two polynomials f, g, we have $(f+g) \star S = (f \star S) + (g \star S)$ — this is clear from the definitions. It is also clear that for any $c \in F$ and $f \in F[\mathtt{X}]$, we have $(cf) \star S = c(f \star S)$. From these two observations, it is immediately clear that $I(S)$ is closed under addition and scalar multiplication. It is also clear from the definition that $I(S)$ is closed under multiplication by $\mathtt{X}$; indeed, if $(\mathtt{X}^i f) \star S = 0$ for all $i \geq 0$, then certainly, $(\mathtt{X}^i(\mathtt{X}f)) \star S = (\mathtt{X}^{i+1} f) \star S = 0$ for all $i \geq 0$. But any subset of $F[\mathtt{X}]$ that is closed under addition, multiplication by elements of F, and multiplication by $\mathtt{X}$ is an ideal in $F[\mathtt{X}]$ (see Exercise 9.52). □

Since all ideals in $F[\mathtt{X}]$ are principal, it follows that $I(S) = (\phi_S)$ for some polynomial $\phi_S \in F[\mathtt{X}]$ — we can make this polynomial unique by choosing the monic associate (if it is non-zero), and we call this polynomial the **minimal polynomial of** S. Note that S is linearly generated if and only if $\phi_S \neq 0$, in which case, all polynomials g satisfying (18.5) are polynomial multiples of ϕ_S.

We can now restate the main objective of this section as follows: given a sufficiently long initial segment of a linearly generated sequence, determine its minimal polynomial. We shall only address a special case of this problem here, namely, the case where the vector space V is just the field F. In this case, we have

$$S = (z_1, z_2, \ldots),$$

where $z_i \in F$ for $i = 1, 2, \ldots$.

Suppose that we do not know ϕ_S, but we know an upper bound $M \geq 0$ on its degree. Then it turns out that the initial segment $z_1, z_2, \ldots z_{2M}$ completely determines ϕ_S, and moreover, we can efficiently compute ϕ_S given the bound M and this initial segment. The following theorem provides the essential ingredient.

Theorem 18.21 *Let $S = (z_1, z_2, \ldots)$ be a sequence of elements of F, and define the reversed formal Laurent series*

$$z := \sum_{i=1}^{\infty} z_i T^{-i},$$

whose coefficients are the elements of the sequence S. Then for any $g \in F[\mathtt{X}]$, we have $g \in I(S)$ if and only if $gz \in F[\mathtt{X}]$. In particular, S is linear generated if and only if z is a rational function, in which case, ϕ_S is the denominator of z when expressed as a fraction in lowest terms.

Proof. Observe that for any polynomial $g \in F[\mathtt{X}]$ and any integer $i \geq 0$, the coefficient of $\mathtt{X}^{-(i+1)}$ in the product gz is equal to $\mathtt{X}^i g \star S$ — just look at the formulas defining these expressions! It follows that g satisfies (18.5) if and only if the coefficients of the negative powers of $\mathtt{X}$ in gz are all zero, i.e., $gz \in F[\mathtt{X}]$. Further, if $g \neq 0$ and $f := gz \in F[\mathtt{X}]$, then $\deg(f) < \deg(g)$ — this follows simply from the fact that $\deg(z) < 0$. All the statements in the theorem follow immediately from these observations. □

Thus, using the algorithm in §18.4.2, we can compute ϕ_S given the bound M on its degree and the first $2M$ elements $z_1, \ldots, z_{2M}$ of S. Just for completeness, we write down this algorithm:

1. Run the Extended Euclidean Algorithm on inputs

$$a := \mathtt{X}^{2M} \quad \text{and} \quad b := z_1\mathtt{X}^{2M-1} + z_2\mathtt{X}^{2M-2} + \cdots + z_{2M},$$

   and let let s', t' be as in Theorem 18.19, using $r^* := M - 1$ and $t^* := M$.

2. Output $\phi := t'/\operatorname{lc}(t')$.

From the above discussion, it is clear that $\phi = \phi_S$, and can be computed using $O(M^2)$ operations in F.

Exercise 18.22 Suppose that $S = (z_1, z_2, \ldots)$ is a linearly generated sequence over F, where each z_i is an element of an F-vector space V. Let $\rho : V \to V'$ be an F-linear map from V to a vector space V', and let $S' = (z'_1, z'_2, \ldots)$, where $z'_i = \rho(z_i)$ for $i = 1, 2, \ldots$. Show that S' is also linearly generated, and moreover, that $\phi_{S'} \mid \phi_S$. □

Exercise 18.23 Suppose that you are given $c_1, \ldots, c_m \in F$ and $z_1, \ldots, z_m \in F$. Suppose that for all $i \geq 0$, we define

$$z_{m+1+i} := \sum_{j=1}^{m} c_j z_{j+i}.$$

Given $k \geq 1$, show how to compute z_k using $O(\operatorname{len}(k)m^2)$ operations in F. □

Example 18.24 Consider again the problem in Example 18.11: F is a field, $f \in F[\mathtt{X}]$ is a monic irreducible polynomial of degree ℓ, and $E := F[\mathtt{X}]/(f) = F[\eta]$, where $\eta := [\mathtt{X} \bmod f]$; we are given an element $\alpha \in E$, and want to compute the minimal polynomial $\phi \in F[\mathtt{X}]$ of α over F.

Using the theory of linearly generated sequences, we can derive an algorithm for this problem that does not rely on Gaussian elimination, and is in some respects more attractive than the algorithm presented in Example 18.11. Consider the sequence $S = (z_1, z_2, \ldots)$, where $z_i = \alpha^{i-1}$, for $i \geq 1$. Evidently, S is a linearly generated sequence over F whose minimal polynomial ϕ_S is precisely ϕ (verify). We now define a "projection map" $\pi : E \to F$, as follows: for $\beta = g(\eta) \in E$, with $\deg(g) < \ell$, we define $\pi(\beta)$ to be the constant term of g. It is easy to see that π is an F-linear map from E onto F. Next, consider the sequence $S' := (z'_1, z'_2, \ldots)$, where $z'_i := \pi(z_i) = \pi(\alpha^{i-1})$, for $i \geq 1$. By Exercise 18.22, S' is also a linearly generated sequence over whose minimal polynomial $\phi_{S'}$ divides ϕ_S. Now, as $\phi_S = \phi$ is irreducible, $\phi_{S'}$ is equal to either 1 or ϕ; moreover, the only sequence whose minimal polynomial is 1 is the all-zero sequence (verify), and since $z'_1 = \pi(1) = 1$, we conclude that $\phi_{S'} = \phi$.

So we have reduced the problem of computing the minimal polynomial of α to that of computing the minimal polynomial of the linearly generated sequence S'. To solve the latter

problem, we can compute the projections $\pi(1), \pi(\alpha), \ldots, \pi(\alpha^{2\ell-1})$ of the first 2ℓ powers of α, and then apply the algorithm above for computing the minimal polynomial of linearly generated sequences of field elements. The cost of computing the projected powers using the obvious method of just computing successive powers of α is $O(\ell)$ operations in E, each of which costs $O(\ell^2)$ operations in F, for a total cost of $O(\ell^3)$ operations in F. The cost of computing the minimal polynomial given these projected powers is just an additional $O(\ell^2)$ operations in F. Thus, the total cost of this algorithm is $O(\ell^3)$ operations in F.

From the point of view of the time complexity, this does not improve on the approach via Gaussian elimination. However, it does improve on the *space* complexity: the Gaussian elimination approach requires us to store an entire matrix, meaning that we need space to store $\Theta(\ell^2)$ elements of F, while the approach based on linearly generated sequences requires space for only $O(\ell)$ elements of F. In practice, such space savings can be important. Furthermore, in Exercise 18.27 below, it is shown how to implement this algorithm so that it uses just $O(\ell^{2.5})$ operations in F, at the expense of space for $O(\ell^{1.5})$ elements of F. Further speed improvements are possible using subquadratic algorithms for polynomial arithmetic, of course.

Note that unlike the algorithm based on Gaussian elimination, this algorithm requires that f is irreducible. □

Exercise 18.25 Let $f \in F[\mathtt{X}]$ be a polynomial over a field F of degree $\ell > 0$, and let $R := F[\mathtt{X}]/(f)$. Show how to compute — given as input the polynomial f defining R, an element $\alpha \in R$, and a polynomial $g \in F[\mathtt{X}]$ of degree $k > 0$ — the value $g(\alpha) \in R$, using just $O(k\ell + k^{1/2}\ell^2)$ operations in F. Hint: first compute a table of powers $1, \alpha, \ldots, \alpha^m$, for $m \approx k^{1/2}$. □

Exercise 18.26 Let $f \in F[\mathtt{X}]$ be a polynomial over a field F of degree $\ell > 0$, and let $R := F[\mathtt{X}]/(f)$. Also, let $\eta := [\mathtt{X} \bmod f] \in R$. A *linear projection* π is an F-linear map from R to F. For computational purposes, we assume that π is represented as a vector $v \in F^{\times \ell}$, so that if $v = (v_0, \ldots, v_{\ell-1})$, and if $\alpha = a_0 + a_1\eta + \cdots + a_{\ell-1}\eta^{\ell-1} \in R$, then $\pi(\alpha) = v_0 a_0 + \cdots v_{\ell-1} a_{\ell-1}$. Thus, given (representations of) π and α, we can compute $\pi(\alpha)$ using $O(\ell)$ operations in F. For $\beta \in R$, let M_β denote the "multiplication by β" map on R that sends $\alpha \in R$ to $\alpha\beta$, which is an F-linear map from R into R.

(a) Show how to compute — given as input the polynomial f defining R, along with a linear projection π and an element $\beta \in R$ — the (representation of) the linear projection $\pi \circ M_\beta$, using $O(\ell^2)$ operations in F.

(b) Show how to compute — given as input the polynomial f defining R, along with a linear projection π, an element $\alpha \in R$, and a parameter $k > 0$ — all of the k values

$$\pi(1), \pi(\alpha), \ldots, \pi(\alpha^{k-1})$$

using just $O(k\ell + k^{1/2}\ell^2)$ operations in F. Hint: same hint as in the previous exercise.

□

Exercise 18.27 Show how to use the result of the previous exercise to improve the running time of the algorithm in Example 18.24 to $O(\ell^{2.5})$ operations in F. □

18.5 Notes

Just as in the case of integer arithmetic, the basic "pencil and paper" quadratic-time algorithms discussed in this chapter for polynomial arithmetic are not the best possible. The fastest known algorithms for multiplication and division of polynomials over a ring R of length ℓ take $O(\ell \operatorname{len}(\ell) \operatorname{len}(\operatorname{len}(\ell)))$ operations in R. The Euclidean and extended Euclidean algorithms for polynomials over a field F can be implemented so as to take $O(\ell \operatorname{len}(\ell)^2 \operatorname{len}(\operatorname{len}(\ell)))$ operations in F, as can the algorithms for Chinese remaindering and rational function reconstruction. See the book by von zur Gathen and Gerhard [73] for details (as well for an analysis of the Euclidean algorithm for polynomials over the field of rational numbers and over function fields).

Depending on the setting and many implementation details, such "asymptotically" fast algorithms for multiplication and division can be significantly faster than the quadratic-time algorithms, even for quite moderately sized inputs of practical interest. However, the fast Euclidean algorithms are only useful for significantly larger inputs.

The interpretation of LaGrange interpolation as "secret sharing" (see §18.3.2), and its application to cryptography, was made by Shamir [65].

Reed-Solomon codes were first propose by Reed and Solomon [59], although the decoder presented here was developed later. Theorem 18.19 was proved by Mills [49]. Berlekamp [13] and Massey [46] discuss an algorithm for finding the minimal polynomial of a linearly generated sequence that is closely related to the one presented here, and which has a similar complexity. This connection between Euclid's algorithm and finding minimal polynomials of linearly generated sequences has been observed by many authors, including Mills [49], Welch and Scholtz [77], and Dornstetter [27]. The algorithm presented in Example 18.24 for computing minimal polynomials is a special case of a technique due to Wiedemann [78] for solving special systems of linear equations over finite fields. Assuming fast polynomial arithmetic, Shoup [69] shows how to implement this algorithm so as to use just $O(\ell^{(\omega+1)/2})$ operations in F, where ω is the exponent for matrix multiplication, and so $(\omega+1)/2 < 1.7$.

Chapter 19

Finite Fields

This chapter develops some of the basic theory of finite fields. The main results concern the existence and uniqueness of finite fields; namely, (1) any finite field has p^w elements, for some prime p and positive integer w, (2) for any such p and w there exists a finite field of cardinality p^w, and (3) any two finite fields of the same cardinality are isomorphic.

19.1 The Characteristic and Cardinality of a Finite Field

Let F be a finite field. Clearly, simply because F is finite, its characteristic must be non-zero, and by the discussion in Example 9.73, its characteristic must be a prime p and we may view $\mathbb{Z}_p$ as a subfield of F. Again because F is finite, its degree $w = [F : \mathbb{Z}_p]$ over $\mathbb{Z}_p$ must be finite (see §17.5). It immediately follows that F has cardinality p^w.

We proved in Theorem 10.2 that any finite subgroup of the multiplicative group of units of a field is cyclic. In particular, for the finite field F, F^* is cyclic. If $\gamma \in F^*$ is a generator for F^*, then in particular, every element of F can be expressed as a polynomial in γ with coefficients in $\mathbb{Z}_p$; that is, $F = \mathbb{Z}_p[\gamma]$. Let $\phi \in \mathbb{Z}_p[\mathtt{X}]$ be the minimal polynomial of γ (see §17.4 and §17.5). So F is isomorphic (as a $\mathbb{Z}_p$-algebra) to $\mathbb{Z}_p[\mathtt{X}]/(\phi)$, where ϕ is an irreducible polynomial over $\mathbb{Z}_p$ of degree w. Conversely, given any irreducible polynomial ϕ over $\mathbb{Z}_p$ of degree w, we can construct the finite field $\mathbb{Z}_p[\mathtt{X}]/(\phi)$ of cardinality p^w. Thus, the question of the existence of a finite fields of a given cardinality p^w reduces to the question of the existence of an irreducible polynomial over $\mathbb{Z}_p$ of degree w.

The observations in the previous paragraph, by the way, give another proof that the cardinality of F must be a power of p, without appealing to the theory of vector spaces and dimension. Indeed, since every element of $\mathbb{Z}_p[\mathtt{X}]/(\phi)$ can be uniquely expressed as $[g \bmod \phi]$, where $g \in \mathbb{Z}_p[\mathtt{X}]$ with $\deg(g) < w$, it follows that $|F| = p^w$.

19.2 Some Useful Divisibility Criteria

Before moving on to the proof that finite fields of every possible cardinality exist, we state two simple but useful theorems:

Theorem 19.1 *Let R be a non-trivial ring, and let k, ℓ be positive integers. The polynomial $\mathtt{X}^k - 1$ divides $\mathtt{X}^\ell - 1$ in $R[\mathtt{X}]$ if and only if k divides ℓ.*

Proof. Let $\ell = kq + r$, with $0 \leq r < k$. We have

$$\mathtt{X}^k \equiv \mathtt{X}^{kq}\mathtt{X}^r \equiv \mathtt{X}^r \pmod{\mathtt{X}^k - 1},$$

and $\mathtt{X}^r \equiv 1 \pmod{\mathtt{X}^k - 1}$ if and only if $r = 0$. □

Theorem 19.2 *Let $a \geq 2$ be an integer and k, ℓ be positive integers. Then $a^k - 1$ divides $a^\ell - 1$ if and only if k divides ℓ.*

Proof. The proof is analogous to that of Theorem 19.1. We leave the details to the reader. □

One may combine these two theorems, obtaining:

Theorem 19.3 *Let $a \geq 2$ be an integer, k, ℓ be positive integers, and R a non-trivial ring. The polynomial $\mathtt{X}^{a^k} - \mathtt{X}$ divides $\mathtt{X}^{a^\ell} - \mathtt{X}$ in $R[\mathtt{X}]$ if and only if k divides ℓ.*

Proof. Because $\mathtt{X}$ is not a zero divisor, we have (verify) $\mathtt{X}^{a^k} - \mathtt{X}$ divides $\mathtt{X}^{a^\ell} - \mathtt{X}$ iff $\mathtt{X}^{a^k - 1} - 1$ divides $\mathtt{X}^{a^\ell - 1} - 1$, and by Theorem 19.1, this happens iff $a^k - 1$ divides $a^\ell - 1$, which by Theorem 19.2 happens iff k divides ℓ. □

19.3 The Existence of Finite Fields

We now get to the proof that there exists a finite field of every prime-power cardinality. We prove a somewhat more general theorem, however.

Throughout this section, F denotes a finite field of cardinality q. Of course, as we have shown, q must be a prime power, say $q = p^w$, and F is an extension field of degree w over $\mathbb{Z}_p$ (possibly, $F = \mathbb{Z}_p$). We shall show that for every $\ell \geq 1$, there exists an extension E of F of degree ℓ. Now, E is itself a finite extension of $\mathbb{Z}_p$, and so, as we have shown, $E = \mathbb{Z}_p[\gamma]$ for some $\gamma \in E$, from which it follows that $E = F[\gamma]$, and hence, E is isomorphic (as an F-algebra) to $F[\mathtt{X}]/(\phi)$, where ϕ is the minimal polynomial of γ over F. So the problem of proving the existence of such a field E is equivalent to proving the existence of an irreducible polynomial of degree ℓ over F.

We begin with a simple generalization of Fermat's Little Theorem (see Theorem 8.72 in §8.5).

Theorem 19.4 *For any $a \in F^*$, we have $a^{q-1} = 1$, and for any $a \in F$, we have $a^q = a$.*

Proof. The multiplicative group of units F^* of F contains $q-1$ elements, and hence, every $a \in F^*$ satisfies the equation $a^{q-1} = 1$. Multiplying this equation by a yields $a^q = a$ for all $a \in F^*$, and this latter equation obviously holds for $a = 0$ as well. □

The following theorem generalizes Example 14.49.

Theorem 19.5 *Let A be an F-algebra. Then the map $\rho : A \to A$ that sends $\alpha \in A$ to α^q is an F-algebra homomorphism.*

Proof. Since A is an F-algebra, it must have characteristic p. Since q is a power of the characteristic, the fact that ρ is a ring homomorphism follows from the discussion in Example 9.74. The fact that ρ is F-linear follows directly from Theorem 19.4. □

Theorem 19.6 *Let E be a finite extension of F, and consider the map $\sigma : E \to E$ that sends $\alpha \in E$ to $\alpha^q \in E$. Then σ is an F-algebra automorphism on E. Moreover, if $\alpha \in E$ is such that $\sigma(\alpha) = \alpha$, then $\alpha \in F$.*

Proof. The fact that σ is an F-algebra homomorphism follows from the previous theorem. Any ring homomorphism from a field into a field is injective (see Exercise 9.77). Surjectivity follows from injectivity and finiteness.

For the second statement, observe that $\sigma(\alpha) = \alpha$ if and only of α is a root of the polynomial $\mathtt{X}^q - \mathtt{X}$, since all q elements of F are already roots of this polynomial, there can be no other roots. □

The map σ defined in Theorem 19.6 is called the **Frobenius map on E over F**. Since the composition of two F-algebra automorphisms is also an F-algebra automorphism, for any $i \geq 0$, the map σ^i that sends $\alpha \in E$ to α^{q^i} is also an F-algebra automorphism.

Theorem 19.7 *We have*

$$\mathtt{X}^q - \mathtt{X} = \prod_{a \in F} (\mathtt{X} - a).$$

Proof. The polynomial

$$(\mathtt{X}^q - \mathtt{X}) - \prod_{a \in F} (\mathtt{X} - a)$$

has degree less than q, but has q distinct roots (every element of F), and hence must be the zero polynomial. □

Let P_k denote the product of all the monic irreducible polynomials in $F[\mathtt{X}]$ of degree k.

Theorem 19.8 *For all positive integers ℓ, we have*

$$\mathtt{X}^{q^\ell} - \mathtt{X} = \prod_{k \mid \ell} P_k,$$

where the product is over all divisors k of ℓ.

Proof. First, we claim that the polynomial $\mathtt{X}^{q^\ell} - \mathtt{X}$ is square-free, i.e., it is not divisible by the square of any non-constant polynomial f. Suppose it were, so that $\mathtt{X}^{q^\ell} - \mathtt{X} = f^2 g$. Taking formal derivatives, we see that

$$-1 = 2f\mathbf{D}(f)g + f^2\mathbf{D}(g).$$

But this is impossible, since it implies that f divides 1. That proves the claim.

So we have reduced the proof to showing that if f is a monic irreducible polynomial of degree k, then f divides $\mathtt{X}^{q^\ell} - \mathtt{X}$ if and only if $k \mid \ell$. Let $E = F[\mathtt{X}]/(f)$, and let η be a root of f in E.

For the first implication, assume that f divides $\mathtt{X}^{q^\ell} - \mathtt{X}$. We want to show that $k \mid \ell$. Now, if $\mathtt{X}^{q^\ell} - \mathtt{X} = fg$, then $\eta^{q^\ell} - \eta = f(\eta)g(\eta) = 0$, so $\eta^{q^\ell} = \eta$. Therefore, if σ is the Frobenius map on E over F, then we have $\sigma^\ell(\eta) = \eta$, and hence (by Theorem 14.50) $\sigma^\ell(\alpha) = \alpha$ for all $\alpha \in E$.

So every element of E is a root of $\mathtt{X}^{q^\ell} - \mathtt{X}$. That is, $\prod_{\alpha \in E}(\mathtt{X} - \alpha)$ divides $\mathtt{X}^{q^\ell} - \mathtt{X}$. Applying Theorem 19.7 to the field E, we see that $\prod_{\alpha \in E}(\mathtt{X} - \alpha) = \mathtt{X}^{q^k} - \mathtt{X}$, and hence $\mathtt{X}^{q^k} - \mathtt{X}$ divides $\mathtt{X}^{q^\ell} - \mathtt{X}$. By Theorem 19.3, this implies k divides ℓ.

For the second implication, suppose that $k \mid \ell$. We want to show that $f \mid \mathtt{X}^{q^\ell} - \mathtt{X}$. Since f is the minimal polynomial of η, and since η is a root of $\mathtt{X}^{q^k} - \mathtt{X}$, we must have that f divides $\mathtt{X}^{q^k} - \mathtt{X}$. Since $k \mid \ell$, and applying Theorem 19.3 once more, we see that $\mathtt{X}^{q^k} - \mathtt{X}$ divides $\mathtt{X}^{q^\ell} - \mathtt{X}$. That proves the second implication, and hence, the theorem. □

For $\ell \geq 1$, let $\Pi(\ell)$ denote the number of monic irreducible polynomials of degree ℓ in $F[\mathtt{X}]$.

Theorem 19.9 *For all $\ell \geq 1$, we have*

$$q^\ell = \sum_{k \mid \ell} k\Pi(k). \tag{19.1}$$

Proof. Just equate the degrees of both sides of the identity in Theorem 19.8. □

From Theorem 19.9 it is easy to deduce that $\Pi(\ell) > 0$ for all ℓ, and in fact, one can prove a density result — essentially a "prime number theorem" for polynomials over finite fields:

Theorem 19.10 *For all $\ell \geq 1$, we have*

$$\frac{q^\ell}{2\ell} \leq \Pi(\ell) \leq \frac{q^\ell}{\ell}, \tag{19.2}$$

and

$$\Pi(\ell) = \frac{q^\ell}{\ell} + O\left(\frac{q^{\ell/2}}{\ell}\right). \tag{19.3}$$

Proof. First, since all the terms in the sum on the right hand side of (19.1) are non-negative, and $\ell\Pi(\ell)$ is one of these terms, we may deduce that $\ell\Pi(\ell) \leq q^\ell$, which proves the second inequality in (19.2). Since this holds for all ℓ, we have

$$\ell\Pi(\ell) = q^\ell - \sum_{\substack{k|\ell \\ k<\ell}} k\Pi(k) \geq q^\ell - \sum_{\substack{k|\ell \\ k<\ell}} q^k \geq q^\ell - \sum_{k=1}^{\lfloor \ell/2 \rfloor} q^k.$$

Let us set

$$S(q,\ell) := \sum_{k=1}^{\lfloor \ell/2 \rfloor} q^k = \frac{q}{q-1}(q^{\lfloor \ell/2 \rfloor} - 1),$$

so that $\ell\Pi(\ell) \geq q^\ell - S(q,\ell)$. It is easy to see that $S(q,\ell) = O(q^{\ell/2})$, which proves (19.3). For the first inequality of (19.2), it suffices to show that $S(q,\ell) \leq q^\ell/2$. One can check this directly for $\ell \in \{1,2,3\}$ (verify), and for $\ell \geq 4$, we have

$$S(q,\ell) \leq q^{\ell/2+1} \leq q^{\ell-1} \leq q^\ell/2.$$

□

We note that the inequalities in (19.2) are tight, in the sense that $\Pi(\ell) = q^\ell/(2\ell)$ when $q = 2$ and $\ell = 2$, and $\Pi(\ell) = q^\ell$ when $\ell = 1$. The first inequality in (19.2) implies not only that $\Pi(\ell) > 0$, but that the fraction of all monic degree ℓ polynomials that are irreducible is at least $1/(2\ell)$, while (19.3) says that this fraction gets arbitrarily close to $1/\ell$ as either q or ℓ are sufficiently large.

Exercise 19.11 Starting from Theorem 19.9, show that

$$\Pi(\ell) = \ell^{-1} \sum_{k|\ell} \mu(k) q^{\ell/k},$$

where μ is the Möbius function (see §2.5). □

Exercise 19.12 How many irreducible polynomials of degree 30 over $\mathbb{Z}_2$ are there? □

In the proof of Theorem 19.8, we made use of a connection between formal derivatives and the square-freeness property for polynomials. The following exercise develops this connection more fully.

Exercise 19.13 Let F be an arbitrary field, and let $f \in F[\mathtt{X}]$ with $\deg(f) > 0$.

(a) Show that if f is not square-free, then $\gcd(f, \mathbf{D}(f)) \neq 1$.

(b) Show that if $\mathbf{D}(f) = 0$, then the characteristic of F must be a prime p, and f must be of the form $f = g(\mathtt{X}^p)$ for some $g \in F[\mathtt{X}]$.

(c) Show that if F is a finite field of characteristic p, and $f = g(\mathtt{X}^p)$, then $f = h^p$ for some $h \in F[\mathtt{X}]$; in fact, if $g = \sum_i g_i \mathtt{X}^i$, then $h = \sum_i g_i^{p^{(w-1)}} \mathtt{X}^i$, where $w := [F : \mathbb{Z}_p]$.

(d) Show that if F is a finite field or a field of characteristic zero, then f is square-free if and only if $d := \gcd(f, \mathbf{D}(f)) = 1$; moreover, if $d \neq 1$, then either $\deg(d) < \deg(f)$, or F has prime characteristic p and $f = h^p$ for some $h \in F[\mathtt{X}]$.

(e) Give an example of a field F of characteristic p and an irreducible polynomial $f \in F[\mathtt{X}]$ such that $f = g(\mathtt{X}^p)$ for some $g \in F[\mathtt{X}]$.

□

19.4 The Subfield Structure and Uniqueness of Finite Fields

We begin with a result that holds for field extensions in general.

Theorem 19.14 *Let E be an extension of a field F, and let σ be an F-algebra automorphism on E. Then the set $E' := \{\alpha \in E : \sigma(\alpha) = \alpha\}$ is a subfield of E containing F.*

Proof. By definition, σ acts as the identity function on F, and so $F \subset E'$, and in particular $1 \in E'$. To show that E' is closed under addition, let $\alpha, \beta \in E'$. Then $\sigma(\alpha+\beta) = \sigma(\alpha) + \sigma(\beta) = \alpha + \beta$, and hence $\alpha + \beta \in E'$. Replacing "+" by "·" in the above argument shows that E' is closed under multiplication. Finally, we need to show that if $0 \neq \alpha \in E'$ and $\beta \in E$ with $\alpha\beta = 1$, then $\beta \in E'$. But $\alpha\beta = 1$ implies $\sigma(\alpha)\sigma(\beta) = \sigma(1)$, which implies $\alpha\sigma(\beta) = 1$, and from this, it follows that $\sigma(\beta) = \beta$. □

The subfield E' in the above theorem is called **the subfield of E fixed by σ**. Turning our attention again to finite fields, the following theorem completely characterizes the subfield structure of a finite field.

Theorem 19.15 *Let E be an extension of degree ℓ of a finite field F, and let σ be the Frobenius map on E over F. Then the intermediate fields E', with $F \subset E' \subset E$, are in one-to-one correspondence with the divisors k of ℓ, where the divisor k corresponds to the subfield of E fixed by σ^k, which has degree k over F.*

Proof. Let q be of cardinality F. Let k be a divisor of ℓ. Now, by Theorem 19.7, the polynomial $\mathtt{X}^{q^\ell} - \mathtt{X}$ splits into distinct linear factors over E, and by Theorem 19.3, the polynomial $\mathtt{X}^{q^k} - \mathtt{X}$ divides $\mathtt{X}^{q^\ell} - \mathtt{X}$. Hence, $\mathtt{X}^{q^k} - \mathtt{X}$ also splits into distinct linear factors over E. This says that the subfield of E fixed by σ^k, which consists of the roots of $\mathtt{X}^{q^k} - \mathtt{X}$, has precisely q^k elements, and hence is an extension of degree k over F. That proves the existence part of the theorem.

As for uniqueness, we have to show that any intermediate is of this type. Let E' be an intermediate field of degree k over F. By Theorem 19.7, we have $\mathtt{X}^{q^k} - \mathtt{X} = \prod_{\alpha \in E'}(\mathtt{X} - \alpha)$ and $\mathtt{X}^{q^\ell} - \mathtt{X} = \prod_{\alpha \in E}(\mathtt{X} - \alpha)$, from which it follows that $\mathtt{X}^{q^k} - \mathtt{X}$ divides $\mathtt{X}^{q^\ell} - \mathtt{X}$, and so by Theorem 19.3, we must have $k \mid \ell$. There can be no other intermediate fields of the same degree k over F, since the elements of such a field would also be roots of $\mathtt{X}^{q^k} - \mathtt{X}$. □

The next theorem shows that up to isomorphism, there is only one finite field of a given cardinality.

Theorem 19.16 *Let E, E' be extensions of the same degree over a finite field F. Then E and E' are isomorphic as F-algebras.*

Proof. Let ℓ be the degree of the extensions. As we have argued before, we have $E' = F[\alpha']$ for some $\alpha' \in E'$, and so E' is isomorphic as an F-algebra to $F[\mathtt{X}]/(\phi)$, where ϕ is the minimal polynomial of α' over F. As ϕ is an irreducible polynomial of degree ℓ, by Theorem 19.8, ϕ divides $\mathtt{X}^{q^\ell} - \mathtt{X}$, and by Theorem 19.7, $\mathtt{X}^{q^\ell} - \mathtt{X} = \prod_{\alpha \in E}(\mathtt{X} - \alpha)$, from which it follows that ϕ has a root $\alpha \in E$. Since ϕ is irreducible, ϕ is the minimal polynomial of α over F, and hence $F[\alpha]$ is isomorphic as an F-algebra to $F[\mathtt{X}]/(\phi)$. Since α has degree ℓ over F, we must have $E = F[\alpha]$. □

Exercise 19.17 This exercise develops an alternative proof for the existence of finite fields — however, it does not yield a density result for irreducible polynomials. Let F be a finite field of cardinality q, and let $\ell \geq 1$ be an integer. Let E be a splitting field for the polynomial $\mathtt{X}^{q^\ell} - \mathtt{X} \in F[\mathtt{X}]$ (see Theorem 17.23). Let E' be the subfield of E fixed by the q^ℓth power map. Show that E' is an extension of F of degree ℓ. □

Exercise 19.18 Let E be an extension of degree ℓ over a finite field F of cardinality q. Show that at least half the elements of E have degree ℓ over F, and that the total number of elements of degree ℓ over F is $q^\ell + O(q^\ell/2)$. □

19.5 Conjugates, Norms and Traces

Throughout this section, F denotes a finite field of cardinality q, E denotes an extension over F of degree ℓ, and σ denotes the Frobenius map on E over F.

For any non-negative integer i, we can define the function σ^i, obtained by composing σ with itself i times, which is also an F-algebra automorphism. The inverse function σ^{-1} is also an F-algebra automorphism, as is σ^i for negative values of i, defined by composing σ^{-1} with

itself $|i|$ times. Under the operation of function composition, the set $G_{E/F} = \{\sigma^i : i \in \mathbb{Z}\}$ forms an abelian group, as the reader may easily verify. Indeed, $G_{E/F}$ is a cyclic group generated by σ. Moreover, σ^ℓ is the identity function, and σ^i for $0 < i < \ell$ cannot be the identity function, since then the polynomial $\mathtt{X}^{q^i} - \mathtt{X}$ would have too many roots. We summarize these observations as follows:

Theorem 19.19 *The set $G_{E/F} := \{\sigma^i : i \in \mathbb{Z}\}$ forms a group with respect to the operation of function composition. Moreover, $G_{E/F}$ is isomorphic to the cyclic group $\mathbb{Z}_\ell$, via the group isomorphism that sends $[i \bmod \ell]$ to σ^i. In particular, the distinct elements of $G_{E/F}$ are σ^i for $0 \leq i < \ell$.*

This group $G_{E/F}$ is called the **Galois group of E over F**.

Consider an element $\alpha \in E$. We say that $\beta \in E$ is **conjugate to α (over F)** if $\beta = \sigma^i(\alpha)$ for some $i \in \mathbb{Z}$. The reader may verify that the "conjugate to" relation is an equivalence relation. We call the equivalence classes of this relation **conjugacy classes**, and we call the elements of the conjugacy class containing α the **conjugates of α**.

Consider the set I_α of all integers i such that $\sigma^i(\alpha) = \alpha$. We claim that I_α is a subgroup of the additive group of integers. Indeed, if $\sigma^i(\alpha) = \alpha$ and $\sigma^j(\alpha) = \alpha$, then

$$\sigma^{i+j}(\alpha) = \sigma^i(\sigma^j(\alpha)) = \sigma^i(\alpha) = \alpha$$

and

$$\alpha = \sigma^{-i}(\sigma^i(\alpha)) = \sigma^{-i}(\alpha).$$

It follows that $I_\alpha = k\mathbb{Z}$ for some non-negative integer k. Moreover, it is clear that $\ell \in I_\alpha$, and so we have $k \mid \ell$. Further, all the conjugates of α are of the form $\sigma^i(\alpha)$ for $0 \leq i < k$, since for any conjugate $\sigma^j(\alpha)$, we can write $j = ka + i$ for $0 \leq i < k$, and

$$\sigma^j(\alpha) = \sigma^i(\sigma^{ka}(\alpha)) = \sigma^i(\alpha).$$

Finally, all of the conjugates $\sigma^i(\alpha)$ for $0 \leq i < k$ are distinct, since $\sigma^i(\alpha) = \sigma^j(\alpha)$ implies that $\sigma^{i-j}(\alpha) = \alpha$, and hence $k \mid (i - j)$.

With α and k as above, consider the polynomial ϕ

$$\phi := \prod_{i=0}^{k-1} (\mathtt{X} - \sigma^i(\alpha)).$$

The coefficients of ϕ obviously lie in E, but we claim that in fact, they lie in F. This is easily seen as follows. Consider the extension of the map σ from E to $E[\mathtt{X}]$ that applies σ coefficient-wise to polynomials. This was discussed in Example 9.68, where we saw that the extended map, which we also denote by σ, is a ring homomorphism from $E[\mathtt{X}]$ into $E[\mathtt{X}]$. Applying σ to ϕ, we obtain

$$\sigma(\phi) = \prod_{i=0}^{k-1} \sigma(\mathtt{X} - \sigma^i(\alpha)) = \prod_{i=0}^{k-1} (\mathtt{X} - \sigma^{i+1}(\alpha)) = \prod_{i=0}^{k-1} (\mathtt{X} - \sigma^i(\alpha)),$$

since $\sigma^k(\alpha) = \alpha$. Thus we see that $\sigma(\phi) = \phi$. Writing $\phi = \sum_i a_i \mathtt{X}^i$, we see that $\sigma(a_i) = a_i$ for all i, and hence by Theorem 19.6, $a_i \in F$ for all i. Hence $\phi \in F[\mathtt{X}]$. We further claim that ϕ is the minimal polynomial of α. To see this, let $f \in F[\mathtt{X}]$ be any polynomial over F with α as a root. Then for any integer j, by Theorem 14.50, we have

$$0 = \sigma^j(0) = \sigma^j(f(\alpha)) = f(\sigma^j(\alpha)).$$

Thus, the conjugates of α are also roots of f, and so ϕ divides f. Since ϕ is the minimal polynomial of α and $\deg(\phi) = k$, it follows that the number k is none other than the degree of α over F.

Let us summarize the above discussion as follows:

Theorem 19.20 *Let $\alpha \in E$ be of degree k over F, and let ϕ be the minimal polynomial of α over F. Then k is the smallest positive integer such that $\sigma^k(\alpha) = \alpha$, the distinct conjugates of α are $\sigma^i(\alpha)$ for $0 \leq i < k$, and and ϕ factors over E (in fact, over $F[\alpha]$) as*

$$\phi = \prod_{i=0}^{k-1} (\mathtt{X} - \sigma^i(\alpha)).$$

Another useful way of reasoning about conjugates is as follows. First, if $\alpha = 0$, then the degree of α over F is 1, and there is nothing more to say, so let us assume that $\alpha \in E^*$. If r is the multiplicative order of α, then note that any conjugate $\sigma^i(\alpha)$ also has multiplicative order r — this follows from the fact that for any positive integer s, $\alpha^s = 1$ if and only if $(\sigma^i(\alpha))^s = 1$. Also, note that we must have $r \mid |E^*| = q^\ell - 1$, i.e., $q^\ell \equiv 1 \pmod{r}$. Focusing now on the fact that σ is the q-power map, we see that the degree k of α is the smallest positive integer such that $\alpha^{q^k} = \alpha$, which holds iff $\alpha^{q^k - 1} = 1$, which holds iff $q^k \equiv 1 \pmod{r}$. Thus, the degree of α over F is simply the multiplicative order of q modulo r. Again, we summarize these observations as a theorem:

Theorem 19.21 *If $\alpha \in E^*$ has multiplicative order r, then the degree of α over F is equal to the multiplicative order of q modulo r.*

Let us define the polynomial

$$\chi := \prod_{i=0}^{\ell-1} (\mathtt{X} - \sigma^i(\alpha)).$$

It is easy to see, using the same type of argument as above, that $\chi \in F[\mathtt{X}]$, and indeed, that

$$\chi = \phi^{\ell/k}.$$

The polynomial χ is called the **characteristic polynomial of α with respect to the extension E of F**.

Two functions that are often useful are the "norm" and "trace." The **norm of α with respect to the extension E of F** is defined as

$$\mathbf{N}_{E/F}(\alpha) := \prod_{i=0}^{\ell-1} \sigma^i(\alpha),$$

while the **trace of α with respect to the extension E of F** is defined as

$$\mathbf{Tr}_{E/F}(\alpha) := \sum_{i=0}^{\ell-1} \sigma^i(\alpha).$$

It is easy to see that both the norm and trace of α are elements of F, as they are fixed by σ; alternatively, one can see this by observing that they appear, possibly with a minus sign, as coefficients of the characteristic polynomial χ — indeed, the constant term of χ is equal to $(-1)^\ell \mathbf{N}_{E/F}(\alpha)$, and the coefficient of $\mathtt{X}^{\ell-1}$ in χ is $-\mathbf{Tr}_{E/F}(\alpha)$.

The following two theorems summarize the most important facts about the norm and trace functions.

Theorem 19.22 *The function $\mathbf{N}_{E/F}$, restricted to E^*, is a group homomorphism from E^* onto F^*.*

Proof. We have

$$\mathbf{N}_{E/F}(\alpha) = \prod_{i=0}^{\ell-1} \alpha^{q^i} = \alpha^{\sum_{i=0}^{\ell-1} q^i} = \alpha^{(q^\ell-1)/(q-1)}.$$

Since E^* is a cyclic group of order $q^\ell - 1$, the image of the $(q^\ell - 1)/(q-1)$-power map on E^* is the unique subgroup of E^* of order $q-1$ (see Theorem 8.75). Since F^* is a subgroup of E^* of order $q-1$, it follows that the image of this power map is F^*. □

Theorem 19.23 *The function $\mathbf{Tr}_{E/F}$ is an F-linear map from E onto F.*

Proof. The fact that $\mathbf{Tr}_{E/F}$ is an F-linear map is a simple consequence of the fact that σ is an F-algebra automorphism (verify). As discussed above, $\mathbf{Tr}_{E/F}$ maps into F. Since the image of $\mathbf{Tr}_{E/F}$ is a subspace of F, the image is either $\{0\}$ or F, and so it suffices to show that $\mathbf{Tr}_{E/F}$ does not map all of E to zero. But an element $\alpha \in E$ is in the kernel of $\mathbf{Tr}_{E/F}$ if and only of α is a root of the polynomial

$$\mathtt{X} + \mathtt{X}^q + \cdots + \mathtt{X}^{q^{\ell-1}},$$

which has degree $q^{\ell-1}$. Since E contains q^ℓ elements, not all elements of E can lie in the kernel of $\mathbf{Tr}_{E/F}$. □

Example 19.24 As an application of some of the above theory, let us investigate the factorization of the polynomial $\mathtt{X}^r - 1$ over F, a finite field of cardinality q. Let us assume that $r > 0$ and is relatively prime to q. Let E be a splitting field of $\mathtt{X}^r - 1$ (see Theorem 17.23), so that E is a finite extension of F in which $\mathtt{X}^r - 1$ splits into linear factors:

$$\mathtt{X}^r - 1 = \prod_{i=1}^{r} (\mathtt{X} - \alpha_i).$$

We claim that the roots α_i of $\mathtt{X}^r - 1$ are distinct — this follows from the Exercise 19.13 and the fact that $\gcd(\mathtt{X}^r - 1, r\mathtt{X}^{r-1}) = 1$.

Next, observe that the r roots of $\mathtt{X}^r - 1$ in E actually form a subgroup of E^*, and since E^* is cyclic, this subgroup must be cyclic as well. So the roots of $\mathtt{X}^r - 1$ form a cyclic subgroup of E of order r. Let ζ be a generator for this group. Then all the roots of $\mathtt{X}^r - 1$ are contained in $F[\zeta]$, and so we may as well assume that $E = F[\zeta]$.

Let us compute the degree of ζ over F. By Theorem 19.21, the degree ℓ of ζ over F is the multiplicative order of q modulo r. Moreover, the $\phi(r)$ roots of $\mathtt{X}^r - 1$ of multiplicative order r are partitioned into $\phi(r)/\ell$ conjugacy classes, each of size ℓ; indeed, as the reader is urged to verify, these conjugacy classes are in one-to-one correspondence with the cosets of the subgroup generated by $[q \bmod r]$ in $\mathbb{Z}_r^*$, where each such coset $C \subset \mathbb{Z}_r^*$ corresponds to the conjugacy class $\{\zeta^a : a \in C\}$.

More generally, for any $s \mid r$, any root of $\mathtt{X}^r - 1$ whose multiplicative order is s has degree k over F, where k is the multiplicative order of q modulo s. As above, the $\phi(s)$ roots of multiplicative order s are partitioned into $\phi(s)/k$ conjugacy classes, which are in one-to-one correspondence with the cosets of the subgroup generated by $[q \bmod s]$ in $\mathbb{Z}_s^*$.

This tells us exactly how $\mathtt{X}^r - 1$ splits into irreducible factors over F. Things are a bit simpler when r is prime, in which case, from the above discussion, we see that

$$\mathtt{X}^r - 1 = (\mathtt{X} - 1) \prod_{i=1}^{(r-1)/\ell} f_i,$$

where each f_i is an irreducible polynomial of degree ℓ, and ℓ is the multiplicative order of q modulo r.

In the above analysis, instead of constructing the field E using Theorem 17.23, one could instead simply construct E as $F[\mathtt{X}]/(\phi)$, where ϕ is any irreducible polynomial of degree ℓ, where ℓ is the multiplicative order of q modulo r. We know that such a polynomial ϕ exists by Theorem 19.10, and since E has cardinality q^ℓ, and $r \mid (q^\ell - 1) = |E^*|$, and E^* is cyclic, we know that E^* contains an element ζ of order r, and each of the r distinct powers of ζ are roots of $\mathtt{X}^r - 1$, and so this E is a splitting field $\mathtt{X}^r - 1$ over F. □

Exercise 19.25 Let E be a finite extension of a finite field F. Show that for $a \in F$, we have $\mathbf{N}_{E/F}(a) = a^\ell$ and $\mathbf{Tr}_{E/F}(a) = \ell a$. □

Exercise 19.26 Let E be a finite extension of a finite field F. Let E' be an intermediate field, $F \subset E' \subset E$. Show that

(a) $\mathbf{N}_{E/F}(\alpha) = \mathbf{N}_{E'/F}(\mathbf{N}_{E/E'}(\alpha))$, and

(b) $\mathbf{Tr}_{E/F}(\alpha) = \mathbf{Tr}_{E'/F}(\mathbf{Tr}_{E/E'}(\alpha))$.

□

Exercise 19.27 Let F be a finite field, and let $f \in F[\mathtt{X}]$ be a monic irreducible polynomial of degree ℓ. Let $E = F[\mathtt{X}]/(f) = F[\eta]$, where $\eta := [\mathtt{X} \bmod f]$. Show that

$$\frac{\mathbf{D}(f)}{f} = \sum_{j=1}^{\infty} \mathbf{Tr}_{E/F}(\eta^{j-1})\mathtt{X}^{-j}.$$

Hint: use Exercise 17.35. □

Exercise 19.28 Let F be a finite field, and $f \in F[\mathtt{X}]$ an irreducible polynomial of degree k over F. Let E be an extension of degree ℓ over F. Show that over E, f factors as the product of d distinct irreducible polynomials, each of degree k/d, where $d = \gcd(k, \ell)$. □

Exercise 19.29 Let E be a finite extension of a finite field F of characteristic p. Show that if $\alpha \in E$ and $0 \neq a \in F$, and if α and $\alpha + a$ are conjugate over F, then p divides the degree of α over F. □

Exercise 19.30 Let F be a finite field of characteristic p. For $a \in F$, consider the polynomial $f := \mathtt{X}^q - \mathtt{X} - a \in F[\mathtt{X}]$.

(a) Show that if $F = \mathbb{Z}_p$ and $a \neq 0$, then f is irreducible.

(b) More generally, show that if $\mathbf{Tr}_{F/\mathbb{Z}_p}(a) \neq 0$, then f is irreducible, and otherwise, f splits into distinct linear factors over F.

□

Exercise 19.31 Let E be a finite extension of a finite field F. Let $\alpha, \beta \in E$, where α has degree a over F, β has degree b over F, and $\gcd(a, b) = 1$. Show that $\alpha + \beta$ has degree ab over F. □

Exercise 19.32 Let E be a finite extension of a finite field F. Show that any F-algebra automorphism on E must be an element of the Galois group $G_{E/F}$. □

Exercise 19.33 Show that for all primes p, the polynomial $\mathtt{X}^4 + 1$ is reducible in $\mathbb{Z}_p[\mathtt{X}]$. (Contrast this to the fact that this polynomial is irreducible in $\mathbb{Q}[\mathtt{X}]$, as discussed in Exercise 17.65.) □

Chapter 20

Algorithms for Finite Fields

This chapter discusses efficient algorithms for factoring polynomials over finite fields, and related problems, such as testing if a given polynomial is irreducible, and generating an irreducible polynomial of given degree.

Throughout this chapter, F denotes a finite field of cardinality q and characteristic p, where $q = p^w$ for some positive integer w.

In addition to performing the usual arithmetic and comparison operations in F, we assume that our algorithms have access to the numbers p, w, and q, and have the ability to generate random elements of F. Generating such a random field element will count as one "operation in F," along with the usual arithmetic operations. Of course, the "standard" way of representing F as either $\mathbb{Z}_p$ (if $w = 1$), or as the ring of polynomials modulo an irreducible polynomial over $\mathbb{Z}_p$ of degree w (if $w > 1$), satisfy the above requirements, and also allow for the implementation of arithmetic operations in F that take time $O(\text{len}(q)^2)$ on a RAM (using simple, quadratic-time arithmetic for polynomials and integers).

20.1 Testing and Constructing Irreducible Polynomials

Let $f \in F[\mathtt{X}]$ be a monic polynomial of degree $\ell > 0$. We develop here an efficient algorithm that determines if f is irreducible.

The idea is a simple application of Theorem 19.8. That theorem says that for any integer $k \geq 1$, the polynomial $\mathtt{X}^{q^k} - \mathtt{X}$ is the product of all monic irreducibles whose degree divides k. Thus, $\gcd(\mathtt{X}^q - \mathtt{X}, f)$ is product of all the distinct linear factors of f. If f has no linear factors, then $\gcd(\mathtt{X}^{q^2} - \mathtt{X}, f)$ is the product of all the distinct quadratic irreducible factors of f. And so on. Now, if f is not irreducible, it must be divisible by some irreducible polynomial of degree at most $\ell/2$, and if g is an irreducible factor of f of minimal degree, say k, then we have $k \leq \ell/2$ and $\gcd(\mathtt{X}^{q^k} - \mathtt{X}, f) \neq 1$. Conversely, if f is irreducible, then $\gcd(\mathtt{X}^{q^k} - \mathtt{X}, f) = 1$ for all $1 \leq k \leq \ell/2$. So to test if f is irreducible, it suffices to check if $\gcd(\mathtt{X}^{q^k} - \mathtt{X}, f) = 1$ for all $1 \leq k \leq \ell/2$ — if so, we may conclude that f is irreducible, and otherwise, we may conclude that f is not irreducible. To carry out the computation efficiently, we note that if $h \equiv \mathtt{X}^{q^k} \pmod{f}$, then $\gcd(h - \mathtt{X}, f) = \gcd(\mathtt{X}^{q^k} - \mathtt{X}, f)$.

The above observations suggest the following algorithm, which takes as input a monic polynomial $f \in F[\mathtt{X}]$ of degree $\ell > 0$, and outputs *true* if f is irreducible, and *false* otherwise:

Algorithm IPT:

$h \leftarrow \mathtt{X} \bmod f$
$k \leftarrow 1$
while $k \leq \lfloor \ell/2 \rfloor$ do
    $h \leftarrow h^q \bmod f$
    if $\gcd(h - \mathtt{X}, f) \neq 1$ then return *false*
    $k \leftarrow k + 1$
return *true*

The correctness of algorithm IPT follows immediately from the above discussion. As for the running time, we have:

Theorem 20.1 *Algorithm IPT uses $O(\ell^3 \operatorname{len}(q))$ operations in F.*

Proof. Consider an execution of a single iteration of the main loop. The cost of the qth-powering step (using a standard repeated-squaring algorithm) is $O(\operatorname{len}(q))$ operations mod f, and so $O(\ell^2 \operatorname{len}(q))$ operations in F. The cost of the gcd computation is $O(\ell^2)$ operations in F. Thus, the cost for a single loop iteration is $O(\ell^2 \operatorname{len}(q))$ operations in F, from which it follows that the cost for the entire algorithm is $O(\ell^3 \operatorname{len}(q))$ operations in F. □

Algorithm IPT is a "polynomial time" algorithm, since the length of the binary encoding of the input is about $\ell \operatorname{len}(q)$, and so the algorithm runs in time polynomial in its input length, assuming that arithmetic operations in F run take time polynomial in $\operatorname{len}(q)$. Indeed, using a standard representation for F, each operation in F takes time $O(\operatorname{len}(q)^2)$ on a RAM, and so the running time on a RAM for the above algorithm would be $O(\ell^3 \operatorname{len}(q)^3)$, i.e., cubic in the bit-length of the input.

Let us now consider the related problem of constructing an irreducible polynomial of specified degree $\ell > 0$. To do this, we can simply use the result of Theorem 19.10, which has the following probabilistic interpretation: if we choose a random, monic polynomial f of degree ℓ over F, then the probability that f is irreducible is $\Theta(1/\ell)$. This suggests the following probabilistic algorithm:

Algorithm RIP:

repeat
    choose $f_0, \ldots, f_{\ell-1} \in F$ at random
    set $f \leftarrow \mathtt{X}^\ell + \sum_{i=0}^{\ell-1} f_i \mathtt{X}^i$
    test if f is irreducible using algorithm IPT
until f is irreducible
output f

Theorem 20.2 *Algorithm RIP uses an expected number of $O(\ell^4 \operatorname{len}(q))$ operations in F, and its output is uniformly distributed over all monic irreducibles of degree ℓ.*

Proof. Because of Theorem 19.10, the expected number of loop iterations of the above algorithm is $O(\ell)$. Since algorithm IPT uses $O(\ell^3 \operatorname{len}(q))$ operations in F, the statement about the running time of algorithm RIP is immediate. The statement about its output distribution is clear. □

The expected running-time estimate in Theorem 20.2 is actually a bit of an overestimate. The reason is that if we generate a random polynomial of degree ℓ, it is likely to have a small irreducible factor, which will be discovered much more rapidly by algorithm IPT. In fact, it is known that the expected value of the least degree irreducible factor of a random monic polynomial of degree ℓ over F is $O(\operatorname{len}(\ell))$, from which it follows that the expected number of operations in F performed by algorithm RIP is actually $O(\ell^3 \operatorname{len}(\ell) \operatorname{len}(q))$.

Example 20.3 We consider, for the third and final time, the problem in Examples 18.11 and 18.24: F is a field, $f \in F[\mathtt{X}]$ is a monic irreducible polynomial of degree ℓ, and $E := F[\mathtt{X}]/(f) = F[\eta]$, where $\eta := [\mathtt{X} \bmod f]$; we are given an element $\alpha \in E$, and want to compute the minimal polynomial $\phi \in F[\mathtt{X}]$ of α over F. We develop an alternative algorithm, based on the theory of finite fields. Unlike the algorithms in Examples 18.11 and 18.24, which in principle work over any field F, the algorithm we develop here *only* works when F is a finite field.

Let q be the cardinality of F. From Theorem 19.20, we know that the degree of α over F is the smallest positive integer k such that $\alpha^{q^k} = \alpha$. By successive qth powering, we can compute the conjugates of α using $O(k \operatorname{len}(q))$ operations in E, and hence $O(k\ell^2 \operatorname{len}(q))$ operations in F.

Now, we could simply compute the minimal polynomial ϕ by directly using the formula

$$\phi(\mathtt{Y}) = \prod_{i=0}^{k-1} (\mathtt{Y} - \alpha^{q^i}). \tag{20.1}$$

This would involve computations with polynomials in the variable $\mathtt{Y}$ whose coefficients lie in the extension field E, although at the end of the computation, we would end up with a polynomial all of whose coefficients lie in F. The cost of this approach would be $O(k^2)$ operations in E, and hence $O(k^2\ell^2)$ operations in F.

A better approach is the following. Substituting η for $\mathtt{Y}$ in the identity (20.1), we have

$$\phi(\eta) = \prod_{i=0}^{k-1} (\eta - \alpha^{q^i}).$$

Using this formula, we we can compute (given the conjugates of α) the value $\phi(\eta) \in E$ using $O(k)$ operations in E, and hence $O(k\ell^2)$ operations in F. Now, $\phi(\eta)$ is an element

of E, and for computational purposes, it is represented as $[g \bmod f]$ for some polynomial $g \in F[\mathtt{X}]$ of degree less than ℓ. Moreover, $\phi(\eta) = [\phi \bmod f]$, and hence $\phi \equiv g \pmod f$. In particular, if $k < \ell$, then $g = \phi$; otherwise, if $k = \ell$, then $g = \phi - f$. In either case, we can recover ϕ from g with an additional $O(\ell)$ operations in F.

Thus, given the conjugates of α, we can compute ϕ using $O(k\ell^2)$ operations in F. Adding in the cost of computing the conjugates, this gives rise to an algorithm that computes the minimal polynomial of α using $O(k\ell^2 \operatorname{len}(q))$ operations in F.

In the worst case, then, this algorithm uses $O(\ell^3 \operatorname{len}(q))$ operations in F. A reasonably careful implementation needs space for storing a constant number elements of E, and hence $O(\ell)$ elements of F. For very small values of q, the efficiency of this algorithm will be comparable to that of the algorithm in Example 18.24, but for large q, it will be much less efficient. Thus, this approach does not really yield a better algorithm, but it does serve to illustrate some of the ideas of the theory of finite fields. □

Exercise 20.4 Let F be a finite field. Design and analyze a *deterministic* algorithm that takes as input a list of irreducible polynomials $f_1, \ldots, f_r \in F[\mathtt{X}]$, where $\ell_i := \deg(f_i)$ for $1 \leq i \leq r$, and $\ell := \sum_{i=1}^{r} \ell_i$. Assuming that the degrees $\ell_1, \ldots, \ell_r$ are pair-wise co-prime, your algorithm should output an irreducible polynomial $f \in F[\mathtt{X}]$ of degree ℓ using $O(\ell^3)$ operations in F. □

Exercise 20.5 Let F be a finite field, let $f \in F[\mathtt{X}]$ be a monic irreducible polynomial of degree $\ell > 0$, and let $E := F[\mathtt{X}]/(f)$, where $\eta := [\mathtt{X} \bmod f]$. Design and analyze a deterministic algorithm that takes as input the polynomial f defining the extension E, and outputs the values

$$s_j := \mathbf{Tr}_{E/F}(\eta^j) \in F \quad (j = 0, \ldots, \ell - 1)$$

using $O(\ell^2)$ operations in F. Show that given an arbitrary $\alpha \in E$, along with the values $s_0, \ldots, s_{\ell-1}$, one can compute $\mathbf{Tr}_{E/F}(\alpha)$ using just $O(\ell)$ operations in F. □

Exercise 20.6 Let F be a finite field. Design and analyze a probabilistic algorithm that given a monic irreducible polynomial $f \in F[\mathtt{X}]$ of degree ℓ as input, generates as output a random monic irreducible polynomial $g \in F[\mathtt{X}]$ of degree ℓ (i.e., g should be uniformly distributed over all such polynomials), using an expected number of $O(\ell^{2.5})$ operations in F. □

Exercise 20.7 Let F be a finite field of cardinality q. Let $f \in F[\mathtt{X}]$ be a monic polynomial of degree $\ell > 0$. Also, let $\eta := [\mathtt{X} \bmod f] \in A$, where A is the F-algebra $A := F[\mathtt{X}]/(f)$.

(a) Show how to compute — given as input $\beta \in A$ and $\eta^{q^m} \in A$ (for some integer $m > 0$) — the value $\beta^{q^m} \in A$, using just $O(\ell^{2.5})$ operations in F. Hint: see Theorems 14.50 and 19.5.

(b) Show how to compute — given as input $\eta^q \in A$ as above and a positive integer m — the value $\eta^{q^m} \in A$ using $O(\ell^{2.5} \operatorname{len}(m))$ operations in F.

□

Exercise 20.8 Let F be a finite field of cardinality q.

(a) Show that a monic polynomial $f \in F[\mathtt{X}]$ of degree $\ell > 0$ is irreducible if and only if $\mathtt{X}^{q^\ell} \equiv \mathtt{X} \pmod{f}$ and $\gcd(\mathtt{X}^{q^{\ell/s}} - \mathtt{X}, f) = 1$ for all primes $s \mid \ell$.

(b) Using part (a) and the result of the previous exercise, show how to determine if f is irreducible using $O(\ell^{2.5}\operatorname{len}(\ell)k + \ell^2 \operatorname{len}(q))$ operations in F, where k is the number of distinct prime factors of ℓ.

(c) Show that the operation count in part (b) can be reduced to $O(\ell^{2.5}\operatorname{len}(\ell)\operatorname{len}(k) + \ell^2 \operatorname{len}(q))$. Hint: see Exercise 11.2.

□

20.2 Factoring Polynomials over Finite Fields: the Cantor-Zassenhaus Algorithm

In this section and the next, we develop efficient algorithms for factoring polynomials over the finite field F.

The algorithm we discuss in this section is due to Cantor and Zassenhaus. The algorithm has two stages:

distinct degree factorization: The input polynomial is decomposed into factors so that each factor is a product of distinct irreducibles of the same degree (and the degree of those irreducibles is also determined).

equal degree factorization: Each of the factors produced in the distinct degree factorization stage are further factored into their irreducible factors.

The algorithm we present for distinct degree factorization is a deterministic, polynomial-time algorithm. The algorithm we present for equal degree factorization is a *probabilistic* algorithm that runs in expected polynomial time (and whose output is always correct).

20.2.1 Distinct degree factorization

The problem, more precisely stated, is this: given a monic polynomial $f \in F[\mathtt{X}]$ of degree ℓ, produce a list of pairs (g, k), where

- each $g \in F[\mathtt{X}]$ is a product of distinct monic irreducible polynomials of degree k, and
- the product of all the g's in the list is equal to f.

This problem can be easily solved using Theorem 19.8, using a simple variation of the algorithm we discussed in §20.1 for irreducibility testing. The basic idea is this. We can compute $g := \gcd(\mathtt{X}^q - \mathtt{X}, f)$, so that g is the product of all the distinct linear factors of f. We can remove the factor g from f, but after doing so, f may still contain some linear factors (if the original polynomial was not square-free), and so we have to repeat the above step until no linear factors are discovered. Having removed all linear factors from f, we next compute $\gcd(\mathtt{X}^{q^2} - \mathtt{X}, f)$, which will be the product of all the distinct quadratic irreducible dividing f, and we can remove these from f — although $\mathtt{X}^{q^2} - \mathtt{X}$ is the product of all linear and quadratic irreducibles, since we have already removed the linear factors from f, the gcd will give us just the quadratic factors of f. As above, we may have to repeat this a few times to remove all the quadratic factors from f. In general, for $1 \leq k \leq \ell$, having removed all the irreducible factors of degree less than k from f, we compute $\gcd(\mathtt{X}^{q^k} - \mathtt{X}, f)$ to obtain the product of all the distinct irreducible factors of f of degree k, repeating as necessary to remove all such factors.

The above discussion yields the following algorithm, which takes as input a monic polynomial $f \in F[\mathtt{X}]$ of degree $\ell > 0$:

Algorithm DDF:

$h \leftarrow \mathtt{X} \bmod f$
$k \leftarrow 1$
while $f \neq 1$ do
    $h \leftarrow h^q \bmod f$
    $g \leftarrow \gcd(h - \mathtt{X}, f)$
    while $g \neq 1$ do
        output (g, k)
        $f \leftarrow f/g$
        $h \leftarrow h \bmod f$
        $g \leftarrow \gcd(h - \mathtt{X}, f)$
    $k \leftarrow k + 1$

The correctness of algorithm DDF follows from the discussion above. As for the running time:

Theorem 20.9 *Algorithm DDF uses $O(\ell^3 \operatorname{len}(q))$ operations in F.*

Proof. Note that the body of the outer loop is executed at most ℓ times, since after ℓ iterations, we will have removed all the factors of f. Thus, we perform at most ℓ qth-powering steps, each of which takes $O(\ell^2 \operatorname{len}(q))$ operations in F, and so the total contribution to the running time of these is $O(\ell^3 \operatorname{len}(q))$ operations in F. We also have to take into account the cost of the gcd's. We perform one gcd operation in every iteration of the main loop, for a

total of ℓ such operations. We also perform an "extra" gcd operation whenever we discover a non-trivial factor of f; however, since we only discover at most ℓ such non-trivial factors, we perform at most ℓ such "extra" gcd operations. So the total number of gcd operations is at most 2ℓ, and as each of these takes $O(\ell^2)$ operations in F, they contribute a term of $O(\ell^3)$ to the total operation count. This term is dominated by the cost of the qth-powering steps (as is the cost of the division step in the inner loop), and so the total cost of algorithm DDF is $O(\ell^3 \operatorname{len}(q))$ operations in F. $\square$

20.2.2 Equal degree factorization

The problem, more precisely stated, is this: given a monic polynomial $g \in F[\mathtt{X}]$ of degree $\ell > 0$, and an integer $k > 0$, such that g is of the form

$$g = g_1 \cdots g_r$$

for distinct monic irreducible polynomials $g_1, \ldots, g_r$, compute these irreducible factors of g. Note that given g and k, the value of r easily determined, $r = \ell/k$.

If $r = 1$, we have nothing to do. So assume that $r > 1$.

By the Chinese Remainder Theorem, we have an F-algebra isomorphism

$$\rho : E_1 \times \cdots \times E_r \to A,$$

where for $1 \leq i \leq r$, E_i is the extension field $F[\mathtt{X}]/(g_i)$ of degree k over F, and A is the F-algebra $A := F[\mathtt{X}]/(g)$.

We have to treat the cases $p = 2$ and $p > 2$ separately. We first treat the case $p = 2$. Let us define the function $\mathcal{F} : A \to A$ that sends $\alpha \in A$ to $\sum_{i=0}^{wk-1} \alpha^{2^i}$ (the algorithm in the case $p > 2$ will only differ in the definition of $\mathcal{F}$). Note that each E_i is an extension of $\mathbb{Z}_2$ of degree wk. For $\alpha \in A$, if $\alpha = \rho(\alpha_1, \ldots, \alpha_r)$, then, just using the fact that ρ is a ring homomorphism, we have

$$\begin{aligned}
\mathcal{F}(\alpha) &= \sum_i (\rho(\alpha_1, \ldots, \alpha_r))^{2^i} \\
&= \sum_i \rho(\alpha_1^{2^i}, \ldots, \alpha_r^{2^i}) \\
&= \rho(\sum_i \alpha_1^{2^i}, \ldots, \sum_i \alpha_r^{2^i}) \\
&= \rho(\mathbf{Tr}_{E_1/\mathbb{Z}_2}(\alpha_1), \ldots, \mathbf{Tr}_{E_r/\mathbb{Z}_2}(\alpha_r)).
\end{aligned}$$

Now, suppose we choose $\alpha \in A$ at random. Then if $\alpha = \rho(\alpha_1, \ldots, \alpha_r)$, the α_i's will be independently distributed, with each α_i uniformly distributed over E_i. Since $\mathbf{Tr}_{E_i/\mathbb{Z}_2}$ is an F-linear map from E_i onto $\mathbb{Z}_2$, it follows that the values $c_i := \mathbf{Tr}_{E_i/\mathbb{Z}_2}(\alpha_i)$ will be independently and uniformly distributed over $\mathbb{Z}_2$. Thus, if $a = \operatorname{rep}(\mathcal{F}(\alpha))$, i.e., $a \in F[\mathtt{X}]$ is the polynomial of degree less than ℓ such that $\mathcal{F}(\alpha) = [a \bmod g]$, then $\gcd(a, g)$ will be the product of those factors g_i of g such that $c_i = 0$. We will fail to get a non-trivial

factorization only if the c_i's are either all 0 or all 1, which in the worst case, when $r = 2$, happens with probability $1/2$.

So our equal degree factorization algorithm in this case is a probabilistic, recursive algorithm that that takes as input a monic polynomial $g \in F[\mathtt{X}]$ of degree ℓ (we allow $\ell = 0$ to simplify the recursion), and an integer $k > 0$, such that g is the product of $r := \ell/k$ distinct monic irreducible polynomials, each of degree k, and runs as follows, where $A := F[\mathtt{X}]/(g)$ and $\mathcal{F} : A \to A$ is the map that sends α to $\sum_{i=0}^{wk-1} \alpha^{2^i}$:

Algorithm EDF:

If $r = 0$ then
    return
if $r = 1$ then
    output g, return

choose α at random from A
$d \leftarrow \gcd(\mathrm{rep}(\mathcal{F}(\alpha)), g)$
recursively factor g and g/d

The correctness of algorithm EDF follows from the above discussion. As for its expected running time, we can get a quick-and-dirty upper bound as follows:

- The expected number of trials until we get a non-trivial split is $O(1)$.
- Each trial costs $O(k\ell^2 \operatorname{len}(q))$ operations in F.
- The algorithm finishes after getting $r - 1$ non-trivial splits.
- Therefore, the total expected cost is $O(rk\ell^2 \operatorname{len}(q))$, or $O(\ell^3 \operatorname{len}(q))$, operations in F.

This analysis gives a bit of an over-estimate — it does not take into account the fact that we expect to get fairly "balanced" splits. The following analysis gives a better result:

Theorem 20.10 *In the case $p = 2$, algorithm EDF uses an expected number of $O(k\ell^2 \operatorname{len}(q) \operatorname{len}(r))$ operations in F.*

Proof. First, let us analyze the cost of a single invocation of the body of the recursive step. This is dominated by the cost of computing $\mathcal{F}(\alpha)$, which is $O(wk\ell^2)$, or $O(k\ell^2 \operatorname{len}(q))$, operations in F.

Second, let us analyze the expected value of the depth D of the recursion tree associated with the computation. Here, we measure D as the maximal depth of any internal node in the recursion tree (corresponding to recursive invocations where "real" work occurs), counting the root to be at depth 1. We claim that $\mathsf{E}[D] = O(\operatorname{len} r)$. To prove this claim, we use of the fact that

$$\mathsf{E}[D] = \sum_{t \geq 1} \mathsf{P}[D \geq t].$$

For any $t \geq 1$ and any distinct pair of indices (i, j), let $\mathcal{A}_{ij}^t$ be the event that the factors g_i and g_j have not been separated from each other after $t-1$ levels of recursion. Now, at any invocation of the body of the recursive step, if g_i and g_j have not been separated, then they will be with probability $1/2$. It follows that

$$\mathsf{P}[\mathcal{A}_{ij}^t] \leq 2^{-(t-1)}.$$

Also note that $D \geq t$ implies that for some (i, j), the event $\mathcal{A}_{ij}^t$ occurs. Hence, for $t \geq 1$, we have

$$\mathsf{P}[D \geq t] \leq \sum_{i,j} \mathsf{P}[\mathcal{A}_{ij}^t] \leq r^2 2^{-t}.$$

So we have

$$\begin{aligned}
\mathsf{E}[D] &= \sum_{t \geq 1} \mathsf{P}[D \geq t] \\
&= \sum_{t \leq 2\log_2 r} \mathsf{P}[D \geq t] + \sum_{t > 2\log_2 r} \mathsf{P}[D \geq t] \\
&\leq 2\log_2 r + \sum_{t > 2\log_2 r} r^2 2^{-t} \\
&\leq 2\log_2 r + \sum_{t \geq 0} 2^{-t} \\
&= 2\log_2 r + 2.
\end{aligned}$$

That proves the claim.

Third, consider any one level in the recursion tree, and suppose there are s internal nodes in the tree at this level, and that there are r_i irreducible factors at the ith node, for $1 \leq i \leq s$, so that. $\sum_{i=1}^s r_i \leq r$. The amount of work done at the ith node at this level is $O(r_i^2 k^3 \operatorname{len}(q))$ operations in F, and so the total amount of work done at this level is $O(\tau)$ operations in F, where

$$\tau = \sum_{i=1}^s r_i^2 k^3 \operatorname{len}(q) = k^3 \operatorname{len}(q) \sum_{i=1}^s r_i^2 \leq k^3 \operatorname{len}(q) (\sum_{i=1}^s r_i)^2 \leq k^3 \operatorname{len}(q) r^2 = k\ell^2 \operatorname{len}(q).$$

Putting this all together, since expected depth of the recursion tree is $O(\operatorname{len}(r))$, and the total amount of work done at any one level in the recursion tree is $O(k\ell^2 \operatorname{len}(q))$ operations in F, it follows that the expected number of operations in F performed by this algorithm is $O(k\ell^2 \operatorname{len}(q) \operatorname{len}(r))$. $\square$

Actually, the above running time estimate is *still* a bit of an over-estimate. The expected number of operations in F is really only $O(k\ell^2 \operatorname{len}(q))$. Intuitively, the reason is that at each recursive step, we expect to split g into two roughly equal pieces, and so heuristically speaking, we expect the running time to behave like $O(k^3 \operatorname{len}(q) C(r))$, where $C(r)$ satisfies the recurrence

$$C(r) = 2C(r/2) + O(r^2).$$

It is a standard fact from the analysis of "divide and conquer" algorithms that $C(r) = O(r^2)$, and so the total running time should be $O(k^3 \operatorname{len}(q) r^2)$, or $O(k\ell^2 \operatorname{len}(q))$, operations in F. The above argument is only heuristic, however, because the "divide and conquer" step is probabilistic, rather than deterministic, as the standard analysis of such algorithms assumes.

Exercise 20.11 Make the above heuristic argument rigorous, and prove that the expected number of operations in F performed by the above algorithm is in fact $O(k\ell^2 \operatorname{len}(q))$. □

Now assume that $p > 2$, so that p, and hence also q, is odd. Each group E_i^* is a cyclic group of order $q^k - 1$. Therefore, the image of the $(q^k - 1)/2$-power map on E_i^* is $\{\pm 1\}$. If we choose $\alpha_i \in E_i$ at random, then either $\alpha_i^{(q^k-1)/2} = 0$, which happens with probability $1/q^k$, or $\alpha_i^{(q^k-1)/2}$ is equally likely to be 1 or -1.

Consider the $(q^k - 1)/2$-power map on A. For $\alpha \in A$, if $\alpha = \rho(\alpha_1, \ldots, \alpha_r)$, we have

$$\alpha^{(q^k-1)/2} = \rho(\alpha_1^{(q^k-1)/2}, \ldots, \alpha_r^{(q^k-1)/2}).$$

Now, suppose we choose $\alpha \in A$ at random. Then if $\alpha = \rho(\alpha_1, \ldots, \alpha_r)$, the α_i's will be independently distributed, with each α_i uniformly distributed over E_i. Moreover, the values $c_i := \alpha_i^{(q^k-1)/2}$ will be independently distributed, with each c_i distributed as:

$$c_i = \begin{cases} 0 & \text{with probability } 1/q^k, \\ 1 & \text{with probability } (q^k - 1)/(2q^k), \\ -1 & \text{with probability } (q^k - 1)/(2q^k). \end{cases}$$

Thus, if $a = \operatorname{rep}(\alpha^{(q^k-1)/2} - 1)$ then $\gcd(a, g)$ will be the product of those factors g_i of g such that $c_i = 1$. We will fail to get a non-trivial factorization only if the c_i's are either all 1 or all not 1. Consider the worst case, namely, when $r = 2$. In this case, a simple calculation shows that the probability that we fail to split these two factors is

$$\left(\frac{q^k - 1}{2q^k}\right)^2 + \left(\frac{q^k + 1}{2q^k}\right)^2 = \frac{1}{2}(1 + 1/q^{2k}).$$

The (very) worst case is when $q^k = 3$, in which case the probability of failure is at most $5/9$.

So our equal degree factorization algorithm in the case is the same as algorithm EDF above, except that we define the function $\mathcal{F} : A \to A$ so that it sends $\alpha \in A$ to $\alpha^{(q^k-1)/2} - 1$.

The same quick-and-dirty analysis given just above Theorem 20.10 applies here as well, but just as before, we can do better:

Theorem 20.12 *In the case $p > 2$, algorithm EDF uses an expected number of $O(k\ell^2 \operatorname{len}(q) \operatorname{len}(r))$ operations in F.*

Proof. The analysis is essentially the same as in the case $p = 2$:

- The cost of a single recursive invocation is $O(k\ell^2 \operatorname{len}(q))$ operations in F.
- The expected value of the depth of the recursion is $O(\operatorname{len}(r))$. The analysis is the same as in the case $p = 2$, except now we use the bound $5/9$, instead of $1/2$, on the probability of failing to split a given pair of irreducible factors. This has the effect of increasing the expectation by a small constant factor (verify).
- The amount of work performed on any one level of the recursion tree is $O(k\ell^2 \operatorname{len}(q))$ operations in F.

□

Again,this estimate is actually somewhat pessimistic — the true value of the expectation is $O(k\ell^2 \operatorname{len}(q))$.

20.2.3 Analysis of the whole algorithm

Given an arbitrary polynomial $f \in F[\mathtt{X}]$ of degree $\ell > 0$, the distinct degree factorization step takes $O(\ell^3 \operatorname{len}(q))$ operations in F. This step produces a number of polynomials that must be subjected to equal degree factorization. If there are s such polynomials, where the ith polynomial has degree ℓ_i, for $1 \leq i \leq s$, then $\sum_{i=1}^{s} \ell_i = \ell$. Now, the equal degree factorization step for the ith polynomial takes an expected number of $O(\ell_i^3 \operatorname{len}(q))$ operations in F (actually, our "quick and dirty" estimates are good enough here), and so it follows that the total expected cost of all the equal degree factorization steps is $O(\sum_i \ell_i^3 \operatorname{len}(q))$, which is $O(\ell^3 \operatorname{len}(q))$, operations in F. Putting this all together, we conclude:

Theorem 20.13 *The Cantor-Zassenhaus factoring algorithm uses an expected number of $O(\ell^3 \operatorname{len}(q))$ operations in F.*

This bound is tight, since in the worst case, when the input is irreducible, the algorithm really does do this much work.

20.2.4 Deterministic factorization algorithms

The algorithm presented above for equal degree factorization is probabilistic. The following exercises develop a deterministic algorithm for this problem. This algorithm is only practical for finite fields of small characteristic, and is anyway mainly of theoretical interest, since from a practical perspective, there is nothing wrong with the above probabilistic method. In all of these exercises, F is a finite field of characteristic p and cardinality q, where $q = p^w$, and we assume that we have access to a basis $\epsilon_1, \ldots, \epsilon_w$ for F as a vector space over $\mathbb{Z}_p$.

Exercise 20.14 Let $g = g_1 \cdots g_r$, where the g_i's are distinct monic irreducible polynomials in $F[\mathtt{X}]$. Assume that $r > 0$, and let $\ell := \deg(g)$. For this exercise, the degrees of the g_i's need not be the same. For an intermediate field F', with $\mathbb{Z}_p \subset F' \subset F$, let us call a set $S = \{\lambda_1, \ldots, \lambda_s\}$ of polynomials in $F[\mathtt{X}]_{<\ell}$ a **separating set for g over F'** if the following conditions hold:

- for $1 \leq i \leq r$ and $1 \leq u \leq s$, there exists $c_{ui} \in F'$ such that $\lambda_u \equiv c_{ui} \pmod{g_i}$, and
- for any distinct pair of indices $1 \leq i < j \leq r$, there exists $1 \leq u \leq s$ such that $c_{ui} \neq c_{uj}$.

Show that if S is a $\mathbb{Z}_p$-separating set for g, then the following algorithm completely factors g using $O(p|S|\ell^2)$ operations in F.

```
C ← {g}
for each λ ∈ S do
    for each a ∈ Z_p do
        C' ← {}
        for each h ∈ C do
            d ← gcd(λ − a, h)
            if d = 1 then
                C' ← C ∪ {h}
            else
                C' ← C ∪ {d, h/d}
        C ← C'
output C
```

□

Exercise 20.15 Let g be as in the previous exercise. Show that if S is a separating set for g over F, then the set

$$S' := \{\sum_{i=0}^{w-1} (\epsilon_j \lambda)^{p^i} \text{ rem } g : 1 \leq j \leq w,\ \lambda \in S\}$$

is a separating set for g over $\mathbb{Z}_p$. Show how to compute this set using $O(|S|\ell^2 \operatorname{len}(p) w(w-1))$ operations in F. □

Exercise 20.16 Let g be as in the previous two exercises, but further suppose that each irreducible factor of g is of the same degree, say k. Let $A := F[\mathtt{X}]/(g)$ and $\eta := [\mathtt{X} \bmod g] \in A$. Define the polynomial $\phi \in A[\mathtt{Y}]$ as follows:

$$\phi := \prod_{i=0}^{k-1} (\mathtt{Y} - \eta^{q^i}).$$

If

$$\phi = \mathtt{Y}^k + \alpha_{k-1}\mathtt{Y}^{k-1} + \cdots + \alpha_0,$$

with $\alpha_0, \ldots, \alpha_{k-1} \in A$, show that the set

$$S := \{\text{rep}(\alpha_i) : 0 \leq i \leq k-1\}$$

is separating set for g over F, and can be computed deterministically using $O(k\,\mathrm{len}(q)+k^2)$ operations in A, and hence $O((k\,\mathrm{len}(q)+k^2)\ell^2)$ operations in F. $\square$

Exercise 20.17 Put together all of the above pieces, together with algorithm DDF, so as to obtain a deterministic algorithm for factoring polynomials over F that uses $(\ell+w+p)^{O(1)}$ operations in F, and make a careful estimate of the running time of your algorithm. $\square$

The following exercises show that the problem of factoring polynomials over F reduces in deterministic polynomial time to the problem of finding roots of polynomials over $\mathbb{Z}_p$.

Exercise 20.18 Now let F and g be as in Exercise 20.14. Suppose that $S = \{\lambda_1, \ldots, \lambda_s\}$ is a separating set for g over $\mathbb{Z}_p$, and $\phi_u \in F[\mathtt{X}]$ is the minimal polynomial over F of $[\lambda_u \bmod g] \in F[\mathtt{X}]/(g)$ for $1 \leq u \leq s$. Show that each ϕ_u is the product of linear factors over $\mathbb{Z}_p$, and that given S along with the roots of all the ϕ_u's, we can deterministically factor g using $(|S|+\ell)^{O(1)}$ operations in F. Hint: see Exercise 17.16. $\square$

Exercise 20.19 Using the previous exercise, show that the problem of factoring a polynomial over a finite field F reduces in deterministic polynomial time to the problem of finding roots of polynomials over the prime field of F. $\square$

20.3 Factoring Polynomials over Finite Fields: Berlekamp's Algorithm

We now develop an alternative algorithm, due to Berlekamp, for factoring a polynomial over the finite field F.

This algorithm usually starts with a pre-processing phase to reduce the problem to that of factoring square-free polynomials. There are a number of ways to carry out this step. We present a simple-minded method here that is sufficient for our purposes.

20.3.1 A simple square-free decomposition algorithm

Let $f \in F[\mathtt{X}]$ be a monic polynomial of degree $\ell > 0$. According to Exercise 19.13, if f is square-free, then $\gcd(f, \mathbf{D}(f)) = 1$; otherwise, either $\gcd(f, \mathbf{D}(f))$ is a non-trivial factor of f, or f is of the form $f = g(\mathtt{X}^p)$; in the latter case, if $g = \sum_i g_i \mathtt{X}^i$, then $f = h^p$, where $h = \sum_i g_i^{p^{(w-1)}} \mathtt{X}^i$.

This suggests the following recursive algorithm. The input is the polynomial f as above, and a parameter s, which is set to 1 on the initial invocation. The output is a list of pairs (g_i, s_i) such that each g_i is a square-free, non-constant polynomial over F and $f = \prod_i g_i^{s_i}$.

Algorithm SFD:

$d \leftarrow \gcd(f, \mathbf{D}(f))$
if $d = 1$ then
    output (f, s)
else if $d \neq f$ then
    recursively process (d, s) and $(f/d, s)$
else
    let $f = \mathtt{X}^\ell + \sum_{i=0}^{\ell-1} f_i \mathtt{X}^i$ — *note that* $f_i = 0$ *except when* $i \equiv 0 \pmod{p}$
    set $h \leftarrow \mathtt{X}^{\ell/p} + \sum_{i=0}^{\ell/p-1} (f_{pi})^{p^{w-1}} \mathtt{X}^i$ — *note that* $h = f^{1/p}$
    recursively process (h, ps)

The correctness of the above algorithm follows from the discussion above. As for the running time:

Theorem 20.20 *Algorithm SFD uses $O(\ell^3 + \ell(w-1)\operatorname{len}(p))$ operations in F.*

Proof. It is fairly easy to see that the total number of recursive invocations is $O(\ell)$ (verify). From this, it follows that the total cost contributed by the gcd computations is $O(\ell^3)$ operations in F. The only remaining cost to consider is that of computing the p^{w-1}th powers in F (if $w = 1$, of course, there is no cost). We claim that the total number of such powering steps is at most ℓ, and hence, if these are implemented using a repeated-squaring algorithm, the total cost of these steps is $O(\ell(w-1)\operatorname{len}(p))$. To prove this claim, let $C(f)$ be the maximum number of p^{w-1}th powering steps performed for an input polynomial f. We prove by induction on the recursion depth of the algorithm that $C(f) \leq \deg(f)$ for all f. Now, if f is square-free, then the algorithm halts immediately without performing any powering steps, and so $C(f) = 0 \leq \deg(f)$. Otherwise, if $d = \gcd(f, \mathbf{D}(f))$ is a proper divisor of f, the algorithm recursively processes d and f/d, and so by induction,

$$C(f) = C(d) + C(f/d) \leq \deg(d) + \deg(f/d) = \deg(f).$$

Otherwise, the algorithm performs $\deg(f)/p$ powering steps, and recursively processes a polynomial h of degree $\deg(f)/p$, and so by induction

$$C(f) = \deg(f)/p + C(h) \leq 2\deg(f)/p \leq \deg(f).$$

□

The term ℓ^3 in the running-time bound in Theorem 20.20 is essential. This cubic behavior is evoked, for example, on inputs that are powers of a single irreducible polynomial of constant degree.

Although it suffices for our immediate purpose as a pre-processing step in Berlekamp's factoring algorithm, algorithm SFD is by no means the most efficient algorithm possible

for square-free decomposition of polynomials. The following exercises develop a faster algorithm. To simplify matters, we first consider the problem over a field of characteristic zero.

Exercise 20.21 Let $f \in F[\mathtt{X}]$ be a monic polynomial over a field F of characteristic zero. Suppose that the factorization of f into irreducibles is

$$f = f_1^{e_1} \cdots f_r^{e_r}.$$

Show that

$$\frac{f}{\gcd(f, \mathbf{D}(f))} = f_1 \cdots f_r.$$

□

Exercise 20.22 Let F be a field of characteristic zero. Consider the following algorithm that takes as input a monic polynomial $f \in F[\mathtt{X}]$ of degree $\ell > 0$:

$j \leftarrow 1,\ \ g \leftarrow f/\gcd(f, \mathbf{D}(f))$
repeat
    $f \leftarrow f/g,\ \ h \leftarrow \gcd(f, g),\ \ m \leftarrow g/h$
    if $m \neq 1$ then output (m, j)
    $g \leftarrow h,\ \ j \leftarrow j + 1$
until $g = 1$

Using the result of the previous exercise, show that this algorithm outputs a list of pairs (g_i, s_i), such that each g_i is square-free, $f = \prod_i g_i^{s_i}$, and the g_i's are pair-wise co-prime. Furthermore, show that this algorithm uses $O(\ell^2)$ operations in F. □

Exercise 20.23 Let $f \in F[\mathtt{X}]$ be a monic polynomial over a field F of characteristic p. Suppose that the factorization of f into irreducibles is

$$f = f_1^{e_1} \cdots f_r^{e_r}.$$

Show that

$$\frac{f}{\gcd(f, \mathbf{D}(f))} = \prod_{\substack{1 \leq i \leq r \\ p \nmid e_i}} f_i.$$

□

Exercise 20.24 Let F be a finite field of characteristic p and cardinality $q = p^w$. Consider the following algorithm that takes as input a monic polynomial $f \in F[\mathtt{X}]$ of degree $\ell > 0$:

$s \leftarrow 1$
repeat
  $j \leftarrow 1,\ \ g \leftarrow f/\gcd(f, \mathbf{D}(f))$
  repeat
    $f \leftarrow f/g,\ \ h \leftarrow \gcd(f, g),\ \ m \leftarrow g/h$
    if $m \neq 1$ then output (m, js)
    $g \leftarrow h,\ \ j \leftarrow j + 1$
  until $g = 1$
  if $f \neq 1$ then
    — *f is a pth power*
    — *we compute a pth root as in algorithm SFD*
    $f \leftarrow f^{1/p},\ \ s \leftarrow ps$
until $f = 1$

Using the result of the previous exercise, show that this algorithm outputs a list of pairs (g_i, s_i), such that each g_i is square-free, $f = \prod_i g_i^{s_i}$, and the g_i's are pair-wise co-prime. Furthermore, show that this algorithm uses $O(\ell^2 + \ell(w-1)\operatorname{len}(p))$ operations in F. □

20.3.2 The main factoring algorithm

Let us now assume we have a monic square-free polynomial f of degree $\ell > 0$ that we want to factor into irreducibles, such as is output by the square-free decomposition algorithm above. We first present the mathematical ideas underpinning the algorithm.

Let A be the F-algebra $A := F[\mathtt{X}]/(f)$. We define a subset B of A as follows:

$$B := \{\alpha \in A : \alpha^q = \alpha\}.$$

It is easy to see that B is a subalgebra of A. Indeed, for $\alpha, \beta \in B$, we have $(\alpha + \beta)^q = \alpha^q + \beta^q = \alpha + \beta$, and similarly, $(\alpha\beta)^q = \alpha^q\beta^q = \alpha\beta$. One also sees that $1_A^q = 1_A$, as our definition of a subring requires. Finally, one sees that since $c^q = c$ for all $c \in F$, for any $\alpha \in B$, we have $(c\beta)^q = c^q\beta^q = c\beta$, and hence B is a subalgebra. The subalgebra B is called the **Berlekamp subalgebra of** A.

Let us take a closer look at the subalgebra B. To do this, suppose that the factorization of f into irreducibles is

$$f = f_1 \cdots f_r,$$

and let

$$\rho : E_1 \times \cdots \times E_r \to A$$

be the F-algebra isomorphism from the Chinese Remainder Theorem, where $E_i := F[\mathtt{X}]/(f_i)$ is an extension field of F of finite degree for $1 \leq i \leq r$. Now, for $\alpha = \rho(\alpha_1, \ldots, \alpha_r) \in A$, we have $\alpha^q = \alpha$ if and only if $\alpha_i^q = \alpha_i$ for $1 \leq i \leq r$; moreover, by

Theorem 19.6, we know that for any $\alpha_i \in E_i$, we have $\alpha_i^q = \alpha_i$ if and only if $\alpha_i \in F$. Thus, we may characterize B as follows:

$$B = \{\rho(c_1, \ldots, c_r) : c_1, \ldots, c_r \in F\}.$$

Since B is a subalgebra of A, then as F-vector spaces, B is a subspace of A. Of course, A has dimension ℓ over F, with the natural basis $1, \eta, \ldots, \eta^{\ell-1}$, where $\eta := [\mathtt{X} \bmod f]$. As for the Berlekamp subalgebra, from the above characterization of B, it is evident that

$$\rho(1,0,0,\ldots,0,0),\ \rho(0,1,0,\ldots,0,0),\ \ldots,\ \rho(0,0,0,\ldots,0,1)$$

is a basis for B over F, and hence, B has dimension r over F.

Now we come to the actual factoring algorithm.

Stage 1: Construct a basis for B

The first stage of Berlekamp's factoring algorithm constructs a basis for B over F. We can easily do this using Gaussian elimination, as follows. Let $\tau : A \to A$ be the map that sends $\alpha \in A$ to $\alpha^q - \alpha$. Since the qth power map on A is an F-algebra homomorphism (see Theorem 19.5) — and in particular, an F-linear map — the map τ is also F-linear. Moreover, the kernel of τ is none other than the Berlekamp subalgebra B. So to find a basis for B, we simply need to find a basis for the kernel of τ using Gaussian elimination, as in §15.4.

To perform the Gaussian elimination, we need to choose an ordered basis for A over F, and construct a matrix Q that represents τ with respect to that ordered basis as in §15.2, so that evaluation of τ corresponds to multiplying a row vector by Q on the right. We are free to choose an ordered basis in any convenient way, and the most convenient ordered basis, of course, is $(1, \eta, \ldots, \eta^{\ell-1})$, as this directly corresponds to the way we represent elements of A for computational purposes. Let $\epsilon : F^{1\times\ell} \to A$ be the F-vector space isomorphism that sends the coordinate vector $(a_0, \ldots, a_{\ell-1})$ to the corresponding element $\sum_i a_i\eta^i \in A$. The maps ϵ and ϵ^{-1} are best thought of as "type conversion operators" that require no actual computation to evaluate. The matrix Q, then, is the $\ell \times \ell$ matrix whose ith row, for $1 \le i \le \ell$, is $\epsilon^{-1}(\tau(\eta^{i-1}))$. Note that if $\alpha := \eta^q$, then

$$\tau(\eta^{i-1}) = (\eta^{i-1})^q - \eta^{i-1} = (\eta^q)^{i-1} - \eta^{i-1} = \alpha^{i-1} - \eta^{i-1}.$$

This observation allows us to construct the rows of Q by first computing α as η^q via repeated squaring, and then just computing successive powers of α.

After we construct the matrix Q, we apply Gaussian elimination to get row vectors $v_1, \ldots, v_r$ that form a basis for the row null space of Q. It is at this point that our algorithm actually discovers the number r of irreducible factors of f. We can then set $\beta_i := \epsilon(v_i)$ for $1 \le i \le r$ to get our basis for B.

Putting this altogether, we have the following algorithm to compute a basis for the Berlekamp subalgebra. The algorithm takes as input a monic square-free polynomial f

of degree $\ell > 0$, and runs as follows, where $A := F[\mathtt{X}]/(f)$, $\eta := [\mathtt{X} \bmod f] \in A$, and $\epsilon : F^{1\times\ell} \to A$ is the map that sends $(a_0, \ldots, a_{\ell-1})$ to $\sum_i a_i \eta^i$:

Algorithm B1:

let Q be an $\ell \times \ell$ matrix over F (initially with undefined entries)
compute $\alpha \leftarrow \eta^q$ using repeated squaring
$\beta \leftarrow 1_A$
for $i \leftarrow 1$ to ℓ do
    — *invariant:* $\beta = \alpha^{i-1} = (\eta^{i-1})^q$
    $Q(i) \leftarrow \epsilon^{-1}(\beta)$
    $Q(i,i) \leftarrow Q(i,i) - 1$
    $\beta \leftarrow \beta\alpha$

compute a basis $v_1, \ldots, v_r$ of the row null space of Q using Gaussian elimination
set $\beta_i \leftarrow \epsilon(v_i)$ for $i = 1, \ldots, r$
output $\beta_1, \ldots, \beta_r$

The correctness of algorithm B1 is clear from the above discussion. As for the running time:

Theorem 20.25 *Algorithm B1 uses $O(\ell^2 \operatorname{len}(q) + \ell^3)$ operations in F.*

Proof. This is just a matter of counting. The computation of α takes $O(\operatorname{len}(q))$ operations in A using repeated squaring, and hence $O(\ell^2 \operatorname{len}(q))$ operations in F. To build the matrix Q, we have to perform an additional $O(\ell)$ operations in A to compute the successive powers of α, which translates into $O(\ell^3)$ operations in F. Finally, the cost of Gaussian elimination is an additional $O(\ell^3)$ operations in F. □

Stage 2: Recursive splitting

The second stage of Berlekamp's factoring algorithm is a probabilistic, recursive algorithm that takes as input a monic square-free polynomial f and an auxiliary list $(\beta_1, \ldots, \beta_r)$ of elements which span the Berlekamp subalgebra B of $A := F[\mathtt{X}]/(f)$. This algorithm is initially invoked with the original input polynomial f to be factored, along with the basis constructed in Stage 1 above.

The algorithm chooses $c_1, \ldots, c_r \in F$ at random, and computes $\beta := \sum_i c_i \beta_i$. The element β will be uniformly distributed over B, and hence, if

$$\beta = \rho(b_1, \ldots, b_r),$$

then the b_i's will be uniformly and independently distributed over F. Analogous to algorithm EDF in §20.2.2, let us define a function $\mathcal{F} : A \to A$ as follows:

$$\mathcal{F}(\alpha) := \begin{cases} \sum_{i=0}^{w-1} \alpha^{2^i} & \text{if } p = 2 \\ \alpha^{(q-1)/2} - 1 & \text{if } p > 2 \end{cases} \tag{20.2}$$

With β as above, then just as in algorithm EDF, we have that $d := \gcd(\mathrm{rep}(\mathcal{F}(\beta)), f)$ will be a non-trivial factor of f with probability at least $1/2$, if $p = 2$, and probability at least $4/9$, if $p > 2$. If we succeed in splitting f in this way, then we proceed recursively, factoring $g_1 := d$ and $g_2 := f/d$. Note, however, that for the recursive step, we have to supply spanning sets for the Berlekamp subalgebras of $A_1 := F[\mathtt{X}]/(g_1)$ and $A_2 := F[\mathtt{X}]/(g_2)$. To do this, we simply reduce each of the given β_i's modulo g_1 and g_2. It is clear that each of these reduced lists form a spanning set for the corresponding Berlekamp subalgebra. To simplify notation, for $\alpha \in A$, and $g \mid f$, let us define $[\alpha \bmod g] := [\mathrm{rep}(\alpha) \bmod g] \in F[\mathtt{X}]/(g)$. In any recursive step, we can tell when we have an irreducible factor, since this happens if and only if the Berlekamp subalgebra coincides with F.

Our recursive splitting algorithm, then, takes as input a monic square-free polynomial f of degree ℓ (we allow $\ell = 0$ to simplify the recursion), along with an auxiliary list $(\beta_1, \ldots, \beta_r)$ of elements that span the Berlekamp subalgebra of $A := F[\mathtt{X}]/(f)$, and runs as follows, where the function $\mathcal{F} : A \to A$ is as defined in (20.2):

Algorithm B2:

if $\ell = 0$ return
if $\beta_1, \ldots, \beta_r \in F$ then
    — *f must be irreducible*
    output f
    return

choose $c_1, \ldots, c_r \in F$ at random
$\beta \leftarrow c_1\beta_1 + \cdots + c_r\beta_r$
$d \leftarrow \gcd(\mathrm{rep}(\mathcal{F}(\beta)), f)$
$g_1 \leftarrow d$, $g_2 \leftarrow f/d$
for $i = 1, 2$, recursively process g_i using the list $([\beta_1 \bmod g_i], \ldots, [\beta_r \bmod g_i])$

Note that in the above recursive specification, the quantity r refers to the number of factors of the *original* input polynomial f, which will not in general be the same as the number of irreducible factors of the factor of f being processed at a particular stage in the recursion.

The correctness of algorithm B2 follows from the above discussion. It is clear that algorithm B2 runs in expected polynomial time, since the expected number of trials until we get a non-trivial split is $O(1)$, the cost of each trial is polynomially bounded, and we are done after $r - 1$ non-trivial splits. A more careful analysis reveals:

Theorem 20.26 *Algorithm B2 uses an expected number of $O(r\ell^2 + \ell^2 \operatorname{len}(q) \operatorname{len}(r))$ operations in F.*

Proof. Let us break the cost (i.e., the number of operations in F) into two parts: the cost C_1 of computing the auxiliary list $([\beta_1 \bmod g_i], \ldots, [\beta_r \bmod g_i])$ in the cases where we

actually have a non-trivial split, and the cost C_2 comprising all other computations.

We claim that

$$C_1 = O(r\ell^2).$$

We leave the proof of this as an exercise (see below).

As for C_2, the analysis is essentially the same as that of algorithm EDF, and we obtain (verify)

$$\mathsf{E}[C_2] = O(\ell^2 \operatorname{len}(q) \operatorname{len}(r)).$$

□

Unlike in the case of algorithm EDF, the above running time estimate is tight, i.e., the factor of $\operatorname{len}(r)$ in the expected running time estimate really needs to be there. This worst-case behavior will be evoked, for example, when the input polynomial is the product of an irreducible factor of degree $\ell/2$, and $r-1$ linear factors — we expect that the large irreducible factor will appear at a depth of $\Omega(\operatorname{len}(r))$ in the recursion tree, and hence will cause an expected number of $\Omega(\ell^2 \operatorname{len}(q) \operatorname{len}(r))$ operations in F to be performed.

Exercise 20.27 Prove the claim made in the proof of Theorem 20.26 that $C_1 = O(r\ell^2)$. □

20.3.3 Analysis of the whole algorithm

Putting together algorithm SFD with algorithms B1 and B2, we get Berlekamp's complete factoring algorithm. The running time bound is easily estimated from the results already proved:

Theorem 20.28 *Berlekamp's factoring algorithm uses an expected number of* $O(\ell^3 + \ell^2 \operatorname{len}(\ell) \operatorname{len}(q))$ *operations in* F.

So we see that Berlekamp's algorithm is in fact faster than the Cantor-Zassenhaus algorithm, whose expected operation count is $O(\ell^3 \operatorname{len}(q))$. The speed advantage of Berlekamp's algorithm grows as q gets large. The one disadvantage of Berlekamp's algorithm is space: it requires space for $\Theta(\ell^2)$ elements of F, while the Cantor-Zassenhaus algorithm requires space for only $O(\ell)$ elements of F.

Exercise 20.29 Using the ideas behind Berlekamp's factoring algorithm, devise a deterministic irreducibility test that given monic polynomial of degree ℓ over a finite field F of cardinality q uses $O(\ell^2 \operatorname{len}(q) + \ell^3)$ operations in F. □

Exercise 20.30 Let $\beta_1, \ldots, \beta_r$ be a basis for the Berlekamp subalgebra of $A := F[\mathtt{X}]/(f)$. Show that the set $S := \{\operatorname{rep}(\beta_1), \ldots, \operatorname{rep}(\beta_r)\}$ is a separating set for f over F (see Exercise 20.14). Use this fact to design a deterministic factoring algorithm based on Berlekamp's method that uses $(\ell + w + p)^{O(1)}$ operations in F, and make a careful estimate of the running time of your algorithm. □

20.4 Notes

In this section, we use the notation "$O\tilde{}(f)$," pronounced "soft-Oh of f," to denote a function that is $O(f(\log(2+|f|))^c)$ for some constant c. For example, with this notation, we can simply say that multiplication, division, and greatest common divisors of degree ℓ polynomials can be computed using $O\tilde{}(\ell)$ operations in F. This notation is useful for simplifying messy expressions involving powers of $\mathrm{len}(\ell)$ and $\mathrm{len}(\mathrm{len}(\ell))$. Of course, from a practical point of view, such "soft-Oh" estimates must be viewed with a certain amount of skepticism.

The average-case analysis of algorithm IPT, assuming its input is random, and the application to the analysis of algorithm RIP, is due to Ben-Or [12]. If one implements algorithm RIP using fast polynomial arithmetic, one gets an expected cost of $O\tilde{}(\ell^2 \,\mathrm{len}(q))$ operations in F. Note that Ben-Or's analysis is a bit incomplete — see Exercise 32 in Chapter 7 of Bach and Shallit [11] for a complete analysis of Ben-Or's claims.

The asymptotically fastest probabilistic algorithm for constructing an irreducible polynomial over F of degree ℓ is due to Shoup [69]. That algorithm uses an expected number of $O\tilde{}(\ell^2 + \ell \,\mathrm{len}(q))$ operations in F, and in fact does not follow the "generate and test" paradigm of algorithm RIP, but uses a completely different approach. As far as *deterministic* algorithms for constructing irreducible polynomials of given degree over F, the only efficient methods known are when the characteristic p of F is small (see Chistov [20], Semaev [64], and Shoup [68]).

The algorithm in Example 20.3 for computing minimal polynomials over finite fields is due to Gordon [30].

The Cantor-Zassenhaus algorithm was initially developed by Cantor and Zassenhaus [18], although many of the basic ideas can be traced back quite a ways. A straightforward implementation of this algorithm using fast polynomial arithmetic uses an expected number of $O\tilde{}(\ell^2 \,\mathrm{len}(q))$ operations in F.

Berlekamp's algorithm was initially developed by Berlekamp [13, 14], but again, many of the basic idea go back a long way. A straightforward implementation using fast polynomial arithmetic uses an expected number of $O\tilde{}(\ell^3 + \ell \,\mathrm{len}(q))$ operations in F, which may be reduced to of $O\tilde{}(\ell^\omega + \ell \,\mathrm{len}(q))$, where ω is the exponent of matrix multiplication.

The square-free decomposition of a polynomial over a field F of characteristic zero can be obtained using an algorithm of Yun [79] using $O\tilde{}(\ell)$ operations in F. For finite fields F of cardinality p^w, one can adapt Yun's algorithm so that it uses $O\tilde{}(\ell + \ell(w-1)\,\mathrm{len}(p))$ operations in F (c.f., Exercise 14.30 in von zur Gathen and Gerhard [73]).

The asymptotically fastest algorithms for factoring polynomials over a finite field F are due to von zur Gathen, Kaltofen, and Shoup: the algorithm of von zur Gathen and Shoup [74] uses an expected number of $O\tilde{}(\ell^2 + \ell \,\mathrm{len}(q))$ operations in F; the algorithm of Kaltofen and Shoup [38] has a cost that is sub-quadratic in the degree — it uses an expected number of $O(\ell^{1.815} \,\mathrm{len}(q)^{0.407})$ operations in F. Although the "fast" algorithms in [74] and [38] are mainly of theoretical interest, a variant in [38], which uses $O\tilde{}(\ell^{2.5} + \ell \,\mathrm{len}(q))$ operations in F, and space for $O(\ell^{1.5})$ elements of F, has proven to be quite practical (see Shoup [70]).

Chapter 21

Deterministic Primality Testing

Until very recently, there was no known deterministic, polynomial time algorithm for testing whether a given integer $n > 1$ is a prime. However, that is no longer the case — the breakthrough algorithm of Agrawal, Kayal, and Saxena, or AKS algorithm for short, is just such an algorithm. Not only is the result itself wonderful, but the algorithm is striking in both its simplicity, and in the fact that the proof of its running time and correctness are completely elementary (though ingenious).

We should stress at the outset that although this result is an important theoretical result, as of yet, it has no real practical significance: probabilistic tests, such as the Miller-Rabin test discussed in §10, are *much* more efficient, and the suitably practical minded person is not at all bothered by the fact that such algorithms may in theory make a mistake with an incredibly small probability.

21.1 The Basic Idea

The algorithm is based on the following fact:

Theorem 21.1 *Let $n > 1$ be an integer and $a \in \mathbb{Z}_n^*$. Then n is prime if and only if in the ring $\mathbb{Z}_n[\mathtt{X}]$*

$$(\mathtt{X} + a)^n = \mathtt{X}^n + a. \tag{21.1}$$

Proof. Note that

$$(\mathtt{X} + a)^n = \mathtt{X}^n + a^n + \sum_{i=1}^{n-1} \binom{n}{i} a^i \mathtt{X}^{n-i}.$$

If n is prime, then by Theorem 8.72 (Fermat's Little Theorem), we have $a^n = a$, and by Exercise 1.16, all of the binomial coefficients $\binom{n}{i}$, for $1 \leq i \leq n-1$, are divisible by n, and hence their images in the ring $\mathbb{Z}_n$ vanish. That proves that the identity (21.1) holds when n is prime.

Conversely, suppose that n is composite. Consider any prime factor p of n, and suppose $n = p^k m$, where $p \nmid m$.

We claim that $p^k \nmid \binom{n}{p}$. To prove the claim, one simply observes that

$$\binom{n}{p} = \frac{n(n-1)\cdots(n-p+1)}{p!},$$

and the numerator of this fraction is an integer divisible by p^k, but no higher power of p, and the denominator is divisible by p, but no higher power of p. That proves the claim.

From the claim, and the fact that $a \in \mathbb{Z}_n^*$, it follows that the coefficient of $\mathtt{X}^{n-p}$ in $(\mathtt{X}+a)^n$ is not zero, and hence the identity (21.1) does not hold. □

Of course, Theorem 21.1 does not immediately give rise to an efficient primality test, since just evaluating the left-hand side of the identity (21.1) takes time $\Omega(n)$ in the worst case. The key observation of Agrawal, Kayal, and Saxena is that if (21.1) holds modulo $\mathtt{X}^r - 1$ for a suitably chosen value of r, and for sufficiently many a, then n must be prime. To make this idea work, one must show that a suitable r exists that is bounded by a polynomial in $\mathrm{len}(n)$, and that the number of different values of a that must be tested is also bounded by a polynomial in $\mathrm{len}(n)$.

21.2 The Algorithm and its Analysis

Here is the primality test. It takes as input an integer $n > 1$.

Algorithm AKS:

1. if n is of the form a^b for integers $a > 1$ and $b > 1$ then
   return *false*
2. find the smallest integer $r > 1$ such that either
   $\gcd(n, r) > 1$
   or
   $\gcd(n, r) = 1$ and $[n \bmod r] \in \mathbb{Z}_r^*$ has order greater than $4\,\mathrm{len}(n)^2$
3. if $r = n$ then
   return *true*
4. if $\gcd(n, r) > 1$ then
   return *false*
5. for $j \leftarrow 1$ to $2\,\mathrm{len}(n)\lfloor r^{1/2} \rfloor + 1$ do
   if $(\mathtt{X}+j)^n \not\equiv \mathtt{X}^n + j \pmod{\mathtt{X}^r - 1}$ in the ring $\mathbb{Z}_n[\mathtt{X}]$ then
   return *false*
6. return *true*

A few remarks on implementation are in order:

- In step (1), we can use the algorithm for perfect-power testing discussed in §10.5, which is a deterministic, polynomial-time algorithm.

- The search for r in step (2) can just be done by brute-force search; likewise, the determination of the order of $[n \bmod r] \in \mathbb{Z}_r^*$ can be done by brute force — after verifying that $\gcd(n, r) = 1$, compute successive powers of n modulo r until we get 1.

We want to prove that algorithm AKS runs in polynomial time and is correct. To prove that it runs in polynomial time, it clearly suffices to prove that there exists an integer r satisfying the condition in step (2) that is bounded by a polynomial in $\mathrm{len}(n)$, since all other computations can be carried out in time $(r + \mathrm{len}(n))^{O(1)}$. Correctness means that if it outputs *true* if and only if n is prime.

The question of running time of algorithm AKS is settled by the following fact:

Theorem 21.2 *For integers $n > 1$ and $m \geq 1$, the least prime r such that $r \nmid n$ and the order of $[n \bmod r] \in \mathbb{Z}_r^*$ is greater than m is $O(m^2 \,\mathrm{len}(n))$.*

Proof. Call a prime r "good" if $r \nmid n$ and the order of $[n \bmod r] \in \mathbb{Z}_r^*$ is greater than m, and otherwise call r "bad." If r is bad, then either $r \mid n$ or $r \mid (n^d - 1)$ for some $1 \leq d \leq m$. Thus, any bad prime r satisfies

$$r \mid n \prod_{d=1}^{m} (n^d - 1).$$

If all primes r up to some given bound $x \geq 2$ are bad, then the product of all primes up to x divides $n \prod_{d=1}^{m} (n^d - 1)$, and so in particular,

$$\prod_{r \leq x} r \leq n \prod_{d=1}^{m} (n^d - 1),$$

where the product is over all primes r up to x. Taking logarithms, we obtain

$$\sum_{r \leq x} \log r \leq \log(n \prod_{d=1}^{m} (n^d - 1)) \leq (\log n)(1 + \sum_{d=1}^{m} d) = (\log n)(1 + m(m+1)/2).$$

But by Theorem 5.7, we have

$$\sum_{r \leq x} \log r \geq cx$$

for some constant $c > 0$, from which it follows that

$$x \leq c^{-1} (\log n)(1 + m(m+1)/2),$$

and the theorem follows. □

From this theorem, it follows that the value of r found in step (2) — which need not be prime — will be $O(\mathrm{len}(n)^5)$. From this, we obtain:

Theorem 21.3 *Algorithm AKS can be implemented so as to run in time $O(\mathrm{len}(n)^{16.5})$.*

Proof. As discussed above, the value of r determined in step (2) will be $O(\mathrm{len}(n)^5)$. It is fairly straightforward to see that the running time of the algorithm is dominated by the running time of step (5). Here, we have to perform $O(r^{1/2}\,\mathrm{len}(n))$ exponentiations to the power n in the ring $\mathbb{Z}_n[\mathtt{X}]/(\mathtt{X}^r - 1)$. Each of these exponentiations takes $O(\mathrm{len}(n))$ operations in $\mathbb{Z}_n[\mathtt{X}]/(\mathtt{X}^r - 1)$, each of which takes $O(r^2)$ operations in $\mathbb{Z}_n$, each of which takes time $O(\mathrm{len}(n)^2)$. This yields a running time bounded by a constant times

$$r^{1/2}\,\mathrm{len}(n) \times \mathrm{len}(n) \times r^2 \times \mathrm{len}(n)^2 = r^{2.5}\,\mathrm{len}(n)^4.$$

Substituting the bound $O(\mathrm{len}(n)^5)$ for r, we obtain the stated bound in the theorem. □

As for the correctness of algorithm AKS, we first show:

Theorem 21.4 *If the input to algorithm AKS is prime, then the output is* true.

Proof. Assume that the input n is prime. The test is step (1) will certainly fail. If the algorithm does not return *true* in step (3), then certainly the test in step (4) will certainly fail as well. If the algorithm reaches step (5), then all of the tests in the loop in step (5) will fail — this follows from Theorem 21.1. Note that for *very* small values of n, we could have $j \equiv 0 \pmod{n}$ for some values of the loop index j, and strictly speaking, Theorem 21.1 only applies for $a \in \mathbb{Z}_n^*$; however, it is clear that for prime n, the identity (21.1) holds for all $a \in \mathbb{Z}_n$. □

The interesting case is the following:

Theorem 21.5 *If the input to algorithm AKS is composite, then the output is* false.

The proof of this theorem is rather long, and is the subject of the remainder of this section.

Suppose the input n is composite. If n is a prime power, then this will be detected in step (1), so we may assume that n is not a prime power. Assume that the algorithm has found a suitable value of r in step (2). Clearly, the test in (3) will fail. If the test in step (4) passes, we are done, so we may assume that this test fails, i.e., that all prime factors of n are greater than r. Our goal now is to show that one of the tests in the loop in step (5) must pass. The proof will be by contradiction: we shall assume that none of the tests pass, and derive a contradiction.

The assumption that none of the tests in step (5) fail means that in the ring $\mathbb{Z}_n[\mathtt{X}]$, the following congruences hold:

$$(\mathtt{X} + j)^n = \mathtt{X}^n + j \pmod{\mathtt{X}^r - 1} \quad (j = 1, \ldots, 2\,\mathrm{len}(n)\lfloor r^{1/2} \rfloor + 1). \tag{21.2}$$

For the rest of the proof, we fix any particular prime divisor p of n — the choice does not matter. Since $p \mid n$, we have a natural homomorphism from $\mathbb{Z}_n[\mathtt{X}]$ to $\mathbb{Z}_p[\mathtt{X}]$ (see Example 9.68), which implies that the congruences (21.2) hold in the ring of polynomials

over $\mathbb{Z}_p$ as well. *From now on, we shall work exclusively with polynomials over* $\mathbb{Z}_p$. Moreover, let us state in somewhat more abstract terms the precise assumptions we are making in order to derive our contradiction — the rest of the proof will rely only on these assumptions, and not on any other details of algorithm AKS.

A0. $n > 1$, $r > 1$, and $\ell \geq 1$ are integers, p is a prime dividing n, and $\gcd(n, r) = 1$.

A1. n is not a prime power.

A2. $p > r$.

A3. The congruences

$$(\mathtt{X} + j)^n = \mathtt{X}^n + j \pmod{\mathtt{X}^r - 1} \quad (j = 1, \ldots, \ell)$$

hold in the ring $\mathbb{Z}_p[\mathtt{X}]$.

A4. The order of $[n \bmod r] \in \mathbb{Z}_r^*$ is greater than $4\,\mathrm{len}(n)^2$.

A5. $\ell > 2\,\mathrm{len}(n)\lfloor r^{1/2} \rfloor$.

From now on, only assumption (A0) will be implicitly in force. The other assumptions will be explicitly invoked as necessary. Our goal is to show that assumptions (A1), (A2), (A3), (A4), and (A5) cannot all be true simultaneously.

Define the $\mathbb{Z}_p$-algebra $A := \mathbb{Z}_p[\mathtt{X}]/(\mathtt{X}^r - 1)$, and let $\eta := [\mathtt{X} \bmod (\mathtt{X}^r - 1)] \in A$, so that $A = \mathbb{Z}_p[\eta]$. Every element of A can be expressed uniquely as $g(\eta) = [g \bmod (\mathtt{X}^r - 1)]$, for $g \in \mathbb{Z}_p[\mathtt{X}]$ of degree less than r, and for an arbitrary polynomial $g \in \mathbb{Z}_p[\mathtt{X}]$, we have $g(\eta) = 0$ if and only if $(\mathtt{X}^r - 1) \mid g$. Note that $\eta \in A^*$ and has multiplicative order r: indeed, $\eta^r = 1$, and $\eta^s - 1$ cannot be zero for $s < r$, since $\mathtt{X}^s - 1$ has degree less than r.

Assumption (A3) implies that we have a number of interesting identities in the $\mathbb{Z}_p$-algebra A:

$$(\eta + j)^n = \eta^n + j \quad (j = 1, \ldots, \ell).$$

For the polynomials $g = \mathtt{X} + j \in \mathbb{Z}_p[\mathtt{X}]$, with j in the given range, these identities say that $g(\eta)^n = g(\eta^n)$. In order to exploit these identities, we study more generally functions that send $g(\eta) \in A$ to $g(\eta^k)$, for general $g(\eta) \in A$ and general k, and we investigate the implications of the assumption that such functions behave like the kth power map on certain values of $g(\eta)$ and k.

Let $\mathbb{Z}^{(r)}$ denote the set of all positive integers k such that $\gcd(r, k) = 1$. Note that the set $\mathbb{Z}^{(r)}$ is multiplicative, i.e., $1 \in \mathbb{Z}^{(r)}$, and for all $k, k' \in \mathbb{Z}^{(r)}$, we have $kk' \in \mathbb{Z}^{(r)}$. Also note that because of our assumption (A0), both n and p are in $\mathbb{Z}^{(r)}$.

For integer $k \in \mathbb{Z}^{(r)}$, let $\hat{\sigma}_k : \mathbb{Z}_p[\mathtt{X}] \to A$ be the polynomial evaluation map that sends $g \in \mathbb{Z}_p[\mathtt{X}]$ to $g(\eta^k)$. This is of course a $\mathbb{Z}_p$-algebra homomorphism, and we have:

Lemma 21.6 *For all $k \in \mathbb{Z}^{(r)}$, the kernel of $\hat{\sigma}_k$ is $(\mathtt{X}^r - 1)$, and the image of $\hat{\sigma}_k$ is A.*

Proof. Let $J := \ker(\hat{\sigma}_k)$, which is an ideal in A, of course. Let k' be a positive integer such that $kk' \equiv 1 \pmod{r}$, which exists because $\gcd(r, k) = 1$.

First, we show that $\mathtt{X}^r - 1 \in J$. To see this, simply observe that

$$\hat{\sigma}_k(\mathtt{X}^r - 1) = (\eta^k)^r - 1 = (\eta^r)^k - 1 = 1^k - 1 = 0.$$

Second, we show that $J \subset (\mathtt{X}^r - 1)$. Let $g \in J$. We want to show that $(\mathtt{X}^r - 1) \mid g$. Now, $g \in J$ means that $g(\eta^k) = 0$. If we set $h := g(\mathtt{X}^k)$, this implies that $h(\eta) = 0$, which means that $(\mathtt{X}^r - 1) \mid h$. So let us write $h = (\mathtt{X}^r - 1)f$, for some $f \in \mathbb{Z}_p[\mathtt{X}]$. Then

$$g(\eta) = g(\eta^{kk'}) = h(\eta^{k'}) = (\eta^{k'r} - 1)f(\eta^{k'}) = 0,$$

which implies that $(\mathtt{X}^r - 1) \mid g$.

Finally, to show that $\hat{\sigma}_k$ is surjective, suppose we are given an arbitrary element of A, which we can express as $g(\eta)$ for some $g \in \mathbb{Z}_p[\mathtt{X}]$. Now set $h := g(\mathtt{X}^{k'})$, and observe that

$$\hat{\sigma}_k(h(\eta)) = h(\eta^k) = g(\eta^{kk'}) = g(\eta).$$

□

Because of Lemma 21.6, then by Theorem 9.60, the map $\sigma_k : A \to A$ that sends $g(\eta) \in A$ to $g(\eta^k)$, for $g \in \mathbb{Z}_p[\mathtt{X}]$, is well defined, and is a ring automorphism — indeed, a $\mathbb{Z}_p$-*algebra* automorphism — on A. Note that for any $k, k' \in \mathbb{Z}^{(r)}$, we have

- $\sigma_k = \sigma_{k'}$ if and only if $\eta^k = \eta^{k'}$ if and only if $k \equiv k' \pmod{r}$, and
- $\sigma_k \circ \sigma_{k'} = \sigma_{k'} \circ \sigma_k = \sigma_{kk'}$.

So in fact, the set of all σ_k form an abelian group (with respect to composition) that is isomorphic to $\mathbb{Z}_r^*$.

It is perhaps helpful (but not necessary for the proof) to examine the behavior of the map σ_k in a bit more detail. Let $\alpha \in A$, and let

$$\alpha = \sum_{i=0}^{r-1} g_i \eta^i$$

be the canonical representation of α. Since $\gcd(r, k) = 1$, the map that $\pi : \{0, \ldots, r-1\} \to \{0, \ldots, r-1\}$ that sends i to $ki \bmod r$ is a permutation whose inverse is the permutation π' that sends i to $k'i \bmod r$, where k' is a multiplicative inverse of k modulo r. Then we have

$$\sigma_k(\alpha) = \sum_{i=0}^{r-1} g_i \eta^{ki} = \sum_{i=0}^{r-1} g_i \eta^{\pi(i)} = \sum_{i=0}^{r-1} g_{\pi'(i)} \eta^i.$$

Thus, the action of σ_k is to permute the coordinate vector $(g_0, \ldots, g_{r-1})$ of α, sending α to the element in A whose coordinate vector is $(g_{\pi'(0)}, \ldots, g_{\pi'(r-1)})$. So we see that although we defined the maps σ_k in a rather "high brow" algebraic fashion, their behavior in concrete terms is actually quite simple.

Recall that the pth power map on A is a $\mathbb{Z}_p$-algebra homomorphism (see Theorem 19.5 or Example 19.5), and so for all $\alpha \in A$, if $\alpha = g(\eta)$ for $g \in \mathbb{Z}_p[\mathtt{X}]$, then (by Theorem 14.50) we have

$$\alpha^p = g(\eta)^p = g(\eta^p) = \sigma_p(\alpha).$$

Thus, σ_p acts just like the pth power map on all elements of A.

We can restate assumption (A3) as follows:

$$(\eta + j)^n = \sigma_n(\eta + j) \quad (j = 1, \ldots, \ell).$$

That is to say, the map σ_n acts just like the nth power map on the elements $\eta + j$ for $1 \leq j \leq \ell$.

Now, although the σ_p map must act like the pth power map on all of A, there is no good reason why the σ_n map should act like the nth power map on any particular element of A, and so the fact that it does so on all the elements $\eta + j$ for $1 \leq j \leq \ell$ looks decidedly suspicious. To turn our suspicions into a contradiction, let us start by defining some notation. For $\alpha \in A$, let us define

$$C(\alpha) := \{k \in \mathbb{Z}^{(r)} : \sigma_k(\alpha) = \alpha^k\},$$

and for $k \in \mathbb{Z}^{(r)}$, let us define

$$D(k) := \{\alpha \in A : \sigma_k(\alpha) = \alpha^k\}.$$

In words: $C(\alpha)$ is the set of all k for which σ_k acts like the kth power map on α, and $D(k)$ is the set of all α for which σ_k acts like the kth power map on α. From the discussion above, we have $p \in C(\alpha)$ for all $\alpha \in A$, and it is also clear that $1 \in C(\alpha)$ for all $\alpha \in A$. Also, it is clear that $\alpha \in D(p)$ for all $\alpha \in A$, and $1_A \in D(k)$ for all $k \in \mathbb{Z}^{(r)}$.

The following two simple lemmas say that the sets $C(\alpha)$ and $D(k)$ are multiplicative.

Lemma 21.7 *For any $\alpha \in A$, if $k \in C(\alpha)$ and $k' \in C(\alpha)$, then $kk' \in C(\alpha)$.*

Proof. If $\sigma_k(\alpha) = \alpha^k$ and $\sigma_{k'}(\alpha) = \alpha^{k'}$, then

$$\sigma_{kk'}(\alpha) = \sigma_k(\sigma_{k'}(\alpha)) = \sigma_k(\alpha^{k'}) = (\sigma_k(\alpha))^{k'} = (\alpha^k)^{k'} = \alpha^{kk'},$$

where we have made use of the homomorphic property of σ_k. □

Lemma 21.8 *For any $k \in \mathbb{Z}^{(r)}$, if $\alpha \in D(k)$ and $\beta \in D(k)$, then $\alpha\beta \in D(k)$.*

Proof. If $\sigma_k(\alpha) = \alpha^k$ and $\sigma_k(\beta) = \beta^k$, then

$$\sigma_k(\alpha\beta) = \sigma_k(\alpha)\sigma_k(\beta) = \alpha^k\beta^k = (\alpha\beta)^k,$$

where again, we have made use of the homomorphic property of σ_k. □

Let us define

- s to be the order of $[p \bmod r] \in \mathbb{Z}_r^*$, and
- t to be the order of the subgroup of $\mathbb{Z}_r^*$ generated by $[p \bmod r]$ and $[n \bmod r]$.

Since $r \mid (p^s - 1)$, if we take any extension field E of degree s over $\mathbb{Z}_p$ (which we know exists by Theorem 19.10), then since E^* is cyclic (Theorem 10.2) and has order $p^s - 1$, we know that there exists an element $\zeta \in E^*$ of order r (Theorem 8.75). Let us define the polynomial evaluation map $\hat{\tau} : \mathbb{Z}_p[\mathtt{X}] \to E$ that sends $g \in \mathbb{Z}_p[\mathtt{X}]$ to $g(\zeta) \in E$. Since $\mathtt{X}^r - 1$ is clearly in the kernel of $\hat{\tau}$, then by Theorem 9.61, the map $\tau : A \to E$ that sends $g(\eta)$ to $g(\zeta)$, for $g \in \mathbb{Z}_p[\mathtt{X}]$, is a well-defined ring homomorphism, and actually, it is a $\mathbb{Z}_p$-algebra homomorphism.

Note: it is not necessary, but if we wish, we can take E to be $\mathbb{Z}_p[\mathtt{X}]/(\phi)$, where ϕ is an irreducible factor of $\mathtt{X}^r - 1$ of degree s, and ζ to be $[\mathtt{X} \bmod \phi]$ (see Example 19.24), in which case the map $\hat{\tau}$ above is just the natural map from $\mathbb{Z}_p[\mathtt{X}]$ to $\mathbb{Z}_p[\mathtt{X}]/(\phi)$.

The key to deriving our contradiction is to examine the set $S := \tau(D(n))$, that is, the image under τ of the set $D(n)$ of all elements $\alpha \in A$ for which σ_n acts like the nth power map.

Lemma 21.9 *Under assumption (A1), we have*

$$|S| \leq n^{2\lfloor t^{1/2} \rfloor}.$$

Proof. Consider the set of integers

$$I := \{n^u p^v : 0 \leq u, v \leq \lfloor t^{1/2} \rfloor\}.$$

We first claim that $|I| > t$. To prove this, we first show that each distinct pair (u, v) gives rise to a distinct value $n^u p^v$. To this end, we make use of our assumption (A1) that n not a prime power, and so is divisible by some prime q other than p. Thus, if $(u', v') \neq (u, v)$, then either

- $u \neq u'$, in which case the power of q in the prime factorization of $n^u p^v$ is different from that in $n^{u'} p^{v'}$, or
- $u = u'$ and $v \neq v'$, in which case the power of p in the prime factorization of $n^u p^v$ is different from that in $n^{u'} p^{v'}$.

The claim now follows from the fact that both u and v range over a set of size $\lfloor t^{1/2} \rfloor + 1 > t^{1/2}$, and so there are strictly greater than t such pairs (u, v).

Next, recall that t was defined to be the order of the subgroup of $\mathbb{Z}_r^*$ generated by $[n \bmod r]$ and $[p \bmod r]$; that is, t is the number of distinct residue classes of the form $[n^u p^v \bmod r]$, where u and v range over all non-negative integers. Since each element of I is of the form $n^u p^v$, and $|I| > t$, we may conclude that there must be two distinct elements of I, call them k and k', that are congruent modulo r. Furthermore, any element of I is a product of two positive integers each of which is at most $n^{\lfloor t^{1/2} \rfloor}$, and so we have $1 \leq k, k' \leq n^{2\lfloor n^{1/2} \rfloor}$.

Now, let $\alpha \in D(n)$. This is equivalent to saying $n \in C(\alpha)$. We always have $1 \in C(\alpha)$ and $p \in C(\alpha)$, and so by Lemma 21.7, we have $n^u p^v \in C(\alpha)$ for all non-negative integers u, v, and so in particular, $k, k' \in C(\alpha)$.

Since both k and k' are in $C(\alpha)$, we have

$$\sigma_k(\alpha) = \alpha^k \quad \text{and} \quad \sigma_{k'}(\alpha) = \alpha^{k'}.$$

Since $k \equiv k' \pmod{r}$, we have $\sigma_k = \sigma_{k'}$, and hence

$$\alpha^k = \alpha^{k'}.$$

Now apply the homomorphism τ, obtaining

$$\tau(\alpha)^k = \tau(\alpha)^{k'}.$$

Since this holds for all $\alpha \in D(n)$, we conclude that all elements of S are roots of the polynomial $\mathtt{X}^k - \mathtt{X}^{k'}$. Since $k \neq k'$, we see that $\mathtt{X}^k - \mathtt{X}^{k'}$ is a non-zero polynomial of degree at most $\max\{k, k'\} \leq n^{2\lfloor t^{1/2} \rfloor}$, and hence can have at most $n^{2\lfloor t^{1/2} \rfloor}$ roots in the field E (Theorem 9.41). □

Lemma 21.10 *Under assumptions (A2) and (A3), we have*

$$|S| \geq 2^{\min(t,\ell)} - 1.$$

Proof. Let $m := \min(t, \ell)$. Under assumption (A3), we have $\eta + j \in D(n)$ for $j = 1, \ldots, m$. Under assumption (A2), we have $p > r > t \geq m$, and hence the integers $j = 1, \ldots, m$ are distinct modulo p. Define

$$P := \{\prod_{j=1}^{m} (\mathtt{X} + j)^{e_j} \in \mathbb{Z}_p[\mathtt{X}] : e_j \in \{0, 1\} \text{ for } j = 1, \ldots, m, \text{ and } \sum_{j=1}^{m} e_j < m\}.$$

That is, we form P by taking products over all subsets $S \subsetneq \{\mathtt{X} + j : j = 1, \ldots, m\}$. Clearly, $|P| = 2^m - 1$.

Define $P(\eta) := \{f(\eta) \in A : f \in P\}$ and $P(\zeta) := \{f(\zeta) \in E : f \in P\}$. Note that $\tau(P(\eta)) = P(\zeta)$, and that by Lemma 21.8, $P(\eta) \subset D(n)$.

Therefore, to prove the lemma, it suffices to show that $|P(\zeta)| = 2^m - 1$. Suppose that this is not the case. This would give rise to polynomials $g, h \in \mathbb{Z}_p[\mathtt{X}]$, such that

$$\deg(g), \deg(h) \leq t - 1, \ g \neq h, \ g(\eta), h(\eta) \in D(n), \text{ and } \tau(g(\eta)) = \tau(h(\eta)).$$

So we have $n \in C(g(\eta))$ and (as always) $1, p \in C(g(\eta))$. Likewise, we have $1, n, p \in C(h(\eta))$. By Lemma 21.7, for all integers k of the form $n^u p^v$, where u and v range over all non-negative integers, we have

$$k \in C(g(\eta)) \quad \text{and} \quad k \in C(h(\eta)).$$

For any such k, since $\tau(g(\eta)) = \tau(h(\eta))$, we have $\tau(g(\eta))^k = \tau(h(\eta))^k$, and hence

$$\begin{aligned} 0 &= \tau(g(\eta))^k - \tau(h(\eta))^k \\ &= \tau(g(\eta)^k) - \tau(h(\eta)^k) \quad (\tau \text{ is a homomorphism}) \\ &= \tau(g(\eta^k)) - \tau(h(\eta^k)) \quad (k \in C(g(\eta)) \text{ and } k \in C(h(\eta))) \\ &= g(\zeta^k) - h(\zeta^k) \quad (\text{definition of } \tau). \end{aligned}$$

Thus, the polynomial $f := g - h \in \mathbb{Z}_p[\mathtt{X}]$ is a non-zero polynomial of degree at most $t-1$, having roots ζ^k in the field E for all k of the form $n^u p^v$. Now, t is by definition the number of distinct residue classes of the form $[n^u p^v \bmod r] \in \mathbb{Z}_r^*$. Also, since ζ has order r in E^*, for integers k, k', we have $\zeta^k = \zeta^{k'}$ if and only if $k \equiv k' \pmod{r}$. Therefore, as k ranges over all integers of the form $n^u p^v$, ζ^k ranges over precisely t distinct values in E. But since all of these values are roots of the polynomial f, which is non-zero and of degree at most $t-1$, this is impossible (Theorem 9.41). □

We are now (finally!) in a position to complete the proof of Theorem 21.5. Under assumptions (A1), (A2), and (A3), Lemmas 21.9 and 21.10 imply that

$$2^{\min(t,\ell)} - 1 \leq |S| \leq n^{2\lfloor t^{1/2} \rfloor}. \tag{21.3}$$

The contradiction is provided by the following:

Lemma 21.11 *Under assumptions (A4) and (A5), we have*

$$2^{\min(t,\ell)} - 1 > n^{2\lfloor t^{1/2} \rfloor}.$$

Proof. Observe that $\log_2 n \leq \mathrm{len}(n)$, and so it suffices to show that

$$2^{\min(t,\ell)} - 1 > 2^{2\,\mathrm{len}(n)\lfloor t^{1/2} \rfloor},$$

and for this, it suffices to show that

$$\min(t, \ell) > 2\,\mathrm{len}(n)\lfloor t^{1/2} \rfloor,$$

since for any integers $a > b \geq 1$, we have $2^a > 2^b + 1$.

To show that $t > 2\,\mathrm{len}(n)\lfloor t^{1/2} \rfloor$, it suffices to show that $t > 2\,\mathrm{len}(n) t^{1/2}$, i.e., $t > 4\,\mathrm{len}(n)^2$. But observe that by definition, t is the order of the subgroup of $\mathbb{Z}_r^*$ generated by $[n \bmod r]$ and $[p \bmod r]$, which is at least as large as the order of $[n \bmod r]$ in $\mathbb{Z}_r^*$, and by assumption (A4), this is larger than $4\,\mathrm{len}(n)^2$.

Finally, directly by assumption (A5), we have $\ell > 2\,\mathrm{len}(n)\lfloor t^{1/2} \rfloor$. □

That concludes the proof of Theorem 21.5.

Exercise 21.12 Show that if Conjecture 5.44 is true, then the value of r discovered in step (2) of algorithm AKS satisfies $r = O(\mathrm{len}(n)^2)$. □

21.3 Notes

The algorithm presented here is due to Agrawal, Kayal, and Saxena. The paper is currently available only on the Internet [5]. The analysis in the original version of the paper made use of a deep number-theoretic result of Fouvry [28], but it was subsequently noticed that the algorithm can be fully analyzed using just elementary arguments (as we have done here).

If fast algorithms for integer and polynomial arithmetic are used, then using the analysis presented here, it is easy to see that the algorithm runs in time $O^{\sim}(\mathrm{len}(n)^{10.5})$ — see §20.4 for a discussion of the "$O^{\sim}$" notation. More generally, it is easy to see that the algorithm runs in time $O^{\sim}(r^{1.5}\,\mathrm{len}(n)^3)$, where r is the value determined in step (2) of the algorithm. In our analysis of the algorithm, we were able to obtain the bound $r = O(\mathrm{len}(n)^5)$, leading to the running-time bound $O^{\sim}(\mathrm{len}(n)^{10.5})$. Using Fouvry's result, one can show that $r = O(\mathrm{len}(n)^3)$, leading to a running-time bound of $O^{\sim}(\mathrm{len}(n)^{7.5})$. Moreover, if Conjecture 5.44 on the density of Sophie Germain primes is true, then one could show that $r = O(\mathrm{len}(n)^2)$ (see Exercise 21.12), which would lead to a running-time bound of $O^{\sim}(\mathrm{len}(n)^6)$.

Prior to this algorithm, the fastest deterministic, rigorously proved primality test was one introduced by Adleman, Pomerance, and Rumely [4], called the *Jacobi Sum Test*, which runs in time

$$O(\mathrm{len}(n)^{c\,\mathrm{len}(\mathrm{len}(\mathrm{len}(n)))})$$

for some constant c. Note that for numbers n with less than 2^{256} bits, the value of $\mathrm{len}(\mathrm{len}(\mathrm{len}(n)))$ is as most 8, and so this algorithm runs in time $O(\mathrm{len}(n)^{8c})$ for any n that one could ever actually write down.

We also mention the earlier work of Adleman and Huang [3], who gave a probabilistic algorithm whose output is always correct, and which runs in expected polynomial time (i.e., a *Las Vegas* algorithm, in the parlance of §7.2).

Appendix A

Notation and Useful Facts

1. *Logarithm notation.* $\log x$ denotes the natural logarithm of x. The logarithm of x to the base b is denoted $\log_b x$.

2. *Power notation.* We use the notation $S^{\times n}$ to denote the cartesian product of n copies of a set S, and for $x \in S$, $x^{\times n}$ denotes the element of $S^{\times n}$ consisting of n copies of x. We reserve the notation S^n to denote the set of all nth powers of S.

3. *Functions.* For any function f from a set A into a set B, if $A' \subset A$, then $f(A') := \{f(a) \in B : a \in A'\}$. For $b \in B$, $f^{-1}(b) := \{a \in A : f(a) = b\}$, and more generally, for $B' \subset B$, $f^{-1}(B') := \{a \in A : f(a) \in B'\}$.

   f is called **one to one** or **injective** if $f(a) = f(b)$ implies $a = b$. f is called **onto** or **surjective** if $f(A) = B$. f is called **bijective** if it is both injective and surjective; in this case, f is called a **bijection**.

4. *Arithmetic with ∞.* We shall sometimes use the symbols "∞" and "$-\infty$" in simple arithmetic expressions involving real numbers. The interpretation given to such expressions is the usual, natural one, e.g., for all real numbers x, y, we have $-\infty < x < x$, $x + \infty = \infty$, $x - \infty = -\infty$, $\infty + \infty = \infty$, and $(-\infty) + (-\infty) = -\infty$. It is possible to assign meaning to other such expressions, but we will not need to; however, some such expressions have no sensible interpretation (e.g., $\infty - \infty$).

5. *Equivalence relations and equivalence classes.* A binary relation $\equiv$ on a set S is called an **equivalence relation** if for all $x, y, z \in S$, $x \equiv x$, $x \equiv y$ implies $y \equiv x$, and $x \equiv y$ and $y \equiv z$ implies $x \equiv z$.

   Such a relation partitions the set S into disjoint **equivalence classes**: for $x \in S$, define $S_x := \{y \in S : x \equiv y\}$; then every such S_x is non-empty, and all $y \in S$ lie in one and only one such such S_x.

6. *Some handy inequalities.* From the series expansion of e^x as a Taylor series around 0, one can easily verify the following:

(1) $1 + x \leq e^x$ for all x;

(2) $e^x \leq 1 + x + x^2$ for all $|x| \leq 1$;

(3) $1 - x \geq e^{-2x}$ for all $0 \leq x \leq 1/2$.

7. *Estimating sums by integrals.* Using elementary calculus, it is easy to estimate sums over a monotone, non-negative sequence in terms of a definite integral, by interpreting the integral as the area under a curve.

   - For a real-valued function f that is non-negative, continuous, and non-increasing on the closed interval $[a, b]$, we have

   $$\int_a^b f(x)dx \leq \sum_{i=a}^{b} f(i) \leq f(a) + \int_a^b f(x)dx.$$

   - For a real-valued function f that is non-negative, continuous, and non-decreasing on the closed interval $[a, b]$, we have

   $$\int_a^b f(x)dx \leq \sum_{i=a}^{b} f(i) \leq f(b) + \int_a^b f(x)dx.$$

8. *Integrating piece-wise continuous functions.* In discussing the Riemann integral $\int_a^b f(x)dx$, many introductory calculus texts only discuss in any detail the case where the integrand f is continuous on the closed interval $[a, b]$, in which case the integral is always well defined. However, the Riemann integral is well defined for much broader classes of functions. For our purposes in this text, it is convenient and sufficient to work with integrands that are **piece-wise continuous** on $[a, b]$, that is, there exist real numbers $x_0, x_1, \ldots, x_k$ and functions $f_1, \ldots, f_k$, such that $a = x_0 \leq x_1 \leq \cdots \leq x_k = b$, and for $1 \leq i \leq k$, the function f_i is continuous on the *closed* interval $[x_{i-1}, x_i]$, and agrees with f on the *open* interval (x_{i-1}, x_i). In this case, f is integrable on $[a, b]$, and indeed

   $$\int_a^b f(x)dx = \sum_{i=1}^{k} \int_{x_{i-1}}^{x_i} f_i(x)dx.$$

   It is not hard to prove this equality, using the basic definition of the Riemann integral; however, for our purposes, we can also just take the value of the expression on the right-hand side as the definition of the integral on the left-hand side.

   We also say that f is piece-wise continuous on $[a, \infty)$ if for all $b \geq a$, f is piece-wise continuous on $[a, b]$. In this case, we may define the improper integral $\int_a^\infty f(x)dx$ as the limit, as $b \to \infty$, of $\int_a^b f(x)dx$, provided the limit exists.

9. *Infinite series.* It is a basic fact from calculus that if an infinite series $\sum_{i=1}^{\infty} x_i$ of non-negative terms converges to a value y, than any infinite series whose terms are a rearrangement of the x_i's converges to the same value y.

An infinite series $\sum_{i=1}^{\infty} x_i$, where now some of the x_i's may be negative, is called **absolutely convergent** if the series $\sum_{i=1}^{\infty} |x_i|$ is convergent. It is a basic fact from calculus that if an infinite series $\sum_{i=1}^{\infty} x_i$ is absolutely convergent, then not only does the series itself converge to some value y, but any infinite series whose terms are a rearrangement of the x_i's also converges to the same value y.

10. *Double infinite series.* The topic of **double infinite series** may not be discussed in a typical introductory calculus course; we summarize here the basic facts that we need. We state these facts without proof, but all of them are fairly straightforward applications of the definitions.

    Suppose that $x_{ij}, i, j = 1, 2, \ldots$ are *non-negative* real numbers. The ith row gives a series $\sum_j x_{ij}$, and if each of these converges, one can form the double infinite series $\sum_i \sum_j x_{ij}$. Similarly, one may for the double infinite series $\sum_j \sum_i x_{ij}$ One may also arrange the terms x_{ij} in a single infinite series $\sum_{ij} x_{ij}$, using some enumeration of the set of pairs (i, j). Then these three series either all diverge or all converge to the same value.

    If we drop the requirement that the x_{ij}'s are non-negative, but instead require that the single infinite series $\sum_{ij} x_{ij}$ is absolutely convergent, then these three series either all converge to the same value.

Bibliography

[1] L. M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *20th Annual Symposium on Foundations of Computer Science*, pages 55–60, 1979.

[2] L. M. Adleman. The function field sieve. In *Algorithmic Number Theory (ANTS-I)*, pages 108–121, 1994.

[3] L. M. Adleman and M.-D. Huang. *Primality testing and two dimensional Abelian varieties over finite fields (Lecture Notes in Mathematics #1512).* Springer-Verlag, 1992.

[4] L. M. Adleman, C. Pomerance, and R. S. Rumely. On distinguishing prime numbers from composite numbers. *Ann. Math.*, 117:173–206, 1983.

[5] M. Agarwal, N. Kayal, and N. Saxena. PRIMES is in P, manuscript, 2002. Available at `http://www.cse.iitk.ac.in/news/primality`.

[6] W. Alford, A. Granville, and C. Pomerance. There are infintely many Carmichael numbers. *Ann. Math.*, 140:703–722, 1994.

[7] T. Apostol. *Introduction to Analytic Number Theory.* Springer-Verlag, 1973.

[8] E. Bach. How to generate factored random numbers. *SIAM J. Computing*, 17:179–193, 1988.

[9] E. Bach. Explicit bounds for primality testing and related problems. *Math. Comp.*, 55:355–380, 1990.

[10] E. Bach. Efficient prediction of Marsaglia-Zaman random number generators. *IEEE Transactions on Information Theory*, 44:1253–1257, 1998.

[11] E. Bach and J. Shallit. *Algorithmic Number Theory*, volume 1. MIT Press, 1996.

[12] M. Ben-Or. Probabilistic algorithms in finite fields. In *22nd Annual Symposium on Foundations of Computer Science*, pages 394–398, 1981.

[13] E. R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill, 1968.

[14] E. R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24(111):713–735, 1970.

[15] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Computing*, 15:364–383, 1986.

[16] J. P. Buhler, H. W. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. In A. K. Lenstra and H. W. Lenstra, editors, *The Development of the Number Field Sieve*, pages 50–94. Springer-Verlag, 1993.

[17] E. Canfield, P. Erdős, and C. Pomerance. On a problem of Oppenheim concerning 'Factorisatio Numerorum'. *Journal of Number Theory*, 17:1–28, 1983.

[18] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary rings. *Acta Inform.*, 28:693–701, 1991.

[19] S. Cavallar, W. M. Lioen, H. J. J. te Riele, B. Dodson, A. K. Lenstra, P. L. Montgomery, B. Murphy, K. Aardal, J. Gilchrist, G. Guillern, P. Leyland, J. Marchand, F. Morain, A. Muffet, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of a 512-bit RSA modulus. In *Advances in Cryptology–Eurocrypt 2000*, pages 1–18, 2000.

[20] A. L. Chistov. Polynomial time construction of a finite field. In *Abstracts of Lectures at 7th All-Union Conference in Mathematical Logic, Novosibirsk*, page 196, 1984. In Russian.

[21] D. Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6:169–180, 1993.

[22] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comp.*, 9(3):23–52, 1990.

[23] R. Crandall and C. Pomerance. *Prime Numbers: A Computational Perspective*. Springer, 2001.

[24] I. Damgård, P. Landrock, and C. Pomerance. Average case error estimates for the strong probable prime test. *Math. Comp.*, 61:177–194, 1993.

[25] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22:644–654, 1976.

[26] J. Dixon. Asymptotocally fast factorization of integers. *Mathematics of Computation*, 36:255–260, 1981.

[27] J. L. Dornstetter. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Trans. Inf. Theory*, IT-33:428–431, 1987.

[28] E. Fouvry. Theoreme de Brun-Titchmarsh; application au theoreme de Fermat. *Invent. Math.*, 79:383–407, 1985.

[29] D. M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM Journal on Discrete Mathematics*, 6:124–138, 1993.

[30] J. Gordon. Very simple method to find the minimal polynomial of an arbitrary non-zero element of a finite field. *Electronic Letters*, 12:663–664, 1976.

[31] H. Halberstam and H. Richert. *Sieve Methods*. Academic Press, 1974.

[32] G. H. Hardy and J. E. Littlewood. Some problems of partito numerorum. III. On the expression of a number as a sum of primes. *Acta Math.*, 44:1–70, 1923.

[33] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, fifth edition, 1984.

[34] D. Heath-Brown. Zero-free regions for Dirichlet L-functions and the least prime in an arithmetic progression. *Proc. London Math. Soc.*, 64:265–338, 1992.

[35] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random number generation from any one-way function. In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[36] R. Impagliazzo and D. Zuckermann. How to recycle random bits. In *30th Annual Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[37] A. Kalai. Generating random factored numbers, easily. In *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, page 412, 2002.

[38] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. In *27th Annual ACM Symposium on Theory of Computing*, pages 398–406, 1995.

[39] A. Karacuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Dokl.*, 7:595–596, 1963.

[40] S. H. Kim and C. Pomerance. The probability that a random probable prime is composite. *Math. Comp.*, 53(188):721–741, 1989.

[41] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, second edition, 1981.

[42] D. Lehmer and R. Powers. On factoring large numbers. *Bulletin of the AMS*, 37:770–776, 1931.

[43] H. W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[44] H. W. Lenstra and C. Pomerance. A rigorous time bound for factoring integers. *J. Amer. Math. Soc.*, 4:483–516, 1992.

[45] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

[46] J. Massey. Shift-register synthesis and BCH coding. *IEEE Trans. Inf. Theory*, IT-15:122–127, 1969.

[47] A. Menesez, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[48] G. Miller. Riemann's hypothesis and tests for primality. *J. Comput. Sys. Sci.*, 13:300–317, 1976.

[49] W. Mills. Continued fractions and linear recurrences. *Mathematics of Computation*, 29:173–180, 1975.

[50] M. Morrison and J. Brillhart. A method of factoring and the factorization of F_7. *Mathematics of Computation*, 29:183–205, 1975.

[51] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994. Translated from *Matematicheskie Zametki*, 55(2):91–101, 1994.

[52] I. Niven and H. Zuckerman. *An Introduction to the Theory of Numbers*. John Wiley and Sons, Inc., second edition, 1966.

[53] J. Oesterlé. Versions effectives du théorème de Chebotarev sous l'hypothèse de Riemann généralisée. *Astérisque*, 61:165–167, 1979.

[54] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inf. Theory*, 24:106–110, 1978.

[55] J. M. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[56] J. M. Pollard. Factoring with cubic integers. In A. K. Lenstra and H. W. Lenstra, editors, *The Development of the Number Field Sieve*, pages 4–10. Springer-Verlag, 1993.

[57] C. Pomerance. Analysis and comparison of some integer factoring algorithms. In H. W. Lenstra and R. Tijdeman, editors, *Computational Methods in Number Theory, Part I*, pages 89–139. Mathematisch Centrum, 1982.

[58] M. O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity, Recent Results and New Directions*, pages 21–39. Academic Press, 1976.

[59] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *SIAM J. Appl. Math.*, pages 300–304, 1960.

[60] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[61] J. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Ill. J. Math.*, 6:64–94, 1962.

[62] O. Schirokauer, D. Weber, and T. Denny. Discrete logarithms: the effectiveness of the index calculus method. In *Algorithmic Number Theory (ANTS-II)*, pages 337–361, 1996.

[63] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–282, 1971.

[64] I. A. Semaev. Construction of irreducible polynomials over finite fields with linearly independent roots. *Mat. Sbornik*, 135:520–532, 1988. In Russian; English translation in *Math. USSR–Sbornik*, 63(2):507–519, 1989.

[65] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[66] P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[67] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41:303–332, 1999.

[68] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Math. Comp.*, 54(189):435–447, 1990.

[69] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symbolic Comp.*, 17(5):371–391, 1994.

[70] V. Shoup. A new polynomial factorization algorithm and its implementation. *J. Symbolic Comp.*, 20(4):363–397, 1995.

[71] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, pages 256–266, 1997.

[72] R. Solovay and V. Strassen. A fast monte-carlo test for primality. *SIAM J. Comput.*, 6:84–85, 1977.

[73] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[74] J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity*, 2:187–224, 1992.

[75] A. Walfisz. *Weylsche Exponentialsummen in der neueren Zahlentheorie*. VEB Deutscher Verlag der Wissenschaften, 1963.

[76] P. Wang, M. Guy, and J. Davenport. p-adic reconstruction of rational numbers. *SIGSAM Bulletin*, 16:2–3, 1982.

[77] L. Welch and R. Scholtz. Continued fractions and Berlekamp's algorithm. *IEEE Trans. Inf. Theory*, IT-25:19–27, 1979.

[78] D. Wiedemann. Solving sparse linear systems over finite fields. *IEEE Trans. Inf. Theory*, IT-32:54–62, 1986.

[79] D. Y. Y. Yun. On square-free decomposition algorithms. In *Proc. ACM Symp. Symbolic and Algebraic Comp.*, pages 26–35, 1976.