§1.11

One approach to distributing public-keys is the so-called *Merkle channel* (see Simmons [1144, p.387]). Merkle proposed that public keys be distributed over so many independent public channels (newspaper, radio, television, etc.) that it would be improbable for an adversary to compromise all of them.

In 1979 Kohnfelder [702] suggested the idea of using *public-key certificates* to facilitate the distribution of public keys over unsecured channels, such that their authenticity can be verified. Essentially the same idea, but by on-line requests, was proposed by Needham and Schroeder (ses Wilkes [1244]).

A provably secure key agreement protocol has been proposed whose security is based on the Heisenberg uncertainty principle of quantum physics. The security of so-called *quantum cryptography* does not rely upon any complexity-theoretic assumptions. For further details on quantum cryptography, consult Chapter 6 of Brassard [192], and Bennett, Brassard, and Ekert [115].

§1.12

For an introduction and detailed treatment of many pseudorandom sequence generators, see Knuth [692]. Knuth cites an example of a complex scheme to generate random numbers which on closer analysis is shown to produce numbers which are far from random, and concludes: *...random numbers should not be generated with a method chosen at random.*

§1.13

The seminal work of Shannon [1121] on secure communications, published in 1949, remains as one of the best introductions to both practice and theory, clearly presenting many of the fundamental ideas including redundancy, entropy, and unicity distance. Various models under which security may be examined are considered by Rueppel [1081], Simmons [1144], and Preneel [1003], among others; see also Goldwasser [476].

Chapter 2

Mathematical Background

Contents in Brief

This chapter is a collection of basic material on probability theory, information theory, complexity theory, number theory, abstract algebra, and finite fields that will be used throughout this book. Further background and proofs of the facts presented here can be found in the references given in §2.7. The following standard notation will be used throughout:

1. $\mathbb{Z}$ denotes the set of *integers*; that is, the set $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
2. $\mathbb{Q}$ denotes the set of *rational numbers*; that is, the set $\{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\}$.
3. $\mathbb{R}$ denotes the set of *real numbers*.
4. π is the mathematical constant; $\pi \approx 3.14159$.
5. e is the base of the natural logarithm; $e \approx 2.71828$.
6. $[a, b]$ denotes the integers x satisfying $a \leq x \leq b$.
7. $\lfloor x \rfloor$ is the largest integer less than or equal to x. For example, $\lfloor 5.2 \rfloor = 5$ and $\lfloor -5.2 \rfloor = -6$.
8. $\lceil x \rceil$ is the smallest integer greater than or equal to x. For example, $\lceil 5.2 \rceil = 6$ and $\lceil -5.2 \rceil = -5$.
9. If A is a finite set, then $|A|$ denotes the number of elements in A, called the *cardinality* of A.
10. $a \in A$ means that element a is a member of the set A.
11. $A \subseteq B$ means that A is a subset of B.
12. $A \subset B$ means that A is a proper subset of B; that is $A \subseteq B$ and $A \neq B$.
13. The *intersection* of sets A and B is the set $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$.
14. The *union* of sets A and B is the set $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
15. The *difference* of sets A and B is the set $A - B = \{x \mid x \in A \text{ and } x \notin B\}$.
16. The *Cartesian product* of sets A and B is the set $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$. For example, $\{a_1, a_2\} \times \{b_1, b_2, b_3\} = \{(a_1, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_1), (a_2, b_2), (a_2, b_3)\}$.

17. A *function* or *mapping* $f : A \longrightarrow B$ is a rule which assigns to each element a in A precisely one element b in B. If $a \in A$ is mapped to $b \in B$ then b is called the *image* of a, a is called a *preimage* of b, and this is written $f(a) = b$. The set A is called the *domain* of f, and the set B is called the *codomain* of f.
18. A function $f : A \longrightarrow B$ is $1-1$ (*one-to-one*) or *injective* if each element in B is the image of at most one element in A. Hence $f(a_1) = f(a_2)$ implies $a_1 = a_2$.
19. A function $f : A \longrightarrow B$ is *onto* or *surjective* if each $b \in B$ is the image of at least one $a \in A$.
20. A function $f : A \longrightarrow B$ is a *bijection* if it is both one-to-one and onto. If f is a bijection between finite sets A and B, then $|A| = |B|$. If f is a bijection between a set A and itself, then f is called a *permutation* on A.
21. $\ln x$ is the natural logarithm of x; that is, the logarithm of x to the base e.
22. $\lg x$ is the logarithm of x to the base 2.
23. $\exp(x)$ is the exponential function e^x.
24. $\sum_{i=1}^{n} a_i$ denotes the sum $a_1 + a_2 + \cdots + a_n$.
25. $\prod_{i=1}^{n} a_i$ denotes the product $a_1 \cdot a_2 \cdot \cdots \cdot a_n$.
26. For a positive integer n, the factorial function is $n! = n(n-1)(n-2)\cdots 1$. By convention, $0! = 1$.

2.1 Probability theory

2.1.1 Basic definitions

2.1 Definition An *experiment* is a procedure that yields one of a given set of outcomes. The individual possible outcomes are called *simple events*. The set of all possible outcomes is called the *sample space*.

This chapter only considers *discrete* sample spaces; that is, sample spaces with only finitely many possible outcomes. Let the simple events of a sample space S be labeled $s_1, s_2, \ldots, s_n$.

2.2 Definition A *probability distribution* P on S is a sequence of numbers $p_1, p_2, \ldots, p_n$ that are all non-negative and sum to 1. The number p_i is interpreted as the *probability* of s_i being the outcome of the experiment.

2.3 Definition An *event* E is a subset of the sample space S. The *probability* that event E occurs, denoted $P(E)$, is the sum of the probabilities p_i of all simple events s_i which belong to E. If $s_i \in S$, $P(\{s_i\})$ is simply denoted by $P(s_i)$.

2.4 Definition If E is an event, the *complementary event* is the set of simple events not belonging to E, denoted $\overline{E}$.

2.5 Fact Let $E \subseteq S$ be an event.

(i) $0 \le P(E) \le 1$. Furthermore, $P(S) = 1$ and $P(\emptyset) = 0$. ($\emptyset$ is the empty set.)
(ii) $P(\overline{E}) = 1 - P(E)$.

(iii) If the outcomes in S are equally likely, then $P(E) = \frac{|E|}{|S|}$.

2.6 Definition Two events E_1 and E_2 are called *mutually exclusive* if $P(E_1 \cap E_2) = 0$. That is, the occurrence of one of the two events excludes the possibility that the other occurs.

2.7 Fact Let E_1 and E_2 be two events.

(i) If $E_1 \subseteq E_2$, then $P(E_1) \leq P(E_2)$.
(ii) $P(E_1 \cup E_2) + P(E_1 \cap E_2) = P(E_1) + P(E_2)$. Hence, if E_1 and E_2 are mutually exclusive, then $P(E_1 \cup E_2) = P(E_1) + P(E_2)$.

2.1.2 Conditional probability

2.8 Definition Let E_1 and E_2 be two events with $P(E_2) > 0$. The *conditional probability of E_1 given E_2*, denoted $P(E_1|E_2)$, is

$$P(E_1|E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)}.$$

$P(E_1|E_2)$ measures the probability of event E_1 occurring, given that E_2 has occurred.

2.9 Definition Events E_1 and E_2 are said to be *independent* if $P(E_1 \cap E_2) = P(E_1)P(E_2)$.

Observe that if E_1 and E_2 are independent, then $P(E_1|E_2) = P(E_1)$ and $P(E_2|E_1) = P(E_2)$. That is, the occurrence of one event does not influence the likelihood of occurrence of the other.

2.10 Fact (*Bayes' theorem*) If E_1 and E_2 are events with $P(E_2) > 0$, then

$$P(E_1|E_2) = \frac{P(E_1)P(E_2|E_1)}{P(E_2)}.$$

2.1.3 Random variables

Let S be a sample space with probability distribution P.

2.11 Definition A *random variable* X is a function from the sample space S to the set of real numbers; to each simple event $s_i \in S$, X assigns a real number $X(s_i)$.

Since S is assumed to be finite, X can only take on a finite number of values.

2.12 Definition Let X be a random variable on S. The *expected value* or *mean* of X is $E(X) = \sum_{s_i \in S} X(s_i)P(s_i)$.

2.13 Fact Let X be a random variable on S. Then $E(X) = \sum_{x \in \mathbb{R}} x \cdot P(X = x)$.

2.14 Fact If $X_1, X_2, \ldots, X_m$ are random variables on S, and $a_1, a_2, \ldots, a_m$ are real numbers, then $E(\sum_{i=1}^m a_i X_i) = \sum_{i=1}^m a_i E(X_i)$.

2.15 Definition The *variance* of a random variable X of mean μ is a non-negative number defined by

$$\mathrm{Var}(X) = E((X - \mu)^2).$$

The *standard deviation* of X is the non-negative square root of $\mathrm{Var}(X)$.

If a random variable has small variance then large deviations from the mean are unlikely to be observed. This statement is made more precise below.

2.16 Fact (*Chebyshev's inequality*) Let X be a random variable with mean $\mu = E(X)$ and variance $\sigma^2 = \mathrm{Var}(X)$. Then for any $t > 0$,

$$P(|X-\mu| \geq t) \leq \frac{\sigma^2}{t^2}.$$

2.1.4 Binomial distribution

2.17 Definition Let n and k be non-negative integers. The *binomial coefficient* $\binom{n}{k}$ is the number of different ways of choosing k distinct objects from a set of n distinct objects, where the order of choice is not important.

2.18 Fact (*properties of binomial coefficients*) Let n and k be non-negative integers.

(i) $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

(ii) $\binom{n}{k} = \binom{n}{n-k}$.

(iii) $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$.

2.19 Fact (*binomial theorem*) For any real numbers a, b, and non-negative integer n, $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$.

2.20 Definition A *Bernoulli trial* is an experiment with exactly two possible outcomes, called *success* and *failure*.

2.21 Fact Suppose that the probability of success on a particular Bernoulli trial is p. Then the probability of exactly k successes in a sequence of n such independent trials is

$$\binom{n}{k} p^k (1-p)^{n-k}, \text{ for each } 0 \leq k \leq n. \quad (2.1)$$

2.22 Definition The probability distribution (2.1) is called the *binomial distribution*.

2.23 Fact The expected number of successes in a sequence of n independent Bernoulli trials, with probability p of success in each trial, is np. The variance of the number of successes is $np(1-p)$.

2.24 Fact (*law of large numbers*) Let X be the random variable denoting the fraction of successes in n independent Bernoulli trials, with probability p of success in each trial. Then for any $\epsilon > 0$,

$$P(|X-p| > \epsilon) \longrightarrow 0, \text{ as } n \longrightarrow \infty.$$

In other words, as n gets larger, the proportion of successes should be close to p, the probability of success in each trial.

2.1.5 Birthday problems

2.25 Definition

(i) For positive integers m, n with $m \geq n$, the number $m^{(n)}$ is defined as follows:

$$m^{(n)} = m(m-1)(m-2)\cdots(m-n+1).$$

(ii) Let m, n be non-negative integers with $m \geq n$. The *Stirling number of the second kind*, denoted $\left\{ {m \atop n} \right\}$, is

$$\left\{ {m \atop n} \right\} = \frac{1}{n!}\sum_{k=0}^{n}(-1)^{n-k}\binom{n}{k}k^m,$$

with the exception that $\left\{ {0 \atop 0} \right\} = 1$.

The symbol $\left\{ {m \atop n} \right\}$ counts the number of ways of partitioning a set of m objects into n non-empty subsets.

2.26 Fact (*classical occupancy problem*) An urn has m balls numbered 1 to m. Suppose that n balls are drawn from the urn one at a time, with replacement, and their numbers are listed. The probability that exactly t different balls have been drawn is

$$P_1(m,n,t) = \left\{ {n \atop t} \right\}\frac{m^{(t)}}{m^n}, \quad 1 \leq t \leq n.$$

The birthday problem is a special case of the classical occupancy problem.

2.27 Fact (*birthday problem*) An urn has m balls numbered 1 to m. Suppose that n balls are drawn from the urn one at a time, with replacement, and their numbers are listed.

(i) The probability of at least one coincidence (i.e., a ball drawn at least twice) is

$$P_2(m,n) = 1 - P_1(m,n,n) = 1 - \frac{m^{(n)}}{m^n}, \quad 1 \leq n \leq m. \tag{2.2}$$

If $n = O(\sqrt{m})$ (see Definition 2.55) and $m \longrightarrow \infty$, then

$$P_2(m,n) \longrightarrow 1 - \exp\left(-\frac{n(n-1)}{2m} + O\left(\frac{1}{\sqrt{m}}\right)\right) \approx 1 - \exp\left(-\frac{n^2}{2m}\right).$$

(ii) As $m \longrightarrow \infty$, the expected number of draws before a coincidence is $\sqrt{\frac{\pi m}{2}}$.

The following explains why probability distribution (2.2) is referred to as the *birthday surprise* or *birthday paradox*. The probability that at least 2 people in a room of 23 people have the same birthday is $P_2(365, 23) \approx 0.507$, which is surprisingly large. The quantity $P_2(365, n)$ also increases rapidly as n increases; for example, $P_2(365, 30) \approx 0.706$.

A different kind of problem is considered in Facts 2.28, 2.29, and 2.30 below. Suppose that there are two urns, one containing m white balls numbered 1 to m, and the other containing m red balls numbered 1 to m. First, n_1 balls are selected from the first urn and their numbers listed. Then n_2 balls are selected from the second urn and their numbers listed. Finally, the number of coincidences between the two lists is counted.

2.28 Fact (*model A*) If the balls from both urns are drawn one at a time, with replacement, then the probability of at least one coincidence is

$$P_3(m,n_1,n_2) = 1 - \frac{1}{m^{n_1+n_2}}\sum_{t_1,t_2} m^{(t_1+t_2)}\left\{ {n_1 \atop t_1} \right\}\left\{ {n_2 \atop t_2} \right\},$$

where the summation is over all $0 \le t_1 \le n_1$, $0 \le t_2 \le n_2$. If $n = n_1 = n_2$, $n = O(\sqrt{m})$ and $m \longrightarrow \infty$, then

$$P_3(m, n_1, n_2) \longrightarrow 1 - \exp\left(-\frac{n^2}{m}\left[1 + O\left(\frac{1}{\sqrt{m}}\right)\right]\right) \approx 1 - \exp\left(-\frac{n^2}{m}\right).$$

2.29 Fact (*model B*) If the balls from both urns are drawn without replacement, then the probability of at least one coincidence is

$$P_4(m, n_1, n_2) = 1 - \frac{m^{(n_1+n_2)}}{m^{(n_1)} m^{(n_2)}}.$$

If $n_1 = O(\sqrt{m})$, $n_2 = O(\sqrt{m})$, and $m \longrightarrow \infty$, then

$$P_4(m, n_1, n_2) \longrightarrow 1 - \exp\left(-\frac{n_1 n_2}{m}\left[1 + \frac{n_1 + n_2 - 1}{2m} + O\left(\frac{1}{m}\right)\right]\right).$$

2.30 Fact (*model C*) If the n_1 white balls are drawn one at a time, with replacement, and the n_2 red balls are drawn without replacement, then the probability of at least one coincidence is

$$P_5(m, n_1, n_2) = 1 - \left(1 - \frac{n_2}{m}\right)^{n_1}.$$

If $n_1 = O(\sqrt{m})$, $n_2 = O(\sqrt{m})$, and $m \longrightarrow \infty$, then

$$P_5(m, n_1, n_2) \longrightarrow 1 - \exp\left(-\frac{n_1 n_2}{m}\left[1 + O\left(\frac{1}{\sqrt{m}}\right)\right]\right) \approx 1 - \exp\left(-\frac{n_1 n_2}{m}\right).$$

2.1.6 Random mappings

2.31 Definition Let $\mathcal{F}_n$ denote the collection of all functions (mappings) from a finite domain of size n to a finite codomain of size n.

Models where random elements of $\mathcal{F}_n$ are considered are called *random mappings models*. In this section the only random mappings model considered is where every function from $\mathcal{F}_n$ is equally likely to be chosen; such models arise frequently in cryptography and algorithmic number theory. Note that $|\mathcal{F}_n| = n^n$, whence the probability that a particular function from $\mathcal{F}_n$ is chosen is $1/n^n$.

2.32 Definition Let f be a function in $\mathcal{F}_n$ with domain and codomain equal to $\{1, 2, \ldots, n\}$. The *functional graph* of f is a directed graph whose *points* (or *vertices*) are the elements $\{1, 2, \ldots, n\}$ and whose *edges* are the ordered pairs $(x, f(x))$ for all $x \in \{1, 2, \ldots, n\}$.

2.33 Example (*functional graph*) Consider the function $f : \{1, 2, \ldots, 13\} \longrightarrow \{1, 2, \ldots, 13\}$ defined by $f(1) = 4$, $f(2) = 11$, $f(3) = 1$, $f(4) = 6$, $f(5) = 3$, $f(6) = 9$, $f(7) = 3$, $f(8) = 11$, $f(9) = 1$, $f(10) = 2$, $f(11) = 10$, $f(12) = 4$, $f(13) = 7$. The functional graph of f is shown in Figure 2.1. □

As Figure 2.1 illustrates, a functional graph may have several *components* (maximal connected subgraphs), each component consisting of a directed *cycle* and some directed *trees* attached to the cycle.

2.34 Fact As n tends to infinity, the following statements regarding the functional digraph of a random function f from $\mathcal{F}_n$ are true:

(i) The expected number of components is $\frac{1}{2} \ln n$.

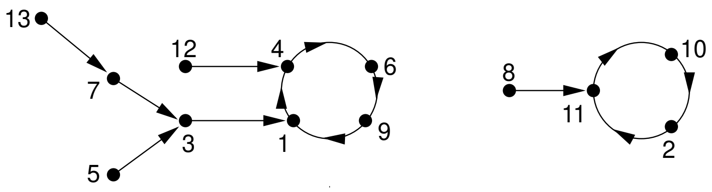

Figure 2.1: *A functional graph (see Example 2.33).*

(ii) The expected number of points which are on the cycles is $\sqrt{\pi n/2}$.

(iii) The expected number of *terminal points* (points which have no preimages) is n/e.

(iv) The expected number of *k-th iterate image points* (x is a k-th iterate image point if $x = \underbrace{f(f(\cdots f}_{k \text{ times}}(y)\cdots))$ for some y) is $(1-\tau_k)n$, where the τ_k satisfy the recurrence $\tau_0 = 0$, $\tau_{k+1} = e^{-1+\tau_k}$ for $k \geq 0$.

2.35 Definition Let f be a random function from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n\}$ and let $u \in \{1, 2, \ldots, n\}$. Consider the sequence of points $u_0, u_1, u_2, \ldots$ defined by $u_0 = u$, $u_i = f(u_{i-1})$ for $i \geq 1$. In terms of the functional graph of f, this sequence describes a path that connects to a cycle.

(i) The number of edges in the path is called the *tail length* of u, denoted $\lambda(u)$.

(ii) The number of edges in the cycle is called the *cycle length* of u, denoted $\mu(u)$.

(iii) The *rho-length* of u is the quantity $\rho(u) = \lambda(u) + \mu(u)$.

(iv) The *tree size* of u is the number of edges in the maximal tree rooted on a cycle in the component that contains u.

(v) The *component size* of u is the number of edges in the component that contains u.

(vi) The *predecessors size* of u is the number of iterated preimages of u.

2.36 Example The functional graph in Figure 2.1 has 2 components and 4 terminal points. The point $u = 3$ has parameters $\lambda(u) = 1$, $\mu(u) = 4$, $\rho(u) = 5$. The tree, component, and predecessors sizes of $u = 3$ are 4, 9, and 3, respectively. □

2.37 Fact As n tends to infinity, the following are the expectations of some parameters associated with a random point in $\{1, 2, \ldots, n\}$ and a random function from $\mathcal{F}_n$: (i) tail length: $\sqrt{\pi n/8}$ (ii) cycle length: $\sqrt{\pi n/8}$ (iii) rho-length: $\sqrt{\pi n/2}$ (iv) tree size: $n/3$ (v) component size: $2n/3$ (vi) predecessors size: $\sqrt{\pi n/8}$.

2.38 Fact As n tends to infinity, the expectations of the maximum tail, cycle, and rho lengths in a random function from $\mathcal{F}_n$ are $c_1\sqrt{n}$, $c_2\sqrt{n}$, and $c_3\sqrt{n}$, respectively, where $c_1 \approx 0.78248$, $c_2 \approx 1.73746$, and $c_3 \approx 2.4149$.

Facts 2.37 and 2.38 indicate that in the functional graph of a random function, most points are grouped together in one giant component, and there is a small number of large trees. Also, almost unavoidably, a cycle of length about $\sqrt{n}$ arises after following a path of length $\sqrt{n}$ edges.

2.2 Information theory

2.2.1 Entropy

Let X be a random variable which takes on a finite set of values $x_1, x_2, \ldots, x_n$, with probability $P(X = x_i) = p_i$, where $0 \leq p_i \leq 1$ for each i, $1 \leq i \leq n$, and where $\sum_{i=1}^{n} p_i = 1$. Also, let Y and Z be random variables which take on finite sets of values.

The entropy of X is a mathematical measure of the amount of information provided by an observation of X. Equivalently, it is the uncertainity about the outcome before an observation of X. Entropy is also useful for approximating the average number of bits required to encode the elements of X.

2.39 Definition The *entropy* or *uncertainty* of X is defined to be $H(X) = -\sum_{i=1}^{n} p_i \lg p_i = \sum_{i=1}^{n} p_i \lg\left(\frac{1}{p_i}\right)$ where, by convention, $p_i \cdot \lg p_i = p_i \cdot \lg\left(\frac{1}{p_i}\right) = 0$ if $p_i = 0$.

2.40 Fact (*properties of entropy*) Let X be a random variable which takes on n values.

(i) $0 \leq H(X) \leq \lg n$.

(ii) $H(X) = 0$ if and only if $p_i = 1$ for some i, and $p_j = 0$ for all $j \neq i$ (that is, there is no uncertainty of the outcome).

(iii) $H(X) = \lg n$ if and only if $p_i = 1/n$ for each i, $1 \leq i \leq n$ (that is, all outcomes are equally likely).

2.41 Definition The *joint entropy* of X and Y is defined to be

$$H(X, Y) = -\sum_{x,y} P(X = x, Y = y) \lg(P(X = x, Y = y)),$$

where the summation indices x and y range over all values of X and Y, respectively. The definition can be extended to any number of random variables.

2.42 Fact If X and Y are random variables, then $H(X, Y) \leq H(X) + H(Y)$, with equality if and only if X and Y are independent.

2.43 Definition If X, Y are random variables, the *conditional entropy of X given $Y = y$* is

$$H(X|Y = y) = -\sum_{x} P(X = x|Y = y) \lg(P(X = x|Y = y)),$$

where the summation index x ranges over all values of X. The *conditional entropy of X given Y*, also called the *equivocation of Y about X*, is

$$H(X|Y) = \sum_{y} P(Y = y) H(X|Y = y),$$

where the summation index y ranges over all values of Y.

2.44 Fact (*properties of conditional entropy*) Let X and Y be random variables.

(i) The quantity $H(X|Y)$ measures the amount of uncertainty remaining about X after Y has been observed.

(ii) $H(X|Y) \geq 0$ and $H(X|X) = 0$.
(iii) $H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$.
(iv) $H(X|Y) \leq H(X)$, with equality if and only if X and Y are independent.

2.2.2 Mutual information

2.45 Definition The *mutual information* or *transinformation* of random variables X and Y is $I(X;Y) = H(X) - H(X|Y)$. Similarly, the transinformation of X and the pair Y, Z is defined to be $I(X;Y,Z) = H(X) - H(X|Y,Z)$.

2.46 Fact (*properties of mutual transinformation*)

(i) The quantity $I(X;Y)$ can be thought of as the amount of information that Y reveals about X. Similarly, the quantity $I(X;Y,Z)$ can be thought of as the amount of information that Y and Z together reveal about X.
(ii) $I(X;Y) \geq 0$.
(iii) $I(X;Y) = 0$ if and only if X and Y are independent (that is, Y contributes no information about X).
(iv) $I(X;Y) = I(Y;X)$.

2.47 Definition The *conditional transinformation* of the pair X, Y given Z is defined to be $I_Z(X;Y) = H(X|Z) - H(X|Y,Z)$.

2.48 Fact (*properties of conditional transinformation*)

(i) The quantity $I_Z(X;Y)$ can be interpreted as the amount of information that Y provides about X, given that Z has already been observed.
(ii) $I(X;Y,Z) = I(X;Y) + I_Y(X;Z)$.
(iii) $I_Z(X;Y) = I_Z(Y;X)$.

2.3 Complexity theory

2.3.1 Basic definitions

The main goal of complexity theory is to provide mechanisms for classifying computational problems according to the resources needed to solve them. The classification should not depend on a particular computational model, but rather should measure the intrinsic difficulty of the problem. The resources measured may include time, storage space, random bits, number of processors, etc., but typically the main focus is time, and sometimes space.

2.49 Definition An *algorithm* is a well-defined computational procedure that takes a variable input and halts with an output.

Of course, the term "well-defined computational procedure" is not mathematically precise. It can be made so by using formal computational models such as Turing machines, random-access machines, or boolean circuits. Rather than get involved with the technical intricacies of these models, it is simpler to think of an algorithm as a computer program written in some specific programming language for a specific computer that takes a variable input and halts with an output.

It is usually of interest to find the most efficient (i.e., fastest) algorithm for solving a given computational problem. The time that an algorithm takes to halt depends on the "size" of the problem instance. Also, the unit of time used should be made precise, especially when comparing the performance of two algorithms.

2.50 Definition The *size* of the input is the total number of bits needed to represent the input in ordinary binary notation using an appropriate encoding scheme. Occasionally, the size of the input will be the number of items in the input.

2.51 Example (*sizes of some objects*)

(i) The number of bits in the binary representation of a positive integer n is $1 + \lfloor \lg n \rfloor$ bits. For simplicity, the size of n will be approximated by $\lg n$.

(ii) If f is a polynomial of degree at most k, each coefficient being a non-negative integer at most n, then the size of f is $(k + 1) \lg n$ bits.

(iii) If A is a matrix with r rows, s columns, and with non-negative integer entries each at most n, then the size of A is $rs \lg n$ bits. □

2.52 Definition The *running time* of an algorithm on a particular input is the number of primitive operations or "steps" executed.

Often a step is taken to mean a bit operation. For some algorithms it will be more convenient to take step to mean something else such as a comparison, a machine instruction, a machine clock cycle, a modular multiplication, etc.

2.53 Definition The *worst-case running time* of an algorithm is an upper bound on the running time for any input, expressed as a function of the input size.

2.54 Definition The *average-case running time* of an algorithm is the average running time over all inputs of a fixed size, expressed as a function of the input size.

2.3.2 Asymptotic notation

It is often difficult to derive the exact running time of an algorithm. In such situations one is forced to settle for approximations of the running time, and usually may only derive the *asymptotic* running time. That is, one studies how the running time of the algorithm increases as the size of the input increases without bound.

In what follows, the only functions considered are those which are defined on the positive integers and take on real values that are always positive from some point onwards. Let f and g be two such functions.

2.55 Definition (*order notation*)

(i) (*asymptotic upper bound*) $f(n) = O(g(n))$ if there exists a positive constant c and a positive integer n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.

(ii) (*asymptotic lower bound*) $f(n) = \Omega(g(n))$ if there exists a positive constant c and a positive integer n_0 such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0$.
(iii) (*asymptotic tight bound*) $f(n) = \Theta(g(n))$ if there exist positive constants c_1 and c_2, and a positive integer n_0 such that $c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$.
(iv) (*o-notation*) $f(n) = o(g(n))$ if for any positive constant $c > 0$ there exists a constant $n_0 > 0$ such that $0 \le f(n) < cg(n)$ for all $n \ge n_0$.

Intuitively, $f(n) = O(g(n))$ means that f grows no faster asymptotically than $g(n)$ to within a constant multiple, while $f(n) = \Omega(g(n))$ means that $f(n)$ grows at least as fast asymptotically as $g(n)$ to within a constant multiple. $f(n) = o(g(n))$ means that $g(n)$ is an upper bound for $f(n)$ that is not asymptotically tight, or in other words, the function $f(n)$ becomes insignificant relative to $g(n)$ as n gets larger. The expression $o(1)$ is often used to signify a function $f(n)$ whose limit as n approaches ∞ is 0.

2.56 Fact (*properties of order notation*) For any functions $f(n)$, $g(n)$, $h(n)$, and $l(n)$, the following are true.

(i) $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$.
(ii) $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
(iii) If $f(n) = O(h(n))$ and $g(n) = O(h(n))$, then $(f+g)(n) = O(h(n))$.
(iv) If $f(n) = O(h(n))$ and $g(n) = O(l(n))$, then $(f \cdot g)(n) = O(h(n)l(n))$.
(v) (*reflexivity*) $f(n) = O(f(n))$.
(vi) (*transitivity*) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.

2.57 Fact (*approximations of some commonly occurring functions*)

(i) (*polynomial function*) If $f(n)$ is a polynomial of degree k with positive leading term, then $f(n) = \Theta(n^k)$.
(ii) For any constant $c > 0$, $\log_c n = \Theta(\lg n)$.
(iii) (*Stirling's formula*) For all integers $n \ge 1$,

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \;\le\; n! \;\le\; \sqrt{2\pi n}\left(\frac{n}{e}\right)^{n+(1/(12n))}.$$

Thus $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta(\frac{1}{n})\right)$. Also, $n! = o(n^n)$ and $n! = \Omega(2^n)$.
(iv) $\lg(n!) = \Theta(n \lg n)$.

2.58 Example (*comparative growth rates of some functions*) Let ϵ and c be arbitrary constants with $0 < \epsilon < 1 < c$. The following functions are listed in increasing order of their asymptotic growth rates:

$$1 < \ln\ln n < \ln n < \exp(\sqrt{\ln n \ln\ln n}) < n^\epsilon < n^c < n^{\ln n} < c^n < n^n < c^{c^n}. \quad \square$$

2.3.3 Complexity classes

2.59 Definition A *polynomial-time algorithm* is an algorithm whose worst-case running time function is of the form $O(n^k)$, where n is the input size and k is a constant. Any algorithm whose running time cannot be so bounded is called an *exponential-time algorithm*.

Roughly speaking, polynomial-time algorithms can be equated with *good* or *efficient* algorithms, while exponential-time algorithms are considered *inefficient*. There are, however, some practical situations when this distinction is not appropriate. When considering polynomial-time complexity, the degree of the polynomial is significant. For example, even though an algorithm with a running time of $O(n^{\ln\ln n})$, n being the input size, is asymptotically slower that an algorithm with a running time of $O(n^{100})$, the former algorithm may be faster in practice for smaller values of n, especially if the constants hidden by the big-O notation are smaller. Furthermore, in cryptography, average-case complexity is more important than worst-case complexity — a necessary condition for an encryption scheme to be considered secure is that the corresponding cryptanalysis problem is difficult on average (or more precisely, almost always difficult), and not just for some isolated cases.

2.60 Definition A *subexponential-time algorithm* is an algorithm whose worst-case running time function is of the form $e^{o(n)}$, where n is the input size.

A subexponential-time algorithm is asymptotically faster than an algorithm whose running time is fully exponential in the input size, while it is asymptotically slower than a polynomial-time algorithm.

2.61 Example (*subexponential running time*) Let A be an algorithm whose inputs are either elements of a finite field $\mathbb{F}_q$ (see §2.6), or an integer q. If the expected running time of A is of the form

$$L_q[\alpha, c] = O\left(\exp\left((c + o(1))(\ln q)^{\alpha}(\ln\ln q)^{1-\alpha}\right)\right), \tag{2.3}$$

where c is a positive constant, and α is a constant satisfying $0 < \alpha < 1$, then A is a subexponential-time algorithm. Observe that for $\alpha = 0$, $L_q[0, c]$ is a polynomial in $\ln q$, while for $\alpha = 1$, $L_q[1, c]$ is a polynomial in q, and thus fully exponential in $\ln q$. □

For simplicity, the theory of computational complexity restricts its attention to *decision problems*, i.e., problems which have either YES or NO as an answer. This is not too restrictive in practice, as all the computational problems that will be encountered here can be phrased as decision problems in such a way that an efficient algorithm for the decision problem yields an efficient algorithm for the computational problem, and vice versa.

2.62 Definition The complexity class **P** is the set of all decision problems that are solvable in polynomial time.

2.63 Definition The complexity class **NP** is the set of all decision problems for which a YES answer can be verified in polynomial time given some extra information, called a *certificate*.

2.64 Definition The complexity class **co-NP** is the set of all decision problems for which a NO answer can be verified in polynomial time using an appropriate certificate.

It must be emphasized that if a decision problem is in **NP**, it may not be the case that the certificate of a YES answer can be easily obtained; what is asserted is that such a certificate does exist, and, if known, can be used to efficiently verify the YES answer. The same is true of the NO answers for problems in **co-NP**.

2.65 Example (*problem in* **NP**) Consider the following decision problem:
COMPOSITES
INSTANCE: A positive integer n.
QUESTION: Is n composite? That is, are there integers $a, b > 1$ such that $n = ab$?

COMPOSITES belongs to **NP** because if an integer n is composite, then this fact can be verified in polynomial time if one is given a divisor a of n, where $1 < a < n$ (the certificate in this case consists of the divisor a). It is in fact also the case that COMPOSITES belongs to **co-NP**. It is still unknown whether or not COMPOSITES belongs to **P**. □

2.66 Fact **P** $\subseteq$ **NP** and **P** $\subseteq$ **co-NP**.

The following are among the outstanding unresolved questions in the subject of complexity theory:

1. Is **P** $=$ **NP**?
2. Is **NP** $=$ **co-NP**?
3. Is **P** $=$ **NP** $\cap$ **co-NP**?

Most experts are of the opinion that the answer to each of the three questions is NO, although nothing along these lines has been proven.

The notion of reducibility is useful when comparing the relative difficulties of problems.

2.67 Definition Let L_1 and L_2 be two decision problems. L_1 is said to *polytime reduce* to L_2, written $L_1 \leq_P L_2$, if there is an algorithm that solves L_1 which uses, as a subroutine, an algorithm for solving L_2, and which runs in polynomial time if the algorithm for L_2 does.

Informally, if $L_1 \leq_P L_2$, then L_2 is at least as difficult as L_1, or, equivalently, L_1 is no harder than L_2.

2.68 Definition Let L_1 and L_2 be two decision problems. If $L_1 \leq_P L_2$ and $L_2 \leq_P L_1$, then L_1 and L_2 are said to be *computationally equivalent*.

2.69 Fact Let L_1, L_2, and L_3 be three decision problems.

(i) (*transitivity*) If $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, then $L_1 \leq_P L_3$.
(ii) If $L_1 \leq_P L_2$ and $L_2 \in$ **P**, then $L_1 \in$ **P**.

2.70 Definition A decision problem L is said to be **NP**-*complete* if

(i) $L \in$ **NP**, and
(ii) $L_1 \leq_P L$ for every $L_1 \in$ **NP**.

The class of all **NP**-complete problems is denoted by **NPC**.

NP-complete problems are the hardest problems in **NP** in the sense that they are at least as difficult as every other problem in **NP**. There are thousands of problems drawn from diverse fields such as combinatorics, number theory, and logic, that are known to be **NP**-complete.

2.71 Example (*subset sum problem*) The *subset sum problem* is the following: given a set of positive integers $\{a_1, a_2, \ldots, a_n\}$ and a positive integer s, determine whether or not there is a subset of the a_i that sum to s. The subset sum problem is **NP**-complete. □

2.72 Fact Let L_1 and L_2 be two decision problems.

(i) If L_1 is **NP**-complete and $L_1 \in$ **P**, then **P** = **NP**.
(ii) If $L_1 \in$ **NP**, L_2 is **NP**-complete, and $L_2 \leq_P L_1$, then L_1 is also **NP**-complete.
(iii) If L_1 is **NP**-complete and $L_1 \in$ **co-NP**, then **NP** = **co-NP**.

By Fact 2.72(i), if a polynomial-time algorithm is found for any single **NP**-complete problem, then it is the case that **P** = **NP**, a result that would be extremely surprising. Hence, a proof that a problem is **NP**-complete provides strong evidence for its intractability. Figure 2.2 illustrates what is widely believed to be the relationship between the complexity classes **P**, **NP**, **co-NP**, and **NPC**.

Fact 2.72(ii) suggests the following procedure for proving that a decision problem L_1 is **NP**-complete:

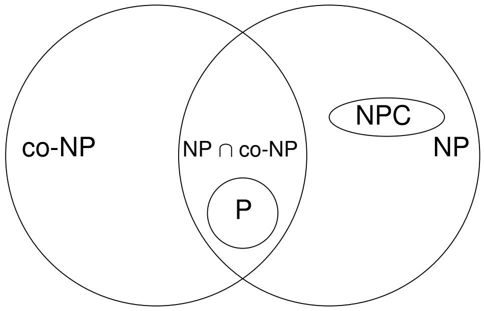

Figure 2.2: *Conjectured relationship between the complexity classes* **P**, **NP**, **co-NP**, *and* **NPC**.

1. Prove that $L_1 \in$ **NP**.
2. Select a problem L_2 that is known to be **NP**-complete.
3. Prove that $L_2 \leq_P L_1$.

2.73 Definition A problem is **NP**-*hard* if there exists some **NP**-complete problem that polytime reduces to it.

Note that the **NP**-hard classification is not restricted to only decision problems. Observe also that an **NP**-complete problem is also **NP**-hard.

2.74 Example (**NP**-*hard problem*) Given positive integers $a_1, a_2, \ldots, a_n$ and a positive integer s, the computational version of the subset sum problem would ask to actually find a subset of the a_i which sums to s, provided that such a subset exists. This problem is **NP**-hard. □

2.3.4 Randomized algorithms

The algorithms studied so far in this section have been *deterministic*; such algorithms follow the same execution path (sequence of operations) each time they execute with the same input. By contrast, a *randomized* algorithm makes random decisions at certain points in the execution; hence their execution paths may differ each time they are invoked with the same input. The random decisions are based upon the outcome of a random number generator. Remarkably, there are many problems for which randomized algorithms are known that are more efficient, both in terms of time and space, than the best known deterministic algorithms.

Randomized algorithms for decision problems can be classified according to the probability that they return the correct answer.

2.75 Definition Let A be a randomized algorithm for a decision problem L, and let I denote an arbitrary instance of L.

(i) A has *0-sided error* if $P(A$ outputs YES $\mid I$'s answer is YES $) = 1$, and $P(A$ outputs YES $\mid I$'s answer is NO $) = 0$.
(ii) A has *1-sided error* if $P(A$ outputs YES $\mid I$'s answer is YES $) \geq \frac{1}{2}$, and $P(A$ outputs YES $\mid I$'s answer is NO $) = 0$.

(iii) A has *2-sided error* if $P(A$ outputs YES $\mid$ I's answer is YES $) \geq \frac{2}{3}$, and $P(A$ outputs YES $\mid$ I's answer is NO $) \leq \frac{1}{3}$.

The number $\frac{1}{2}$ in the definition of 1-sided error is somewhat arbitrary and can be replaced by any positive constant. Similarly, the numbers $\frac{2}{3}$ and $\frac{1}{3}$ in the definition of 2-sided error, can be replaced by $\frac{1}{2}+\epsilon$ and $\frac{1}{2}-\epsilon$, respectively, for any constant ϵ, $0<\epsilon<\frac{1}{2}$.

2.76 Definition The *expected running time* of a randomized algorithm is an upper bound on the expected running time for each input (the expectation being over all outputs of the random number generator used by the algorithm), expressed as a function of the input size.

The important randomized complexity classes are defined next.

2.77 Definition (*randomized complexity classes*)

(i) The complexity class **ZPP** ("zero-sided probabilistic polynomial time") is the set of all decision problems for which there is a randomized algorithm with 0-sided error which runs in expected polynomial time.
(ii) The complexity class **RP** ("randomized polynomial time") is the set of all decision problems for which there is a randomized algorithm with 1-sided error which runs in (worst-case) polynomial time.
(iii) The complexity class **BPP** ("bounded error probabilistic polynomial time") is the set of all decision problems for which there is a randomized algorithm with 2-sided error which runs in (worst-case) polynomial time.

2.78 Fact $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP}$ and $\mathbf{RP} \subseteq \mathbf{NP}$.

2.4 Number theory

2.4.1 The integers

The set of integers $\{\ldots,-3,-2,-1,0,1,2,3,\ldots\}$ is denoted by the symbol $\mathbb{Z}$.

2.79 Definition Let a, b be integers. Then a *divides* b (equivalently: a is a *divisor* of b, or a is a *factor* of b) if there exists an integer c such that $b=ac$. If a divides b, then this is denoted by $a|b$.

2.80 Example (i) $-3|18$, since $18=(-3)(-6)$. (ii) $173|0$, since $0=(173)(0)$. □

The following are some elementary properties of divisibility.

2.81 Fact (*properties of divisibility*) For all a, b, $c \in \mathbb{Z}$, the following are true:

(i) $a|a$.
(ii) If $a|b$ and $b|c$, then $a|c$.
(iii) If $a|b$ and $a|c$, then $a|(bx+cy)$ for all $x,y \in \mathbb{Z}$.
(iv) If $a|b$ and $b|a$, then $a=\pm b$.

2.82 Definition (*division algorithm for integers*) If a and b are integers with $b \geq 1$, then ordinary long division of a by b yields integers q (the *quotient*) and r (the *remainder*) such that

$$a = qb + r, \quad \text{where } 0 \leq r < b.$$

Moreover, q and r are unique. The remainder of the division is denoted $a \bmod b$, and the quotient is denoted $a \operatorname{div} b$.

2.83 Fact Let $a, b \in \mathbb{Z}$ with $b \neq 0$. Then $a \operatorname{div} b = \lfloor a/b \rfloor$ and $a \bmod b = a - b\lfloor a/b \rfloor$.

2.84 Example If $a = 73$, $b = 17$, then $q = 4$ and $r = 5$. Hence $73 \bmod 17 = 5$ and $73 \operatorname{div} 17 = 4$. □

2.85 Definition An integer c is a *common divisor* of a and b if $c|a$ and $c|b$.

2.86 Definition A non-negative integer d is the *greatest common divisor* of integers a and b, denoted $d = \gcd(a, b)$, if

(i) d is a common divisor of a and b; and
(ii) whenever $c|a$ and $c|b$, then $c|d$.

Equivalently, $\gcd(a, b)$ is the largest positive integer that divides both a and b, with the exception that $\gcd(0, 0) = 0$.

2.87 Example The common divisors of 12 and 18 are $\{\pm 1, \pm 2, \pm 3, \pm 6\}$, and $\gcd(12, 18) = 6$. □

2.88 Definition A non-negative integer d is the *least common multiple* of integers a and b, denoted $d = \operatorname{lcm}(a, b)$, if

(i) $a|d$ and $b|d$; and
(ii) whenever $a|c$ and $b|c$, then $d|c$.

Equivalently, $\operatorname{lcm}(a, b)$ is the smallest non-negative integer divisible by both a and b.

2.89 Fact If a and b are positive integers, then $\operatorname{lcm}(a, b) = a \cdot b / \gcd(a, b)$.

2.90 Example Since $\gcd(12, 18) = 6$, it follows that $\operatorname{lcm}(12, 18) = 12 \cdot 18/6 = 36$. □

2.91 Definition Two integers a and b are said to be *relatively prime* or *coprime* if $\gcd(a, b) = 1$.

2.92 Definition An integer $p \geq 2$ is said to be *prime* if its only positive divisors are 1 and p. Otherwise, p is called *composite*.

The following are some well known facts about prime numbers.

2.93 Fact If p is prime and $p|ab$, then either $p|a$ or $p|b$ (or both).

2.94 Fact There are an infinite number of prime numbers.

2.95 Fact (*prime number theorem*) Let $\pi(x)$ denote the number of prime numbers $\leq x$. Then

$$\lim_{x \to \infty} \frac{\pi(x)}{x / \ln x} = 1.$$

This means that for large values of x, $\pi(x)$ is closely approximated by the expression $x/\ln x$. For instance, when $x = 10^{10}$, $\pi(x) = 455,052,511$, whereas $\lfloor x/\ln x \rfloor = 434,294,481$. A more explicit estimate for $\pi(x)$ is given below.

2.96 Fact Let $\pi(x)$ denote the number of primes $\leq x$. Then for $x \geq 17$

$$\pi(x) > \frac{x}{\ln x}$$

and for $x > 1$

$$\pi(x) < 1.25506\frac{x}{\ln x}.$$

2.97 Fact (*fundamental theorem of arithmetic*) Every integer $n \geq 2$ has a factorization as a product of prime powers:

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k},$$

where the p_i are distinct primes, and the e_i are positive integers. Furthermore, the factorization is unique up to rearrangement of factors.

2.98 Fact If $a = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, $b = p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$, where each $e_i \geq 0$ and $f_i \geq 0$, then

$$\gcd(a,b) = p_1^{\min(e_1,f_1)} p_2^{\min(e_2,f_2)} \cdots p_k^{\min(e_k,f_k)}$$

and

$$\operatorname{lcm}(a,b) = p_1^{\max(e_1,f_1)} p_2^{\max(e_2,f_2)} \cdots p_k^{\max(e_k,f_k)}.$$

2.99 Example Let $a = 4864 = 2^8 \cdot 19$, $b = 3458 = 2 \cdot 7 \cdot 13 \cdot 19$. Then $\gcd(4864, 3458) = 2 \cdot 19 = 38$ and $\operatorname{lcm}(4864, 3458) = 2^8 \cdot 7 \cdot 13 \cdot 19 = 442624$. □

2.100 Definition For $n \geq 1$, let $\phi(n)$ denote the number of integers in the interval $[1, n]$ which are relatively prime to n. The function ϕ is called the *Euler phi function* (or the *Euler totient function*).

2.101 Fact (*properties of Euler phi function*)

(i) If p is a prime, then $\phi(p) = p - 1$.

(ii) The Euler phi function is *multiplicative*. That is, if $\gcd(m, n) = 1$, then $\phi(mn) = \phi(m) \cdot \phi(n)$.

(iii) If $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ is the prime factorization of n, then

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_k}\right).$$

Fact 2.102 gives an explicit lower bound for $\phi(n)$.

2.102 Fact For all integers $n \geq 5$,

$$\phi(n) > \frac{n}{6 \ln \ln n}.$$

2.4.2 Algorithms in $\mathbb{Z}$

Let a and b be non-negative integers, each less than or equal to n. Recall (Example 2.51) that the number of bits in the binary representation of n is $\lfloor \lg n \rfloor + 1$, and this number is approximated by $\lg n$. The number of bit operations for the four basic integer operations of addition, subtraction, multiplication, and division using the classical algorithms is summarized in Table 2.1. These algorithms are studied in more detail in §14.2. More sophisticated techniques for multiplication and division have smaller complexities.

Operation		Bit complexity
Addition	$a+b$	$O(\lg a + \lg b) = O(\lg n)$
Subtraction	$a-b$	$O(\lg a + \lg b) = O(\lg n)$
Multiplication	$a \cdot b$	$O((\lg a)(\lg b)) = O((\lg n)^2)$
Division	$a = qb + r$	$O((\lg q)(\lg b)) = O((\lg n)^2)$

Table 2.1: *Bit complexity of basic operations in $\mathbb{Z}$.*

The greatest common divisor of two integers a and b can be computed via Fact 2.98. However, computing a gcd by first obtaining prime-power factorizations does not result in an efficient algorithm, as the problem of factoring integers appears to be relatively difficult. The Euclidean algorithm (Algorithm 2.104) is an efficient algorithm for computing the greatest common divisor of two integers that does not require the factorization of the integers. It is based on the following simple fact.

2.103 Fact If a and b are positive integers with $a > b$, then $\gcd(a, b) = \gcd(b, a \bmod b)$.

2.104 Algorithm Euclidean algorithm for computing the greatest common divisor of two integers

INPUT: two non-negative integers a and b with $a \geq b$.
OUTPUT: the greatest common divisor of a and b.

1. While $b \neq 0$ do the following:
   1.1 Set $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$.
2. Return(a).

2.105 Fact Algorithm 2.104 has a running time of $O((\lg n)^2)$ bit operations.

2.106 Example (*Euclidean algorithm*) The following are the division steps of Algorithm 2.104 for computing $\gcd(4864, 3458) = 38$:

$$\begin{aligned}
4864 &= 1 \cdot 3458 + 1406 \\
3458 &= 2 \cdot 1406 + 646 \\
1406 &= 2 \cdot 646 + 114 \\
646 &= 5 \cdot 114 + 76 \\
114 &= 1 \cdot 76 + 38 \\
76 &= 2 \cdot 38 + 0.
\end{aligned}$$

□

The Euclidean algorithm can be extended so that it not only yields the greatest common divisor d of two integers a and b, but also integers x and y satisfying $ax + by = d$.

2.107 Algorithm Extended Euclidean algorithm

INPUT: two non-negative integers a and b with $a \geq b$.
OUTPUT: $d = \gcd(a, b)$ and integers x, y satisfying $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$, and return(d,x,y).
2. Set $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$.
3. While $b > 0$ do the following:
   3.1 $q \leftarrow \lfloor a/b \rfloor$, $r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$.
   3.2 $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, and $y_1 \leftarrow y$.
4. Set $d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$, and return(d,x,y).

2.108 Fact Algorithm 2.107 has a running time of $O((\lg n)^2)$ bit operations.

2.109 Example (*extended Euclidean algorithm*) Table 2.2 shows the steps of Algorithm 2.107 with inputs $a = 4864$ and $b = 3458$. Hence $\gcd(4864, 3458) = 38$ and $(4864)(32) + (3458)(-45) = 38$. □

q	r	x	y	a	b	x_2	x_1	y_2	y_1
−	−	−	−	4864	3458	1	0	0	1
1	1406	1	−1	3458	1406	0	1	1	−1
2	646	−2	3	1406	646	1	−2	−1	3
2	114	5	−7	646	114	−2	5	3	−7
5	76	−27	38	114	76	5	−27	−7	38
1	38	32	−45	76	38	−27	32	38	−45
2	0	−91	128	38	0	32	−91	−45	128

Table 2.2: *Extended Euclidean algorithm (Algorithm 2.107) with inputs $a = 4864$, $b = 3458$.*

Efficient algorithms for gcd and extended gcd computations are further studied in §14.4.

2.4.3 The integers modulo n

Let n be a positive integer.

2.110 Definition If a and b are integers, then a is said to be *congruent to b modulo n*, written $a \equiv b \pmod{n}$, if n divides $(a-b)$. The integer n is called the *modulus* of the congruence.

2.111 Example (i) $24 \equiv 9 \pmod{5}$ since $24 - 9 = 3 \cdot 5$.
(ii) $-11 \equiv 17 \pmod{7}$ since $-11 - 17 = -4 \cdot 7$. □

2.112 Fact (*properties of congruences*) For all $a, a_1, b, b_1, c \in \mathbb{Z}$, the following are true.

(i) $a \equiv b \pmod{n}$ if and only if a and b leave the same remainder when divided by n.
(ii) (*reflexivity*) $a \equiv a \pmod{n}$.
(iii) (*symmetry*) If $a \equiv b \pmod{n}$ then $b \equiv a \pmod{n}$.

(iv) (*transitivity*) If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$.
(v) If $a \equiv a_1 \pmod{n}$ and $b \equiv b_1 \pmod{n}$, then $a + b \equiv a_1 + b_1 \pmod{n}$ and $ab \equiv a_1b_1 \pmod{n}$.

The *equivalence class* of an integer a is the set of all integers congruent to a modulo n. From properties (ii), (iii), and (iv) above, it can be seen that for a fixed n the relation of congruence modulo n partitions $\mathbb{Z}$ into equivalence classes. Now, if $a = qn + r$, where $0 \le r < n$, then $a \equiv r \pmod{n}$. Hence each integer a is congruent modulo n to a unique integer between 0 and $n-1$, called the *least residue* of a modulo n. Thus a and r are in the same equivalence class, and so r may simply be used to represent this equivalence class.

2.113 Definition The *integers modulo* n, denoted $\mathbb{Z}_n$, is the set of (equivalence classes of) integers $\{0, 1, 2, \ldots, n-1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_n$ are performed modulo n.

2.114 Example $\mathbb{Z}_{25} = \{0, 1, 2, \ldots, 24\}$. In $\mathbb{Z}_{25}$, $13 + 16 = 4$, since $13 + 16 = 29 \equiv 4 \pmod{25}$. Similarly, $13 \cdot 16 = 8$ in $\mathbb{Z}_{25}$. □

2.115 Definition Let $a \in \mathbb{Z}_n$. The *multiplicative inverse* of a modulo n is an integer $x \in \mathbb{Z}_n$ such that $ax \equiv 1 \pmod{n}$. If such an x exists, then it is unique, and a is said to be *invertible*, or a *unit*; the inverse of a is denoted by a^{-1}.

2.116 Definition Let $a, b \in \mathbb{Z}_n$. *Division* of a by b modulo n is the product of a and b^{-1} modulo n, and is only defined if b is invertible modulo n.

2.117 Fact Let $a \in \mathbb{Z}_n$. Then a is invertible if and only if $\gcd(a, n) = 1$.

2.118 Example The invertible elements in $\mathbb{Z}_9$ are 1, 2, 4, 5, 7, and 8. For example, $4^{-1} = 7$ because $4 \cdot 7 \equiv 1 \pmod{9}$. □

The following is a generalization of Fact 2.117.

2.119 Fact Let $d = \gcd(a, n)$. The congruence equation $ax \equiv b \pmod{n}$ has a solution x if and only if d divides b, in which case there are exactly d solutions between 0 and $n-1$; these solutions are all congruent modulo n/d.

2.120 Fact (*Chinese remainder theorem, CRT*) If the integers $n_1, n_2, \ldots, n_k$ are pairwise relatively prime, then the system of simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

has a unique solution modulo $n = n_1 n_2 \cdots n_k$.

2.121 Algorithm (*Gauss's algorithm*) The solution x to the simultaneous congruences in the Chinese remainder theorem (Fact 2.120) may be computed as $x = \sum_{i=1}^{k} a_i N_i M_i \bmod n$, where $N_i = n/n_i$ and $M_i = N_i^{-1} \bmod n_i$. These computations can be performed in $O((\lg n)^2)$ bit operations.

Another efficient practical algorithm for solving simultaneous congruences in the Chinese remainder theorem is presented in §14.5.

2.122 Example The pair of congruences $x \equiv 3 \pmod 7$, $x \equiv 7 \pmod{13}$ has a unique solution $x \equiv 59 \pmod{91}$. □

2.123 Fact If $\gcd(n_1, n_2) = 1$, then the pair of congruences $x \equiv a \pmod{n_1}$, $x \equiv a \pmod{n_2}$ has a unique solution $x \equiv a \pmod{n_1 n_2}$.

2.124 Definition The *multiplicative group* of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$. In particular, if n is a prime, then $\mathbb{Z}_n^* = \{a \mid 1 \leq a \leq n-1\}$.

2.125 Definition The *order* of $\mathbb{Z}_n^*$ is defined to be the number of elements in $\mathbb{Z}_n^*$, namely $|\mathbb{Z}_n^*|$.

It follows from the definition of the Euler phi function (Definition 2.100) that $|\mathbb{Z}_n^*| = \phi(n)$. Note also that if $a \in \mathbb{Z}_n^*$ and $b \in \mathbb{Z}_n^*$, then $a \cdot b \in \mathbb{Z}_n^*$, and so $\mathbb{Z}_n^*$ is closed under multiplication.

2.126 Fact Let $n \geq 2$ be an integer.

(i) (*Euler's theorem*) If $a \in \mathbb{Z}_n^*$, then $a^{\phi(n)} \equiv 1 \pmod n$.

(ii) If n is a product of distinct primes, and if $r \equiv s \pmod{\phi(n)}$, then $a^r \equiv a^s \pmod n$ for all integers a. In other words, when working modulo such an n, exponents can be reduced modulo $\phi(n)$.

A special case of Euler's theorem is Fermat's (little) theorem.

2.127 Fact Let p be a prime.

(i) (*Fermat's theorem*) If $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod p$.

(ii) If $r \equiv s \pmod{p-1}$, then $a^r \equiv a^s \pmod p$ for all integers a. In other words, when working modulo a prime p, exponents can be reduced modulo $p - 1$.

(iii) In particular, $a^p \equiv a \pmod p$ for all integers a.

2.128 Definition Let $a \in \mathbb{Z}_n^*$. The *order* of a, denoted $\operatorname{ord}(a)$, is the least positive integer t such that $a^t \equiv 1 \pmod n$.

2.129 Fact If the order of $a \in \mathbb{Z}_n^*$ is t, and $a^s \equiv 1 \pmod n$, then t divides s. In particular, $t \mid \phi(n)$.

2.130 Example Let $n = 21$. Then $\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$. Note that $\phi(21) = \phi(7)\phi(3) = 12 = |\mathbb{Z}_{21}^*|$. The orders of elements in $\mathbb{Z}_{21}^*$ are listed in Table 2.3. □

$a \in \mathbb{Z}_{21}^*$	1	2	4	5	8	10	11	13	16	17	19	20
order of a	1	6	3	6	2	6	6	2	3	6	6	2

Table 2.3: *Orders of elements in* $\mathbb{Z}_{21}^*$.

2.131 Definition Let $\alpha \in \mathbb{Z}_n^*$. If the order of α is $\phi(n)$, then α is said to be a *generator* or a *primitive element* of $\mathbb{Z}_n^*$. If $\mathbb{Z}_n^*$ has a generator, then $\mathbb{Z}_n^*$ is said to be *cyclic*.

2.132 Fact (*properties of generators of* $\mathbb{Z}_n^*$)

(i) $\mathbb{Z}_n^*$ has a generator if and only if $n = 2, 4, p^k$ or $2p^k$, where p is an odd prime and $k \geq 1$. In particular, if p is a prime, then $\mathbb{Z}_p^*$ has a generator.

(ii) If α is a generator of $\mathbb{Z}_n^*$, then $\mathbb{Z}_n^* = \{\alpha^i \bmod n \mid 0 \leq i \leq \phi(n) - 1\}$.

(iii) Suppose that α is a generator of $\mathbb{Z}_n^*$. Then $b = \alpha^i \bmod n$ is also a generator of $\mathbb{Z}_n^*$ if and only if $\gcd(i, \phi(n)) = 1$. It follows that if $\mathbb{Z}_n^*$ is cyclic, then the number of generators is $\phi(\phi(n))$.

(iv) $\alpha \in \mathbb{Z}_n^*$ is a generator of $\mathbb{Z}_n^*$ if and only if $\alpha^{\phi(n)/p} \not\equiv 1 \pmod{n}$ for each prime divisor p of $\phi(n)$.

2.133 Example $\mathbb{Z}_{21}^*$ is not cyclic since it does not contain an element of order $\phi(21) = 12$ (see Table 2.3); note that 21 does not satisfy the condition of Fact 2.132(i). On the other hand, $\mathbb{Z}_{25}^*$ is cyclic, and has a generator $\alpha = 2$. □

2.134 Definition Let $a \in \mathbb{Z}_n^*$. a is said to be a *quadratic residue* modulo n, or a *square* modulo n, if there exists an $x \in \mathbb{Z}_n^*$ such that $x^2 \equiv a \pmod{n}$. If no such x exists, then a is called a *quadratic non-residue* modulo n. The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic non-residues is denoted by $\overline{Q}_n$.

Note that by definition $0 \notin \mathbb{Z}_n^*$, whence $0 \notin Q_n$ and $0 \notin \overline{Q}_n$.

2.135 Fact Let p be an odd prime and let α be a generator of $\mathbb{Z}_p^*$. Then $a \in \mathbb{Z}_p^*$ is a quadratic residue modulo p if and only if $a = \alpha^i \bmod p$, where i is an even integer. It follows that $|Q_p| = (p-1)/2$ and $|\overline{Q}_p| = (p-1)/2$; that is, half of the elements in $\mathbb{Z}_p^*$ are quadratic residues and the other half are quadratic non-residues.

2.136 Example $\alpha = 6$ is a generator of $\mathbb{Z}_{13}^*$. The powers of α are listed in the following table.

i	0	1	2	3	4	5	6	7	8	9	10	11
$\alpha^i \bmod 13$	1	6	10	8	9	2	12	7	3	5	4	11

Hence $Q_{13} = \{1, 3, 4, 9, 10, 12\}$ and $\overline{Q}_{13} = \{2, 5, 6, 7, 8, 11\}$. □

2.137 Fact Let n be a product of two distinct odd primes p and q, $n = pq$. Then $a \in \mathbb{Z}_n^*$ is a quadratic residue modulo n if and only if $a \in Q_p$ and $a \in Q_q$. It follows that $|Q_n| = |Q_p| \cdot |Q_q| = (p-1)(q-1)/4$ and $|\overline{Q}_n| = 3(p-1)(q-1)/4$.

2.138 Example Let $n = 21$. Then $Q_{21} = \{1, 4, 16\}$ and $\overline{Q}_{21} = \{2, 5, 8, 10, 11, 13, 17, 19, 20\}$. □

2.139 Definition Let $a \in Q_n$. If $x \in \mathbb{Z}_n^*$ satisfies $x^2 \equiv a \pmod{n}$, then x is called a *square root* of a modulo n.

2.140 Fact (*number of square roots*)

(i) If p is an odd prime and $a \in Q_p$, then a has exactly two square roots modulo p.

(ii) More generally, let $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ where the p_i are distinct odd primes and $e_i \geq 1$. If $a \in Q_n$, then a has precisely 2^k distinct square roots modulo n.

2.141 Example The square roots of 12 modulo 37 are 7 and 30. The square roots of 121 modulo 315 are 11, 74, 101, 151, 164, 214, 241, and 304. □

2.4.4 Algorithms in $\mathbb{Z}_n$

Let n be a positive integer. As before, the elements of $\mathbb{Z}_n$ will be represented by the integers $\{0, 1, 2, \ldots, n-1\}$.

Observe that if $a, b \in \mathbb{Z}_n$, then

$$(a+b) \bmod n = \begin{cases} a+b, & \text{if } a+b < n, \\ a+b-n, & \text{if } a+b \geq n. \end{cases}$$

Hence modular addition (and subtraction) can be performed without the need of a long division. Modular multiplication of a and b may be accomplished by simply multiplying a and b as integers, and then taking the remainder of the result after division by n. Inverses in $\mathbb{Z}_n$ can be computed using the extended Euclidean algorithm as next described.

2.142 Algorithm Computing multiplicative inverses in $\mathbb{Z}_n$

INPUT: $a \in \mathbb{Z}_n$.
OUTPUT: $a^{-1} \bmod n$, provided that it exists.

1. Use the extended Euclidean algorithm (Algorithm 2.107) to find integers x and y such that $ax + ny = d$, where $d = \gcd(a, n)$.
2. If $d > 1$, then $a^{-1} \bmod n$ does not exist. Otherwise, return(x).

Modular exponentiation can be performed efficiently with the repeated square-and-multiply algorithm (Algorithm 2.143), which is crucial for many cryptographic protocols. One version of this algorithm is based on the following observation. Let the binary representation of k be $\sum_{i=0}^{t} k_i 2^i$, where each $k_i \in \{0, 1\}$. Then

$$a^k = \prod_{i=0}^{t} a^{k_i 2^i} = (a^{2^0})^{k_0} (a^{2^1})^{k_1} \cdots (a^{2^t})^{k_t}.$$

2.143 Algorithm Repeated square-and-multiply algorithm for exponentiation in $\mathbb{Z}_n$

INPUT: $a \in \mathbb{Z}_n$, and integer $0 \leq k < n$ whose binary representation is $k = \sum_{i=0}^{t} k_i 2^i$.
OUTPUT: $a^k \bmod n$.

1. Set $b \leftarrow 1$. If $k = 0$ then return(b).
2. Set $A \leftarrow a$.
3. If $k_0 = 1$ then set $b \leftarrow a$.
4. For i from 1 to t do the following:
   - 4.1 Set $A \leftarrow A^2 \bmod n$.
   - 4.2 If $k_i = 1$ then set $b \leftarrow A \cdot b \bmod n$.
5. Return(b).

2.144 Example (*modular exponentiation*) Table 2.4 shows the steps involved in the computation of $5^{596} \bmod 1234 = 1013$. □

The number of bit operations for the basic operations in $\mathbb{Z}_n$ is summarized in Table 2.5. Efficient algorithms for performing modular multiplication and exponentiation are further examined in §14.3 and §14.6.

i	0	1	2	3	4	5	6	7	8	9
k_i	0	0	1	0	1	0	1	0	0	1
A	5	25	625	681	1011	369	421	779	947	925
b	1	1	625	625	67	67	1059	1059	1059	1013

Table 2.4: *Computation of* $5^{596} \bmod 1234$.

Operation		Bit complexity
Modular addition	$(a+b) \bmod n$	$O(\lg n)$
Modular subtraction	$(a-b) \bmod n$	$O(\lg n)$
Modular multiplication	$(a \cdot b) \bmod n$	$O((\lg n)^2)$
Modular inversion	$a^{-1} \bmod n$	$O((\lg n)^2)$
Modular exponentiation	$a^k \bmod n, k < n$	$O((\lg n)^3)$

Table 2.5: *Bit complexity of basic operations in* $\mathbb{Z}_n$.

2.4.5 The Legendre and Jacobi symbols

The Legendre symbol is a useful tool for keeping track of whether or not an integer a is a quadratic residue modulo a prime p.

2.145 Definition Let p be an odd prime and a an integer. The *Legendre symbol* $\left(\frac{a}{p}\right)$ is defined to be

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p \mid a, \\ 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \overline{Q}_p. \end{cases}$$

2.146 Fact (*properties of Legendre symbol*) Let p be an odd prime and $a, b \in \mathbb{Z}$. Then the Legendre symbol has the following properties:

(i) $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p$. In particular, $\left(\frac{1}{p}\right) = 1$ and $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$. Hence $-1 \in Q_p$ if $p \equiv 1 \pmod 4$, and $-1 \in \overline{Q}_p$ if $p \equiv 3 \pmod 4$.

(ii) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. Hence if $a \in \mathbb{Z}_p^*$, then $\left(\frac{a^2}{p}\right) = 1$.

(iii) If $a \equiv b \pmod p$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.

(iv) $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$. Hence $\left(\frac{2}{p}\right) = 1$ if $p \equiv 1$ or $7 \pmod 8$, and $\left(\frac{2}{p}\right) = -1$ if $p \equiv 3$ or $5 \pmod 8$.

(v) (*law of quadratic reciprocity*) If q is an odd prime distinct from p, then

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)(-1)^{(p-1)(q-1)/4}.$$

In other words, $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$ unless both p and q are congruent to 3 modulo 4, in which case $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$.

The Jacobi symbol is a generalization of the Legendre symbol to integers n which are odd but not necessarily prime.

2.147 Definition Let $n \geq 3$ be odd with prime factorization $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. Then the *Jacobi symbol* $\left(\frac{a}{n}\right)$ is defined to be

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}.$$

Observe that if n is prime, then the Jacobi symbol is just the Legendre symbol.

2.148 Fact (*properties of Jacobi symbol*) Let $m \geq 3$, $n \geq 3$ be odd integers, and $a, b \in \mathbb{Z}$. Then the Jacobi symbol has the following properties:

(i) $\left(\frac{a}{n}\right) = 0, 1,$ or -1. Moreover, $\left(\frac{a}{n}\right) = 0$ if and only if $\gcd(a, n) \neq 1$.
(ii) $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$. Hence if $a \in \mathbb{Z}_n^*$, then $\left(\frac{a^2}{n}\right) = 1$.
(iii) $\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{a}{n}\right)$.
(iv) If $a \equiv b \pmod{n}$, then $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.
(v) $\left(\frac{1}{n}\right) = 1$.
(vi) $\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}$. Hence $\left(\frac{-1}{n}\right) = 1$ if $n \equiv 1 \pmod{4}$, and $\left(\frac{-1}{n}\right) = -1$ if $n \equiv 3 \pmod{4}$.
(vii) $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$. Hence $\left(\frac{2}{n}\right) = 1$ if $n \equiv 1$ or $7 \pmod{8}$, and $\left(\frac{2}{n}\right) = -1$ if $n \equiv 3$ or $5 \pmod{8}$.
(viii) $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)(-1)^{(m-1)(n-1)/4}$. In other words, $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)$ unless both m and n are congruent to 3 modulo 4, in which case $\left(\frac{m}{n}\right) = -\left(\frac{n}{m}\right)$.

By properties of the Jacobi symbol it follows that if n is odd and $a = 2^e a_1$ where a_1 is odd, then

$$\left(\frac{a}{n}\right) = \left(\frac{2^e}{n}\right)\left(\frac{a_1}{n}\right) = \left(\frac{2}{n}\right)^e \left(\frac{n \bmod a_1}{a_1}\right)(-1)^{(a_1-1)(n-1)/4}.$$

This observation yields the following recursive algorithm for computing $\left(\frac{a}{n}\right)$, which does not require the prime factorization of n.

2.149 Algorithm Jacobi symbol (and Legendre symbol) computation

JACOBI(a,n)
INPUT: an odd integer $n \geq 3$, and an integer a, $0 \leq a < n$.
OUTPUT: the Jacobi symbol $\left(\frac{a}{n}\right)$ (and hence the Legendre symbol when n is prime).

1. If $a = 0$ then return(0).
2. If $a = 1$ then return(1).
3. Write $a = 2^e a_1$, where a_1 is odd.
4. If e is even then set $s \leftarrow 1$. Otherwise set $s \leftarrow 1$ if $n \equiv 1$ or $7 \pmod{8}$, or set $s \leftarrow -1$ if $n \equiv 3$ or $5 \pmod{8}$.
5. If $n \equiv 3 \pmod{4}$ and $a_1 \equiv 3 \pmod{4}$ then set $s \leftarrow -s$.
6. Set $n_1 \leftarrow n \bmod a_1$.
7. If $a_1 = 1$ then return(s); otherwise return($s \cdot$ JACOBI(n_1,a_1)).

2.150 Fact Algorithm 2.149 has a running time of $O((\lg n)^2)$ bit operations.

2.151 Remark (*finding quadratic non-residues modulo a prime p*) Let p denote an odd prime. Even though it is known that half of the elements in $\mathbb{Z}_p^*$ are quadratic non-residues modulo p (see Fact 2.135), there is no *deterministic* polynomial-time algorithm known for finding one. A *randomized* algorithm for finding a quadratic non-residue is to simply select random integers $a \in \mathbb{Z}_p^*$ until one is found satisfying $\left(\frac{a}{p}\right) = -1$. The expected number iterations before a non-residue is found is 2, and hence the procedure takes expected polynomial-time.

2.152 Example (*Jacobi symbol computation*) For $a = 158$ and $n = 235$, Algorithm 2.149 computes the Jacobi symbol $\left(\frac{158}{235}\right)$ as follows:

$$\begin{aligned}\left(\frac{158}{235}\right) &= \left(\frac{2}{235}\right)\left(\frac{79}{235}\right) = (-1)\left(\frac{235}{79}\right)(-1)^{78\cdot 234/4} = \left(\frac{77}{79}\right) \\ &= \left(\frac{79}{77}\right)(-1)^{76\cdot 78/4} = \left(\frac{2}{77}\right) = -1.\end{aligned}$$ □

Unlike the Legendre symbol, the Jacobi symbol $\left(\frac{a}{n}\right)$ does not reveal whether or not a is a quadratic residue modulo n. It is indeed true that if $a \in Q_n$, then $\left(\frac{a}{n}\right) = 1$. However, $\left(\frac{a}{n}\right) = 1$ does not imply that $a \in Q_n$.

2.153 Example (*quadratic residues and non-residues*) Table 2.6 lists the elements in $\mathbb{Z}_{21}^*$ and their Jacobi symbols. Recall from Example 2.138 that $Q_{21} = \{1, 4, 16\}$. Observe that $\left(\frac{5}{21}\right) = 1$ but $5 \notin Q_{21}$. □

$a \in \mathbb{Z}_{21}^*$	1	2	4	5	8	10	11	13	16	17	19	20
$a^2 \bmod n$	1	4	16	4	1	16	16	1	4	16	4	1
$\left(\frac{a}{3}\right)$	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1
$\left(\frac{a}{7}\right)$	1	1	1	-1	1	-1	1	-1	1	-1	-1	-1
$\left(\frac{a}{21}\right)$	1	-1	1	1	-1	-1	-1	-1	1	1	-1	1

Table 2.6: *Jacobi symbols of elements in $\mathbb{Z}_{21}^*$.*

2.154 Definition Let $n \geq 3$ be an odd integer, and let $J_n = \{a \in \mathbb{Z}_n^* \mid \left(\frac{a}{n}\right) = 1\}$. The set of *pseudosquares* modulo n, denoted $\widetilde{Q}_n$, is defined to be the set $J_n - Q_n$.

2.155 Fact Let $n = pq$ be a product of two distinct odd primes. Then $|Q_n| = |\widetilde{Q}_n| = (p-1)(q-1)/4$; that is, half of the elements in J_n are quadratic residues and the other half are pseudosquares.

2.4.6 Blum integers

2.156 Definition A *Blum integer* is a composite integer of the form $n = pq$, where p and q are distinct primes each congruent to 3 modulo 4.

2.157 Fact Let $n = pq$ be a Blum integer, and let $a \in Q_n$. Then a has precisely four square roots modulo n, exactly one of which is also in Q_n.

2.158 Definition Let n be a Blum integer and let $a \in Q_n$. The unique square root of a in Q_n is called the *principal square root* of a modulo n.

2.159 Example (*Blum integer*) For the Blum integer $n = 21$, $J_n = \{1, 4, 5, 16, 17, 20\}$ and $\widetilde{Q}_n = \{5, 17, 20\}$. The four square roots of $a = 4$ are 2, 5, 16, and 19, of which only 16 is also in Q_{21}. Thus 16 is the principal square root of 4 modulo 21. □

2.160 Fact If $n = pq$ is a Blum integer, then the function $f : Q_n \longrightarrow Q_n$ defined by $f(x) = x^2 \bmod n$ is a permutation. The inverse function of f is:

$$f^{-1}(x) = x^{((p-1)(q-1)+4)/8} \bmod n.$$

2.5 Abstract algebra

This section provides an overview of basic algebraic objects and their properties, for reference in the remainder of this handbook. Several of the definitions in §2.5.1 and §2.5.2 were presented earlier in §2.4.3 in the more concrete setting of the algebraic structure $\mathbb{Z}_n^*$.

2.161 Definition A *binary operation* $*$ on a set S is a mapping from $S \times S$ to S. That is, $*$ is a rule which assigns to each ordered pair of elements from S an element of S.

2.5.1 Groups

2.162 Definition A *group* $(G, *)$ consists of a set G with a binary operation $*$ on G satisfying the following three axioms.

(i) The group operation is *associative*. That is, $a * (b * c) = (a * b) * c$ for all $a, b, c \in G$.
(ii) There is an element $1 \in G$, called the *identity element*, such that $a * 1 = 1 * a = a$ for all $a \in G$.
(iii) For each $a \in G$ there exists an element $a^{-1} \in G$, called the *inverse* of a, such that $a * a^{-1} = a^{-1} * a = 1$.

A group G is *abelian* (or *commutative*) if, furthermore,

(iv) $a * b = b * a$ for all $a, b \in G$.

Note that multiplicative group notation has been used for the group operation. If the group operation is addition, then the group is said to be an *additive* group, the identity element is denoted by 0, and the inverse of a is denoted $-a$.

Henceforth, unless otherwise stated, the symbol $*$ will be omitted and the group operation will simply be denoted by juxtaposition.

2.163 Definition A group G is *finite* if $|G|$ is finite. The number of elements in a finite group is called its *order*.

2.164 Example The set of integers $\mathbb{Z}$ with the operation of addition forms a group. The identity element is 0 and the inverse of an integer a is the integer $-a$. □

2.165 Example The set $\mathbb{Z}_n$, with the operation of addition modulo n, forms a group of order n. The set $\mathbb{Z}_n$ with the operation of multiplication modulo n is not a group, since not all elements have multiplicative inverses. However, the set $\mathbb{Z}_n^*$ (see Definition 2.124) is a group of order $\phi(n)$ under the operation of multiplication modulo n, with identity element 1. □

2.166 Definition A non-empty subset H of a group G is a *subgroup* of G if H is itself a group with respect to the operation of G. If H is a subgroup of G and $H \neq G$, then H is called a *proper* subgroup of G.

2.167 Definition A group G is *cyclic* if there is an element $\alpha \in G$ such that for each $b \in G$ there is an integer i with $b = \alpha^i$. Such an element α is called a *generator* of G.

2.168 Fact If G is a group and $a \in G$, then the set of all powers of a forms a cyclic subgroup of G, called the subgroup *generated by* a, and denoted by $\langle a \rangle$.

2.169 Definition Let G be a group and $a \in G$. The *order* of a is defined to be the least positive integer t such that $a^t = 1$, provided that such an integer exists. If such a t does not exist, then the order of a is defined to be ∞.

2.170 Fact Let G be a group, and let $a \in G$ be an element of finite order t. Then $|\langle a \rangle|$, the size of the subgroup generated by a, is equal to t.

2.171 Fact (*Lagrange's theorem*) If G is a finite group and H is a subgroup of G, then $|H|$ divides $|G|$. Hence, if $a \in G$, the order of a divides $|G|$.

2.172 Fact Every subgroup of a cyclic group G is also cyclic. In fact, if G is a cyclic group of order n, then for each positive divisor d of n, G contains exactly one subgroup of order d.

2.173 Fact Let G be a group.

(i) If the order of $a \in G$ is t, then the order of a^k is $t/\gcd(t,k)$.

(ii) If G is a cyclic group of order n and $d|n$, then G has exactly $\phi(d)$ elements of order d. In particular, G has $\phi(n)$ generators.

2.174 Example Consider the multiplicative group $\mathbb{Z}_{19}^* = \{1, 2, \ldots, 18\}$ of order 18. The group is cyclic (Fact 2.132(i)), and a generator is $\alpha = 2$. The subgroups of $\mathbb{Z}_{19}^*$, and their generators, are listed in Table 2.7. □

Subgroup	Generators	Order
$\{1\}$	1	1
$\{1, 18\}$	18	2
$\{1, 7, 11\}$	$7, 11$	3
$\{1, 7, 8, 11, 12, 18\}$	$8, 12$	6
$\{1, 4, 5, 6, 7, 9, 11, 16, 17\}$	$4, 5, 6, 9, 16, 17$	9
$\{1, 2, 3, \ldots, 18\}$	$2, 3, 10, 13, 14, 15$	18

Table 2.7: *The subgroups of* $\mathbb{Z}_{19}^*$.

2.5.2 Rings

2.175 Definition A *ring* $(R, +, \times)$ consists of a set R with two binary operations arbitrarily denoted $+$ (addition) and $\times$ (multiplication) on R, satisfying the following axioms.

(i) $(R, +)$ is an abelian group with identity denoted 0.

(ii) The operation $\times$ is associative. That is, $a \times (b \times c) = (a \times b) \times c$ for all $a, b, c \in R$.
(iii) There is a multiplicative identity denoted 1, with $1 \neq 0$, such that $1 \times a = a \times 1 = a$ for all $a \in R$.
(iv) The operation $\times$ is *distributive* over $+$. That is, $a \times (b + c) = (a \times b) + (a \times c)$ and $(b + c) \times a = (b \times a) + (c \times a)$ for all $a, b, c \in R$.

The ring is a *commutative ring* if $a \times b = b \times a$ for all $a, b \in R$.

2.176 Example The set of integers $\mathbb{Z}$ with the usual operations of addition and multiplication is a commutative ring. □

2.177 Example The set $\mathbb{Z}_n$ with addition and multiplication performed modulo n is a commutative ring. □

2.178 Definition An element a of a ring R is called a *unit* or an *invertible element* if there is an element $b \in R$ such that $a \times b = 1$.

2.179 Fact The set of units in a ring R forms a group under multiplication, called the *group of units* of R.

2.180 Example The group of units of the ring $\mathbb{Z}_n$ is $\mathbb{Z}_n^*$ (see Definition 2.124). □

2.5.3 Fields

2.181 Definition A *field* is a commutative ring in which all non-zero elements have multiplicative inverses.

2.182 Definition The *characteristic* of a field is 0 if $\overbrace{1 + 1 + \cdots + 1}^{m \text{ times}}$ is never equal to 0 for any $m \geq 1$. Otherwise, the characteristic of the field is the least positive integer m such that $\sum_{i=1}^{m} 1$ equals 0.

2.183 Example The set of integers under the usual operations of addition and multiplication is not a field, since the only non-zero integers with multiplicative inverses are 1 and -1. However, the rational numbers $\mathbb{Q}$, the real numbers $\mathbb{R}$, and the complex numbers $\mathbb{C}$ form fields of characteristic 0 under the usual operations. □

2.184 Fact $\mathbb{Z}_n$ is a field (under the usual operations of addition and multiplication modulo n) if and only if n is a prime number. If n is prime, then $\mathbb{Z}_n$ has characteristic n.

2.185 Fact If the characteristic m of a field is not 0, then m is a prime number.

2.186 Definition A subset F of a field E is a *subfield* of E if F is itself a field with respect to the operations of E. If this is the case, E is said to be an *extension field* of F.

2.5.4 Polynomial rings

2.187 Definition If R is a commutative ring, then a *polynomial* in the indeterminate x over the ring R is an expression of the form

$$f(x) = a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0$$

where each $a_i \in R$ and $n \geq 0$. The element a_i is called the *coefficient* of x^i in $f(x)$. The largest integer m for which $a_m \neq 0$ is called the *degree* of $f(x)$, denoted $\deg f(x)$; a_m is called the *leading coefficient* of $f(x)$. If $f(x) = a_0$ (a *constant polynomial*) and $a_0 \neq 0$, then $f(x)$ has degree 0. If all the coefficients of $f(x)$ are 0, then $f(x)$ is called the *zero polynomial* and its degree, for mathematical convenience, is defined to be $-\infty$. The polynomial $f(x)$ is said to be *monic* if its leading coefficient is equal to 1.

2.188 Definition If R is a commutative ring, the *polynomial ring* $R[x]$ is the ring formed by the set of all polynomials in the indeterminate x having coefficients from R. The two operations are the standard polynomial addition and multiplication, with coefficient arithmetic performed in the ring R.

2.189 Example (*polynomial ring*) Let $f(x) = x^3 + x + 1$ and $g(x) = x^2 + x$ be elements of the polynomial ring $\mathbb{Z}_2[x]$. Working in $\mathbb{Z}_2[x]$,

$$f(x) + g(x) = x^3 + x^2 + 1$$

and

$$f(x) \cdot g(x) = x^5 + x^4 + x^3 + x. \qquad \square$$

For the remainder of this section, F will denote an arbitrary field. The polynomial ring $F[x]$ has many properties in common with the integers (more precisely, $F[x]$ and $\mathbb{Z}$ are both *Euclidean domains*, however, this generalization will not be pursued here). These similarities are investigated further.

2.190 Definition Let $f(x) \in F[x]$ be a polynomial of degree at least 1. Then $f(x)$ is said to be *irreducible over* F if it cannot be written as the product of two polynomials in $F[x]$, each of positive degree.

2.191 Definition (*division algorithm for polynomials*) If $g(x), h(x) \in F[x]$, with $h(x) \neq 0$, then ordinary polynomial long division of $g(x)$ by $h(x)$ yields polynomials $q(x)$ and $r(x) \in F[x]$ such that

$$g(x) = q(x)h(x) + r(x), \text{ where } \deg r(x) < \deg h(x).$$

Moreover, $q(x)$ and $r(x)$ are unique. The polynomial $q(x)$ is called the *quotient*, while $r(x)$ is called the *remainder*. The remainder of the division is sometimes denoted $g(x) \bmod h(x)$, and the quotient is sometimes denoted $g(x) \operatorname{div} h(x)$ (cf. Definition 2.82).

2.192 Example (*polynomial division*) Consider the polynomials $g(x) = x^6+x^5+x^3+x^2+x+1$ and $h(x) = x^4 + x^3 + 1$ in $\mathbb{Z}_2[x]$. Polynomial long division of $g(x)$ by $h(x)$ yields

$$g(x) = x^2 h(x) + (x^3 + x + 1).$$

Hence $g(x) \bmod h(x) = x^3 + x + 1$ and $g(x) \operatorname{div} h(x) = x^2$. $\square$

2.193 Definition If $g(x), h(x) \in F[x]$ then $h(x)$ *divides* $g(x)$, written $h(x)|g(x)$, if $g(x) \bmod h(x) = 0$.

Let $f(x)$ be a fixed polynomial in $F[x]$. As with the integers (Definition 2.110), one can define congruences of polynomials in $F[x]$ based on division by $f(x)$.

2.194 Definition If $g(x), h(x) \in F[x]$, then $g(x)$ is said to be *congruent to* $h(x)$ *modulo* $f(x)$ if $f(x)$ divides $g(x) - h(x)$. This is denoted by $g(x) \equiv h(x) \pmod{f(x)}$.

2.195 Fact (*properties of congruences*) For all $g(x), h(x), g_1(x), h_1(x), s(x) \in F[x]$, the following are true.

(i) $g(x) \equiv h(x) \pmod{f(x)}$ if and only if $g(x)$ and $h(x)$ leave the same remainder upon division by $f(x)$.
(ii) (*reflexivity*) $g(x) \equiv g(x) \pmod{f(x)}$.
(iii) (*symmetry*) If $g(x) \equiv h(x) \pmod{f(x)}$, then $h(x) \equiv g(x) \pmod{f(x)}$.
(iv) (*transitivity*) If $g(x) \equiv h(x) \pmod{f(x)}$ and $h(x) \equiv s(x) \pmod{f(x)}$, then $g(x) \equiv s(x) \pmod{f(x)}$.
(v) If $g(x) \equiv g_1(x) \pmod{f(x)}$ and $h(x) \equiv h_1(x) \pmod{f(x)}$, then $g(x) + h(x) \equiv g_1(x) + h_1(x) \pmod{f(x)}$ and $g(x)h(x) \equiv g_1(x)h_1(x) \pmod{f(x)}$.

Let $f(x)$ be a fixed polynomial in $F[x]$. The *equivalence class* of a polynomial $g(x) \in F[x]$ is the set of all polynomials in $F[x]$ congruent to $g(x)$ modulo $f(x)$. From properties (ii), (iii), and (iv) above, it can be seen that the relation of congruence modulo $f(x)$ partitions $F[x]$ into equivalence classes. If $g(x) \in F[x]$, then long division by $f(x)$ yields unique polynomials $q(x), r(x) \in F[x]$ such that $g(x) = q(x)f(x) + r(x)$, where $\deg r(x) < \deg f(x)$. Hence every polynomial $g(x)$ is congruent modulo $f(x)$ to a unique polynomial of degree less than $\deg f(x)$. The polynomial $r(x)$ will be used as representative of the equivalence class of polynomials containing $g(x)$.

2.196 Definition $F[x]/(f(x))$ denotes the set of (equivalence classes of) polynomials in $F[x]$ of degree less than $n = \deg f(x)$. Addition and multiplication are performed modulo $f(x)$.

2.197 Fact $F[x]/(f(x))$ is a commutative ring.

2.198 Fact If $f(x)$ is irreducible over F, then $F[x]/(f(x))$ is a field.

2.5.5 Vector spaces

2.199 Definition A *vector space* V over a field F is an abelian group $(V, +)$, together with a multiplication operation $\bullet : F \times V \longrightarrow V$ (usually denoted by juxtaposition) such that for all $a, b \in F$ and $v, w \in V$, the following axioms are satisfied.

(i) $a(v + w) = av + aw$.
(ii) $(a + b)v = av + bv$.
(iii) $(ab)v = a(bv)$.
(iv) $1v = v$.

The elements of V are called *vectors*, while the elements of F are called *scalars*. The group operation $+$ is called *vector addition*, while the multiplication operation is called *scalar multiplication*.

2.200 Definition Let V be a vector space over a field F. A *subspace* of V is an additive subgroup U of V which is closed under scalar multiplication, i.e., $av \in U$ for all $a \in F$ and $v \in U$.

2.201 Fact A subspace of a vector space is also a vector space.

2.202 Definition Let $S = \{v_1, v_2, \ldots, v_n\}$ be a finite subset of a vector space V over a field F.

(i) A *linear combination* of S is an expression of the form $a_1v_1 + a_2v_2 + \cdots + a_nv_n$, where each $a_i \in F$.
(ii) The *span* of S, denoted $\langle S\rangle$, is the set of all linear combinations of S. The span of S is a subspace of V.
(iii) If U is a subspace of V, then S is said to *span* U if $\langle S\rangle = U$.
(iv) The set S is *linearly dependent* over F if there exist scalars $a_1, a_2, \ldots, a_n$, not all zero, such that $a_1v_1 + a_2v_2 + \cdots + a_nv_n = 0$. If no such scalars exist, then S is *linearly independent* over F.
(v) A linearly independent set of vectors that spans V is called a *basis* for V.

2.203 Fact Let V be a vector space.

(i) If V has a finite spanning set, then it has a basis.
(ii) If V has a basis, then in fact all bases have the same number of elements.

2.204 Definition If a vector space V has a basis, then the number of elements in a basis is called the *dimension* of V, denoted $\dim V$.

2.205 Example If F is any field, then the n-fold Cartesian product $V = F \times F \times \cdots \times F$ is a vector space over F of dimension n. The *standard basis* for V is $\{e_1, e_2, \ldots, e_n\}$, where e_i is a vector with a 1 in the i^{th} coordinate and 0's elsewhere. □

2.206 Definition Let E be an extension field of F. Then E can be viewed as a vector space over the subfield F, where vector addition and scalar multiplication are simply the field operations of addition and multiplication in E. The dimension of this vector space is called the *degree* of E over F, and denoted by $[E : F]$. If this degree is finite, then E is called a *finite extension* of F.

2.207 Fact Let F, E, and L be fields. If L is a finite extension of E and E is a finite extension of F, then L is also a finite extension of F and

$$[L : F] = [L : E][E : F].$$

2.6 Finite fields

2.6.1 Basic properties

2.208 Definition A *finite field* is a field F which contains a finite number of elements. The *order* of F is the number of elements in F.

2.209 Fact (*existence and uniqueness of finite fields*)

(i) If F is a finite field, then F contains p^m elements for some prime p and integer $m \geq 1$.

(ii) For every prime power order p^m, there is a unique (up to isomorphism) finite field of order p^m. This field is denoted by $\mathbb{F}_{p^m}$, or sometimes by $GF(p^m)$.

Informally speaking, two fields are *isomorphic* if they are structurally the same, although the representation of their field elements may be different. Note that if p is a prime then $\mathbb{Z}_p$ is a field, and hence every field of order p is isomorphic to $\mathbb{Z}_p$. Unless otherwise stated, the finite field $\mathbb{F}_p$ will henceforth be identified with $\mathbb{Z}_p$.

2.210 Fact If $\mathbb{F}_q$ is a finite field of order $q = p^m$, p a prime, then the characteristic of $\mathbb{F}_q$ is p. Moreover, $\mathbb{F}_q$ contains a copy of $\mathbb{Z}_p$ as a subfield. Hence $\mathbb{F}_q$ can be viewed as an extension field of $\mathbb{Z}_p$ of degree m.

2.211 Fact (*subfields of a finite field*) Let $\mathbb{F}_q$ be a finite field of order $q = p^m$. Then every subfield of $\mathbb{F}_q$ has order p^n, for some n that is a positive divisor of m. Conversely, if n is a positive divisor of m, then there is exactly one subfield of $\mathbb{F}_q$ of order p^n; an element $a \in \mathbb{F}_q$ is in the subfield $\mathbb{F}_{p^n}$ if and only if $a^{p^n} = a$.

2.212 Definition The non-zero elements of $\mathbb{F}_q$ form a group under multiplication called the *multiplicative group* of $\mathbb{F}_q$, denoted by $\mathbb{F}_q^*$.

2.213 Fact $\mathbb{F}_q^*$ is a cyclic group of order $q-1$. Hence $a^q = a$ for all $a \in \mathbb{F}_q$.

2.214 Definition A generator of the cyclic group $\mathbb{F}_q^*$ is called a *primitive element* or *generator* of $\mathbb{F}_q$.

2.215 Fact If $a, b \in \mathbb{F}_q$, a finite field of characteristic p, then

$$(a+b)^{p^t} = a^{p^t} + b^{p^t} \text{ for all } t \geq 0.$$

2.6.2 The Euclidean algorithm for polynomials

Let $\mathbb{Z}_p$ be the finite field of order p. The theory of greatest common divisors and the Euclidean algorithm for integers carries over in a straightforward manner to the polynomial ring $\mathbb{Z}_p[x]$ (and more generally to the polynomial ring $F[x]$, where F is any field).

2.216 Definition Let $g(x), h(x) \in \mathbb{Z}_p[x]$, where not both are 0. Then the *greatest common divisor* of $g(x)$ and $h(x)$, denoted $\gcd(g(x), h(x))$, is the monic polynomial of greatest degree in $\mathbb{Z}_p[x]$ which divides both $g(x)$ and $h(x)$. By definition, $\gcd(0,0) = 0$.

2.217 Fact $\mathbb{Z}_p[x]$ is a *unique factorization domain*. That is, every non-zero polynomial $f(x) \in \mathbb{Z}_p[x]$ has a factorization

$$f(x) = af_1(x)^{e_1} f_2(x)^{e_2} \cdots f_k(x)^{e_k},$$

where the $f_i(x)$ are distinct monic irreducible polynomials in $\mathbb{Z}_p[x]$, the e_i are positive integers, and $a \in \mathbb{Z}_p$. Furthermore, the factorization is unique up to rearrangement of factors.

The following is the polynomial version of the Euclidean algorithm (cf. Algorithm 2.104).

2.218 Algorithm Euclidean algorithm for $\mathbb{Z}_p[x]$

INPUT: two polynomials $g(x), h(x) \in \mathbb{Z}_p[x]$.
OUTPUT: the greatest common divisor of $g(x)$ and $h(x)$.

1. While $h(x) \neq 0$ do the following:
   1.1 Set $r(x) \leftarrow g(x) \bmod h(x)$, $g(x) \leftarrow h(x)$, $h(x) \leftarrow r(x)$.
2. Return($g(x)$).

2.219 Definition A $\mathbb{Z}_p$*-operation* means either an addition, subtraction, multiplication, inversion, or division in $\mathbb{Z}_p$.

2.220 Fact Suppose that $\deg g(x) \leq m$ and $\deg h(x) \leq m$. Then Algorithm 2.218 has a running time of $O(m^2)$ $\mathbb{Z}_p$-operations, or equivalently, $O(m^2(\lg p)^2)$ bit operations.

As with the case of the integers (cf. Algorithm 2.107), the Euclidean algorithm can be extended so that it also yields two polynomials $s(x)$ and $t(x)$ satisfying

$$s(x)g(x) + t(x)h(x) = \gcd(g(x), h(x)).$$

2.221 Algorithm Extended Euclidean algorithm for $\mathbb{Z}_p[x]$

INPUT: two polynomials $g(x), h(x) \in \mathbb{Z}_p[x]$.
OUTPUT: $d(x) = \gcd(g(x), h(x))$ and polynomials $s(x)$, $t(x) \in \mathbb{Z}_p[x]$ which satisfy $s(x)g(x) + t(x)h(x) = d(x)$.

1. If $h(x) = 0$ then set $d(x) \leftarrow g(x)$, $s(x) \leftarrow 1$, $t(x) \leftarrow 0$, and return($d(x)$,$s(x)$,$t(x)$).
2. Set $s_2(x) \leftarrow 1$, $s_1(x) \leftarrow 0$, $t_2(x) \leftarrow 0$, $t_1(x) \leftarrow 1$.
3. While $h(x) \neq 0$ do the following:
   3.1 $q(x) \leftarrow g(x) \operatorname{div} h(x)$, $r(x) \leftarrow g(x) - h(x)q(x)$.
   3.2 $s(x) \leftarrow s_2(x) - q(x)s_1(x)$, $t(x) \leftarrow t_2(x) - q(x)t_1(x)$.
   3.3 $g(x) \leftarrow h(x)$, $h(x) \leftarrow r(x)$.
   3.4 $s_2(x) \leftarrow s_1(x)$, $s_1(x) \leftarrow s(x)$, $t_2(x) \leftarrow t_1(x)$, and $t_1(x) \leftarrow t(x)$.
4. Set $d(x) \leftarrow g(x)$, $s(x) \leftarrow s_2(x)$, $t(x) \leftarrow t_2(x)$.
5. Return($d(x)$,$s(x)$,$t(x)$).

2.222 Fact (*running time of Algorithm 2.221*)

(i) The polynomials $s(x)$ and $t(x)$ given by Algorithm 2.221 have small degree; that is, they satisfy $\deg s(x) < \deg h(x)$ and $\deg t(x) < \deg g(x)$.
(ii) Suppose that $\deg g(x) \leq m$ and $\deg h(x) \leq m$. Then Algorithm 2.221 has a running time of $O(m^2)$ $\mathbb{Z}_p$-operations, or equivalently, $O(m^2(\lg p)^2)$ bit operations.

2.223 Example (*extended Euclidean algorithm for polynomials*) The following are the steps of Algorithm 2.221 with inputs $g(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + 1$ and $h(x) = x^9 + x^6 + x^5 + x^3 + x^2 + 1$ in $\mathbb{Z}_2[x]$.

Initialization
$s_2(x) \leftarrow 1$, $s_1(x) \leftarrow 0$, $t_2(x) \leftarrow 0$, $t_1(x) \leftarrow 1$.

Iteration 1

$q(x) \leftarrow x+1,\ r(x) \leftarrow x^8+x^7+x^6+x^2+x,$
$s(x) \leftarrow 1,\ t(x) \leftarrow x+1,$
$g(x) \leftarrow x^9+x^6+x^5+x^3+x^2+1,\ h(x) \leftarrow x^8+x^7+x^6+x^2+1,$
$s_2(x) \leftarrow 0,\ s_1(x) \leftarrow 1,\ t_2(x) \leftarrow 1,\ t_1(x) \leftarrow x+1.$

Iteration 2

$q(x) \leftarrow x+1,\ r(x) \leftarrow x^5+x^2+x+1,$
$s(x) \leftarrow x+1,\ t(x) \leftarrow x^2,$
$g(x) \leftarrow x^8+x^7+x^6+x^2+1,\ h(x) \leftarrow x^5+x^2+x+1,$
$s_2(x) \leftarrow 1,\ s_1(x) \leftarrow x+1,\ t_2(x) \leftarrow x+1,\ t_1(x) \leftarrow x^2.$

Iteration 3

$q(x) \leftarrow x^3+x^2+x+1,\ r(x) \leftarrow x^3+x+1,$
$s(x) \leftarrow x^4,\ t(x) \leftarrow x^5+x^4+x^3+x^2+x+1,$
$g(x) \leftarrow x^5+x^2+x+1,\ h(x) \leftarrow x^3+x+1,$
$s_2(x) \leftarrow x+1,\ s_1(x) \leftarrow x^4,\ t_2(x) \leftarrow x^2,\ t_1(x) \leftarrow x^5+x^4+x^3+x^2+x+1.$

Iteration 4

$q(x) \leftarrow x^2+1,\ r(x) \leftarrow 0,$
$s(x) \leftarrow x^6+x^4+x+1,\ t(x) \leftarrow x^7+x^6+x^2+x+1,$
$g(x) \leftarrow x^3+x+1,\ h(x) \leftarrow 0,$
$s_2(x) \leftarrow x^4,\ s_1(x) \leftarrow x^6+x^4+x+1,$
$t_2(x) \leftarrow x^5+x^4+x^3+x^2+x+1,\ t_1(x) \leftarrow x^7+x^6+x^2+x+1.$

Hence $\gcd(g(x),h(x)) = x^3+x+1$ and

$$(x^4)g(x) + (x^5+x^4+x^3+x^2+x+1)h(x) = x^3+x+1. \qquad \square$$

2.6.3 Arithmetic of polynomials

A commonly used representation for the elements of a finite field $\mathbb{F}_q$, where $q = p^m$ and p is a prime, is a *polynomial basis representation*. If $m = 1$, then $\mathbb{F}_q$ is just $\mathbb{Z}_p$ and arithmetic is performed modulo p. Since these operations have already been studied in Section 2.4.2, it is henceforth assumed that $m \geq 2$. The representation is based on Fact 2.198.

2.224 Fact Let $f(x) \in \mathbb{Z}_p[x]$ be an irreducible polynomial of degree m. Then $\mathbb{Z}_p[x]/(f(x))$ is a finite field of order p^m. Addition and multiplication of polynomials is performed modulo $f(x)$.

The following fact assures that all finite fields can be represented in this manner.

2.225 Fact For each $m \geq 1$, there exists a monic irreducible polynomial of degree m over $\mathbb{Z}_p$. Hence, every finite field has a polynomial basis representation.

An efficient algorithm for finding irreducible polynomials over finite fields is presented in §4.5.1. Tables 4.6 and 4.7 list some irreducible polynomials over the finite field $\mathbb{Z}_2$.

Henceforth, the elements of the finite field $\mathbb{F}_{p^m}$ will be represented by polynomials in $\mathbb{Z}_p[x]$ of degree $< m$. If $g(x), h(x) \in \mathbb{F}_{p^m}$, then addition is the usual addition of polynomials in $\mathbb{Z}_p[x]$. The product $g(x)h(x)$ can be formed by first multiplying $g(x)$ and $h(x)$ as polynomials by the ordinary method, and then taking the remainder after polynomial division by $f(x)$. Multiplicative inverses in $\mathbb{F}_{p^m}$ can be computed by using the extended Euclidean algorithm for the polynomial ring $\mathbb{Z}_p[x]$.

2.226 Algorithm Computing multiplicative inverses in $\mathbb{F}_{p^m}$

INPUT: a non-zero polynomial $g(x) \in \mathbb{F}_{p^m}$. (The elements of the field $\mathbb{F}_{p^m}$ are represented as $\mathbb{Z}_p[x]/(f(x))$, where $f(x) \in \mathbb{Z}_p[x]$ is an irreducible polynomial of degree m over $\mathbb{Z}_p$.)
OUTPUT: $g(x)^{-1} \in \mathbb{F}_{p^m}$.

1. Use the extended Euclidean algorithm for polynomials (Algorithm 2.221) to find two polynomials $s(x)$ and $t(x) \in \mathbb{Z}_p[x]$ such that $s(x)g(x) + t(x)f(x) = 1$.
2. Return($s(x)$).

Exponentiation in $\mathbb{F}_{p^m}$ can be done efficiently by the repeated square-and-multiply algorithm (cf. Algorithm 2.143).

2.227 Algorithm Repeated square-and-multiply algorithm for exponentiation in $\mathbb{F}_{p^m}$

INPUT: $g(x) \in \mathbb{F}_{p^m}$ and an integer $0 \le k < p^m - 1$ whose binary representation is $k = \sum_{i=0}^{t} k_i 2^i$. (The field $\mathbb{F}_{p^m}$ is represented as $\mathbb{Z}_p[x]/(f(x))$, where $f(x) \in \mathbb{Z}_p[x]$ is an irreducible polynomial of degree m over $\mathbb{Z}_p$.)
OUTPUT: $g(x)^k \bmod f(x)$.

1. Set $s(x)\leftarrow 1$. If $k = 0$ then return($s(x)$).
2. Set $G(x)\leftarrow g(x)$.
3. If $k_0 = 1$ then set $s(x)\leftarrow g(x)$.
4. For i from 1 to t do the following:
   - 4.1 Set $G(x)\leftarrow G(x)^2 \bmod f(x)$.
   - 4.2 If $k_i = 1$ then set $s(x)\leftarrow G(x) \cdot s(x) \bmod f(x)$.
5. Return($s(x)$).

The number of $\mathbb{Z}_p$-operations for the basic operations in $\mathbb{F}_{p^m}$ is summarized in Table 2.8.

Operation		Number of $\mathbb{Z}_p$-operations
Addition	$g(x) + h(x)$	$O(m)$
Subtraction	$g(x) - h(x)$	$O(m)$
Multiplication	$g(x) \cdot h(x)$	$O(m^2)$
Inversion	$g(x)^{-1}$	$O(m^2)$
Exponentiation	$g(x)^k$, $k < p^m$	$O((\lg p)m^3)$

Table 2.8: *Complexity of basic operations in $\mathbb{F}_{p^m}$.*

In some applications (cf. §4.5.3), it may be preferable to use a primitive polynomial to define a finite field.

2.228 Definition An irreducible polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree m is called a *primitive polynomial* if x is a generator of $\mathbb{F}_{p^m}^*$, the multiplicative group of all the non-zero elements in $\mathbb{F}_{p^m} = \mathbb{Z}_p[x]/(f(x))$.

2.229 Fact The irreducible polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree m is a primitive polynomial if and only if $f(x)$ divides $x^k - 1$ for $k = p^m - 1$ and for no smaller positive integer k.

2.230 Fact For each $m \geq 1$, there exists a monic primitive polynomial of degree m over $\mathbb{Z}_p$. In fact, there are precisely $\phi(p^m - 1)/m$ such polynomials.

2.231 Example (*the finite field* $\mathbb{F}_{2^4}$ *of order* 16) It can be verified (Algorithm 4.69) that the polynomial $f(x) = x^4 + x + 1$ is irreducible over $\mathbb{Z}_2$. Hence the finite field $\mathbb{F}_{2^4}$ can be represented as the set of all polynomials over $\mathbb{F}_2$ of degree less than 4. That is,

$$\mathbb{F}_{2^4} = \{a_3x^3 + a_2x^2 + a_1x + a_0 \mid a_i \in \{0, 1\}\}.$$

For convenience, the polynomial $a_3x^3 + a_2x^2 + a_1x + a_0$ is represented by the vector $(a_3a_2a_1a_0)$ of length 4, and

$$\mathbb{F}_{2^4} = \{(a_3a_2a_1a_0) \mid a_i \in \{0, 1\}\}.$$

The following are some examples of field arithmetic.

(i) Field elements are simply added componentwise: for example, $(1011) + (1001) = (0010)$.

(ii) To multiply the field elements (1101) and (1001), multiply them as polynomials and then take the remainder when this product is divided by $f(x)$:

$$\begin{aligned}(x^3 + x^2 + 1) \cdot (x^3 + 1) &= x^6 + x^5 + x^2 + 1 \\ &\equiv x^3 + x^2 + x + 1 \pmod{f(x)}.\end{aligned}$$

Hence $(1101) \cdot (1001) = (1111)$.

(iii) The multiplicative identity of $\mathbb{F}_{2^4}$ is (0001).

(iv) The inverse of (1011) is (0101). To verify this, observe that

$$\begin{aligned}(x^3 + x + 1) \cdot (x^2 + 1) &= x^5 + x^2 + x + 1 \\ &\equiv 1 \pmod{f(x)},\end{aligned}$$

whence $(1011) \cdot (0101) = (0001)$.

$f(x)$ is a primitive polynomial, or, equivalently, the field element $x = (0010)$ is a generator of $\mathbb{F}^*_{2^4}$. This may be checked by verifying that all the non-zero elements in $\mathbb{F}_{2^4}$ can be obtained as a powers of x. The computations are summarized in Table 2.9. □

A list of some primitive polynomials over finite fields of characteristic two is given in Table 4.8.

2.7 Notes and further references

§2.1

A classic introduction to probability theory is the first volume of the book by Feller [392]. The material on the birthday problem (§2.1.5) is summarized from Nishimura and Sibuya [931]. See also Girault, Cohen, and Campana [460]. The material on random mappings (§2.1.6) is summarized from the excellent article by Flajolet and Odlyzko [413].

§2.2

The concept of entropy was introduced in the seminal paper of Shannon [1120]. These ideas were then applied to develop a mathematical theory of secrecy systems by Shannon [1121]. Hellman [548] extended the Shannon theory approach to cryptography, and this work was further generalized by Beauchemin and Brassard [80]. For an introduction to information theory see the books by Welsh [1235] and Goldie and Pinch [464]. For more complete treatments, consult Blahut [144] and McEliece [829].

i	$x^i \bmod x^4+x+1$	vector notation
0	1	(0001)
1	x	(0010)
2	x^2	(0100)
3	x^3	(1000)
4	$x+1$	(0011)
5	x^2+x	(0110)
6	x^3+x^2	(1100)
7	x^3+x+1	(1011)
8	x^2+1	(0101)
9	x^3+x	(1010)
10	x^2+x+1	(0111)
11	x^3+x^2+x	(1110)
12	x^3+x^2+x+1	(1111)
13	x^3+x^2+1	(1101)
14	x^3+1	(1001)

Table 2.9: *The powers of x modulo $f(x) = x^4+x+1$.*

§2.3

Among the many introductory-level books on algorithms are those of Cormen, Leiserson, and Rivest [282], Rawlins [1030], and Sedgewick [1105]. A recent book on complexity theory is Papadimitriou [963]. Example 2.58 is from Graham, Knuth, and Patashnik [520, p.441]. For an extensive list of **NP**-complete problems, see Garey and Johnson [441].

§2.4

Two introductory-level books in number theory are Giblin [449] and Rosen [1069]. Good number theory books at a more advanced level include Koblitz [697], Hardy and Wright [540], Ireland and Rosen [572], and Niven and Zuckerman [932]. The most comprehensive works on the design and analysis of algorithms, including number theoretic algorithms, are the first two volumes of Knuth [691, 692]. Two more recent books exclusively devoted to this subject are Bach and Shallit [70] and Cohen [263]. Facts 2.96 and 2.102 are due to Rosser and Schoenfeld [1070]. Shallit [1108] describes and analyzes three algorithms for computing the Jacobi symbol.

§2.5

Among standard references in abstract algebra are the books by Herstein [556] and Hungerford [565].

§2.6

An excellent introduction to finite fields is provided in McEliece [830]. An encyclopedic treatment of the theory and applications of finite fields is given by Lidl and Niederreitter [764]. Two books which discuss various methods of representing the elements of a finite field are those of Jungnickel [646] and Menezes et al. [841].