Arnold L. Rosenberg, Denis Trystram

Understand Mathematics, Understand Computing

Discrete Mathematics that All Computing Students Should Know
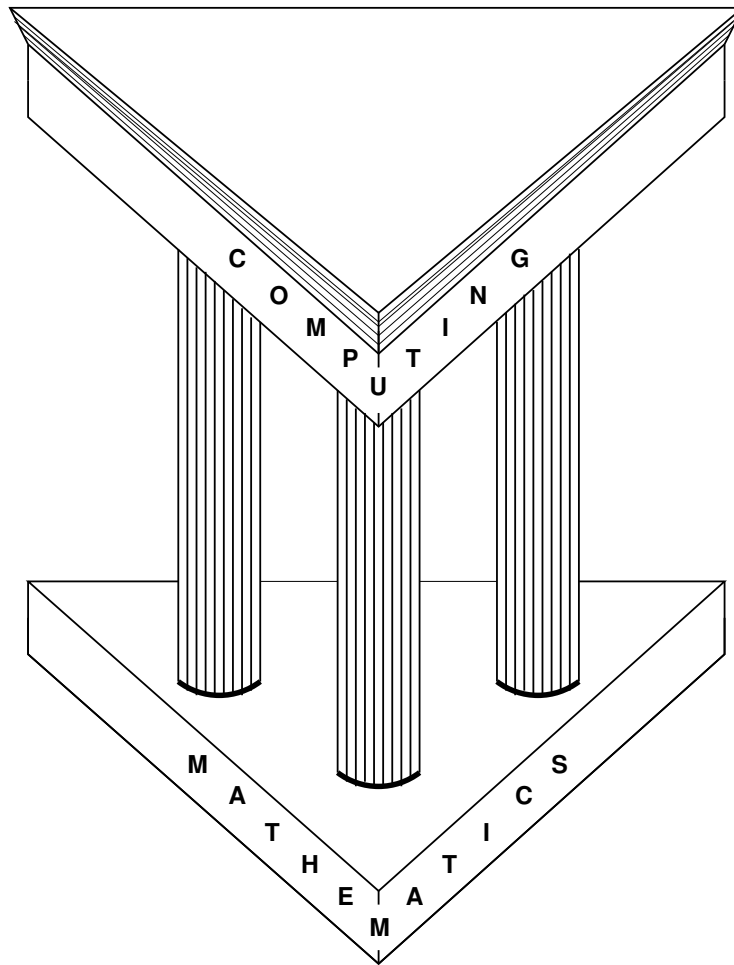
June 5, 2018

Understand Mathematics, Understand Computing
Discrete Mathematics that All Computing Students Should Know

Arnold L. Rosenberg
Distinguished University Professor Emeritus
University of Massachusetts
Amherst, MA 01003, USA
`rsnbrg@cs.umass.edu`

Denis Trystram
Distinguished Professor
University of Grenoble
Grenoble, FRANCE
`denis.trystram@imag.fr`

Mathematics is the foundation on which the edifice of Computing stands

Contents

Chapter 1
INTRODUCTION

Entia non sunt multiplicanda praeter necessitatem.

Occam's Razor

This famous admonition by William of Occam (14th cent.) to strive for simplicity is worth heeding when seeking mathematical models of computational phenomena.

1.1 Overview

In the early days of computing, all aspects of the field were considered the domain of the "techies"—the engineers and scientists and mathematicians who designed the early computers and figured out how to use them to solve a range of problems that is expanding even to this day. Back then, one expected every computer-oriented professional to have a mastery of many mathematical topics. Times—and the field of computing—have changed: classical computing curricula, which have traditional led to a BS degree within a school of science or engineering have been joined by curricula that lead to a BA degree or a degree in IT or in business or .... Within this new world, many aspiring computer professionals arguably need little knowledge of mathematics. However, many argue that the computing field suffers for this lack of mathematics, which has weakened the ability of many practitioners to design and perform experiments, to analyze their results, and to formulate well-reasoned conclusions based on these results. This situation has denied computer science the popular confidence enjoyed by other empirical disciplines; indeed, many would question the math-deprived practitioners' ability to reason rigorously about basic computational phenomena. Yet others, however, argue that such "scientific" acumen is not needed by practitioners in many of the newer segments of the computing field.

The preceding discussion is important to computing educators because we serve a large population of students, with quite diverse needs and aspirations. We strive to keep this diversity in mind within this essay. As we discuss a range of mathematical topics for possible inclusion as prerequisites for the undergraduate study of comput-

ing, we try to indicate why the selected topics are needed. Readers can then evaluate which topics are needed for students enrolled in their computing program.

1.2 The Elements of Rigorous Reasoning

1.2.1 Basic Reasoning

Distinguishing name from object. A fundamental stumbling block in the road to cogent reasoning arises from the inability to distinguish names from the objects they denote. A prime example within the world of computing resides in the inability to distinguish a function (which can be viewed as an infinite set of argument-value pairs) from a program, which can be viewed as a name for the function. **Note** that the often-used view of a function as a *rule* for assigning values to arguments should be avoided, because it suggests – *erroneously* – that an implementable such rule always exists!

Quantitative reasoning. Students should understand the foundational distinction between "growing without bound" and being innite. Within this theme, they should appreciate situations such as the following. Every integer, and every polynomial with integer coefficients, is finite, but there are infinitely many integers and infinitely many polynomials. Students should be able to verify (cogently but not necessarily via any particular formalism) assertions such as the following.

- Let us be given polynomials $p(x)$ of degree a and $q(x)$ of degree $b > a$, where a, b need not be integers. There must exist a constant $X_{p,q}$ (i.e., a constant that depends on the properties of polynomials p and q) such that for all $x > X_{p,q}$, $p(x) < q(x)$.
  Thus, polynomials having bigger degrees eventually *majorize*—i.e., have larger values than—polynomials having smaller degrees.
- Continuing with polynomial q of degree b: For any real number $c > 1$, there exists a constant $Y_{c;q}$ (i.e., a constant that depends on the properties of polynomial q and constant c) such that for all $x > Y_{c;q}$, $c^x > q(x)$.
  Thus, exponential functions eventually *majorize* polynomials.

1.2.1.1 The Elements of Formal Reasoning

induction
    proof by contradiction

1.2.1.2 The Elements of Empirical Reasoning

Empirical reasoning does not convey the certitude that formal reasoning does.

1.3 Our Approach to Mathematical Preliminaries

"If your only tool is a hammer . . . "

We now review a broad range of mathematical concepts that are central to the study and practice of CS/CE. As we develop these concepts, we shall repeatedly observe instances of the following "self-evident truth" (which is what "axiom" means).

> **The conceptual axiom**. *One's ability to think deeply about a complicated concept is always enhanced by having more than one way to think about the concept.*

Chapter 2
TECHNIQUES FOR "DOING" MATHEMATICS

2.1 Reasoning via Rigorous Proof

We survey some important proof techniques

2.1.1 Proof, Implication, Equivalence

2.1.2 Proof by (Finite) Induction

It is crucial that the reader appreciate the fact that proofs by induction, such as we dicuss in this section—cf., Propositions 5.4.3 and 5.4.3—are important tools for *verifying* the correctness of alleged results, but induction by itself is not a tool for *discovering* new results.

2.1.2.1 The proof technique

2.1.2.2 Sample proofs: verifying summation formulas

We illustrate the proof technique of (Finite) Induction by proving the correctness of two familiar summation formulas: (1) the sum of the first n positive integers and (2) the sum of the first n odd positive integers.

Proposition 2.1 *For all $n \in \mathbb{N}$,*

$$S_n \stackrel{def}{=} \sum_{i=1}^{n} i \stackrel{def}{=} 1 + 2 + \cdots + (n-1) + n$$

$$= \frac{1}{2}n(n+1) \tag{2.1}$$

$$= \binom{n+1}{2}.$$

Proof. For every positive integer m, let $\mathbf{P}(m)$ be the proposition

$$1+2+\cdots+m = \binom{m+1}{2}.$$

Let us proceed according to the standard format of an inductive argument.

1. Because $\binom{2}{2} = 1$, proposition $\mathbf{P}(1)$ is true.

2. Let us assume, for the sake of induction, that proposition $\mathbf{P}(m)$ is true for all positive integers strictly smaller than n. In particular, then, $\mathbf{P}(n-1)$ is true.

3. Consider now the summation

$$1+2+\cdots+(n-1)+n.$$

Because $\mathbf{P}(n-1)$ is true, we know that

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

By direct calculation, we see that

$$\begin{aligned}
\binom{n}{2} + n &= \frac{n(n-1)}{2} + n \\
&= \frac{n^2 - n + 2n}{2} \\
&= \frac{n^2 + n}{2} \\
&= \binom{n+1}{2}
\end{aligned}$$

Because n is an arbitrary positive integer, we conclude that $\mathbf{P}(n)$ is true whenever

- $\mathbf{P}(1)$ is true
- *and* $\mathbf{P}(m)$ is true for all $m < n$.

By the Principle of (Finite) Induction, then, we conclude that $\mathbf{P}(n)$ is true for all $n \in \mathbb{N}^+$. $\square$

We turn now to our second summation, which asserts that each perfect square of a positive integer, say, n^2, is the sum of the first n odd integers, $1, 3, 5, \ldots, 2n-1$. This proof complements the constructive proofs of the same result in Proposition 5.3.1.

Proposition 2.2 *For all $n \in \mathbb{N}^+$,*

$$\sum_{k=1}^{n} (2k-1) \;=\; 1+3+5+\cdots+(2n-1) \;=\; n^2.$$

That, is, the nth perfect square is the sum of the first n odd integers.

Verification. For every positive integer m, let $\mathbf{P}(m)$ be the proposition

$$m^2 \;=\; 1+3+5+\cdots+2m-1.$$

Let us proceed according to the standard format of an inductive argument.

1. Because $1 \cdot 1 = 1$, proposition $\mathbf{P}(1)$ is true.

2. Let us assume, for the sake of induction, that proposition $\mathbf{P}(m)$ is true for all positive integers strictly smaller than n. In particular, then, $\mathbf{P}(n-1)$ is true.

3. Consider now the summation

$$1+3+5+\cdots+2n-3+2n-1$$

Because $\mathbf{P}(n-1)$ is true, we know that

$$1+3+5+\cdots+2n-3+2n-1 \;=\; (n-1)^2+2n-1.$$

By direct calculation, we see that

$$(n-1)^2+2n-1 \;=\; (n^2-2n+1)+(2n-1) \;=\; n^2.$$

Because n is an arbitrary positive integer, we conclude that $\mathbf{P}(n)$ is true whenever

- $\mathbf{P}(1)$ is true
- *and* $\mathbf{P}(m)$ is true for all $m < n$.

By the Principle of (Finite) Induction, then, we conclude that $\mathbf{P}(n)$ is true for all $n \in \mathbb{N}^+$. □

2.1.2.3 Making guesses: the method of undetermined coefficients

Proofs by induction, as encountered in Propositions 5.4.3 and 5.4.3, are important tools for *verifying* the correctness of alleged results, but induction by itself is not a tool for *discovering* new results. This section is devoted to the *Method of Undetermined Coefficients*, a tool that sometimes yields the "guesses" that can then be verified via induction. We illustrate the method by deriving a formula for the sum of the first n perfect squares. Our derivation builds on prior knowledge of two facts:

1. A trivial proof by counting verifies that

$$\sum_{k=0}^{n} k^0 \;=\; \sum_{k=1}^{n} 1 \;=\; n.$$

2. We know from Proposition 5.4.3 that

$$\sum_{k=0}^{n} k^1 = \sum_{k=1}^{n} k = \frac{1}{2}(n^2+n)$$

It seems to be silly to include the case $n = 0$ in our summations, since that term does not affect the sum, but that case tells us that the "constant term" in all of these polynomials— i.e., the coefficient of k^0—is always $c_0 = 0$.

Proposition 2.3 *For all $n \in \mathbb{N}$,*

$$S_n^{(2)} \stackrel{\text{def}}{=} \sum_{i=1}^{n} i^2 \stackrel{\text{def}}{=} 1 + 4 + \cdots + (n-1)^2 + n^2$$

$$= \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \tag{2.2}$$

The target quantity $S_n^{(2)}$ in the proposition is often expressed in a more aesthetic form:

$$S_n^{(2)} = \frac{1}{6}n(n+1)(2n+1) = \frac{2n+1}{3} \cdot \binom{n}{2}.$$

Proof. Since summing 0th powers thus gives us a degree-1 polynomial, and summing 1st powers gives us a degree-2 polynomial, it is not unreasonable to guess that summing 2nd powers would give us a degree-3 polynomial. This turns out to be a good guess! To prove this assertion, we must determine values for constants c_1, c_2, c_3 such that

$$\sum_{i=0}^{n} k^2 = c_3 n^3 + c_2 n^2 + c_1 n. \tag{2.3}$$

(Including the case $n = 0$ leaves us with *three* constants to determine rather than four: we know that the coefficient of k^0 is $c_0 = 0$.)

We begin our determination of the constants c_1, c_2, c_3 by instantiating the symbolic polynomial in (5.25) at the smallest three values of n. Any three values will work; using the *smallest* ones simplifies our calculations. These instantiations leaves us with the following system of linear equations. (The summations in (2.4) indicate where each linear equation in the system comes from.)

$$1. \sum_{i=0}^{1} k^2 = c_3 + c_2 + c_1 = 1$$

$$2. \sum_{i=0}^{2} k^2 = 8c_3 + 4c_2 + 2c_1 = 5 \tag{2.4}$$

$$3. \sum_{i=0}^{3} k^2 = 27c_3 + 9c_2 + 3c_1 = 14$$

We use a form of the *Gaussian elimination* algorithm[1] to solve the system by "eliminating variables." First, we rewrite equation 1 as

---

[1] This algorithm is defined and validated in sources such as [20].

$$8c_3 + 8c_2 + 8c_1 = 8$$

and subtract equation 2 from it, thereby obtaining the 2-variable equation

$$4c_2 + 6c_1 = 3. \tag{2.5}$$

We perform a similar calculation based on equations 1 and 3 in system (2.4): We rewrite equation 1 as

$$27c_3 + 27c_2 + 27c_1 = 27$$

and subtract equation 3 from it, thereby obtaining the 2-variable equation

$$18c_2 + 24c_1 = 13. \tag{2.6}$$

We now rewrite equations (2.5) and (2.6) to obtain the simplified system

$$72c_2 + 108c_1 = 54$$
$$72c_2 + 96c_1 = 52$$

We now see, via subtraction, that

$$12c_1 = 2$$

or, equivalently,

$$c_1 = 1/6. \tag{2.7}$$

Now that we know the value of c_1, we can make the indicated substitution and further simplify system (2.4). Since we now have only two variables, we can also eliminate any one of the three equations in (2.4). We eliminate equation 3 in the system in order to simplify our calculations from this point on. We now have the system

$$\begin{array}{ll} 1. \ c_3 + c_2 = 5/6 \\ 2. \ 8c_3 + 4c_2 = 14/3 \end{array} \tag{2.8}$$

We rewrite equation 1 in (2.8) as

$$8c_3 + 8c_2 = 20/3$$

and subtract equation 2 from this version of equation 1. We thereby discover that

$$4c_2 = 2,$$

or, equivalently,

$$c_2 = 1/2. \tag{2.9}$$

Using the values we have discovered for c_1 and c_2, in equations (2.7) and (2.9), respectively, we finally use equation 1 of system (2.4) to determine the value of c_3:

$$c_3 = 1 - 1/2 - 1/6 = 1/3. \tag{2.10}$$

We have, thus, derived equation (5.24).

The careful reader will note that we are not really done yet. We have derived our expression for $S_n^{(2)}$ under the as-yet unjustified assumption that $S_n^{(2)}$ really is a cubic (i.e., degree-3) polynomial. What we need now is an induction that will verify our result. With our previous illustrations as models, we shall leave this final task to the reader.
****
IS THIS OK?
****************   □

With more (calculational) work, but no new (mathematical) ideas, one can derive explicit expressions for the sum of the first n kth powers, i.e., the quantity $S_n^{(k)}$, for any positive integer k.

2.1.3 Proof by Contradiction

2.1.3.1 The Proof Technique

The basic principle that underlies proof by contradiction is that the following *meta-mathematical* assertions are logically equivalent.

- Proposition P IMPLIES Proposition Q
- Proposition $\sim Q$ IMPLIES Proposition $\sim P$

As with many of these *metamathematical* principles, there is a corresponding *mathematical* equivalence, in this case, Proposition 3.4.

$$[P \Rightarrow Q] \equiv [(\sim Q) \Rightarrow (\sim P)]$$

2.1.3.2 Sample Proofs

A. There are infinitely many primes. The following result is traditionally attributed to the Greek mathematician Euclid, one of the patriarchs of mathematics.

Proposition 2.4 *There are infinitely many prime numbers.*

Proof. Let us assume, contrarily, that there are only finitely many primes. Say, in particular, that the following r-element sequence enumerates all (and only) primes, in order of magnitude:
   Prime-Numbers $= \langle P_1, P_2, \ldots, P_r \rangle$
where

- $P_1 = 2$
- $P_2 = 3$

- $P_i < P_{i+1}$ for all $i \in \{1, 2, \ldots, r-1\}$.

We verify the *falseness* of the alleged completeness of the sequence **Prime-Numbers** by analyzing the positive integer

$$n \;=\; 1 + \prod_{i=1}^{r} P_i \;=\; 1 \,+\, (P_1 \cdot P_2 \cdot \cdots \cdot P_r).$$

We make three crucial observations.

1. We note first that *the number n is not divisible by any prime number in the sequence* **Prime-Numbers**.
   To see this, note that for each P_k in the sequence,

   $$n/P_k \;=\; \frac{1}{P_k} \,+\, \prod_{i \neq k} P_i.$$

   Because $P_k \geq 2$, we see that n/P_k obeys the inequalities

   $$\prod_{i \neq k} P_i \;<\; n/P_k \;<\; 1 + \prod_{i \neq k} P_i.$$

   The discreteness of the set $\mathbb{Z}$—see Section 4.1.1.1.A—implies that n/P_k is not an integer, because it lies strictly between two adjacent integers.
2. We note next that, because of assertion 1, if the sequence **Prime-Numbers** actually contained *all* of the prime numbers, then we would have to conclude that *the number n is not divisible by any prime number.*
3. Finally, we remark that the Fundamental Theorem of Arithmetic (Theorem 4.2) implies that *every integer is divisible by (at least one) prime number*.

We have a chain of assertions that lead to a mutual inconsistency: on the one hand, the integer *n* has no prime-integer divisor; on the other hand, no integer can fail to have a prime-integer divisor! Let us analyze how we arrived at this uncomfortable place.

- At the front end of this uncomfortable string of assertions we have the assumption that there are only finitely many prime numbers. We have (as yet) no substantiation for this assertion.
- At the back end of this uncomfortable string of assertions we have the (*rock solid*) Fundamental Theorem of Arithmetic (Theorem 4.2).
- In between these two assertions we have a sequence of assertions, each of which follows from its predecessors via irrefutable rules of inference.

It follows that the *only* brick in this edifice that could be faulty—i.e., the only assertion that could be false—is the assumption that there are only finitely many prime numbers. Since this assumption leads to an inconsistent set of assertions, we must conclude that the assumption is false! We conclude from this classical proof by contradiction that there are infinitely many prime numbers. □

2.1.4 Proofs via the Pigeonhole Principle

The proof technique we discuss now builds on an observation that is almost embarrassingly obvious—yet its simplicity is exceeded by its importance as a source of strikingly surprising results.

2.1.4.1 The Proof Technique

The technique, known variously as *the pigeonhole principle* or *Dirichlet's Box Principle* (after the French mathematician Peter Gustav Lejeune Dirichlet), exploits the fact that if one has n objects (say, pigeons) and $m < n$ boxes (they're the pigeonholes), then any way of putting pigeons into boxes must place at least two pigeons into the same box.

2.1.4.2 Sample (Fun) Applications/Proofs

A. Choosing a pair of matching socks. You have n pairs of socks, the socks in each pair having a distinct color (one pair of red socks, one pair of blue socks, …). Since you wake up "very slowly", you want to grab some number of unpaired socks that is cerain to yield at least one pair of smame-color socks. Clearly, if you grab any $n + 1$ socks (the pigeons), the pigeonhole principle guarantees that you have at least one monochromatic pair, because there are only n distinct sock-colors (the boxes).

B. Finding birthday-mates. You are attending a conference and wander into a lecture that has 367 attendees (including you). It is certain that at least two attendees share the same birthday: there are 366 possible birthdays (the boxes for a leap year) and 367 birthday-possessors (the pigeons).

C. On meeting new people. You are attending a cocktail party that is populated by n couples. In order to create a warm atmosphere, the host requests that each attendee shake the hand of every attendee that he or she does not know. The pigeonhole principle guarantees that some two attendees shake the same number of hands. To wit, the number of people that each attendee *does not know* belongs to the set $\{0, 1, \ldots, 2n - 2\}$, because each person knows him/herself and his/her partner. Because there are $2n$ handshakers (the pigeons) and $2n - 1$ numbers of hands to shake (the boxes), some two shakers must shake the same numbers of hands.

D. Friends and strangers at a party. We turn now to a somewhat more surprising result that can be proved using the pigeonhole principle. While we phrase the result in anthropomorphic, "homely", terms, its formal statement identifies it as a genre of "unavoidable subgraph" phenomenon within the theory of *graphs*.

> Graphs are an immensely important mathematical construct that models myriad situations that involve objects (possibly people) and interrelationships between pairs of objects. Chapter 8 is devoted to studying graphs and the situations they can be used to model—including the problem discussed here.

Here is the "homely" version of the *Friends and Strangers* problem.

Proposition 2.5 *In any gathering of six people, at least one of the following assertions is true.*
A. *There is a group of three people who know each other.*
B. *There is a group of three people none of whom knows either of the others.*

Proof. Let the gathering consist of six indistinguishable people, named P_1, P_2, P_3, P_4, P_5, P_6. Focus on an arbitrary person, say P_5. (This choice "sounds" more arbitrary than P_1—but, of course, is not.) Now, there are 5 people, namely, P_1, P_2, P_3, P_4, P_6, each of whom P_5 either *knows* or *doesn't know*.

Clearly, some 3 of these 5 people "lie on the same side of the *know/don't-know* fence." This follows from the pigeonhole principle: we have *two* boxes (*know* and *doesn't know*) and *five* pigeons (the people P_1, P_2, P_3, P_4, P_6). Any way of putting the pigeons into the boxes will place three people into one of the boxes.

Say, with no loss of generality, that P_5 *knows* P_1, P_2, P_3.

> Why can we claim that the selected situation— "P_5 *knows* P_1, P_2, P_3"—can be assumed "with no loss of generality"? One should *always* ask this question about such a claim! In the current case, the claim follows from the following facts.
>
> (a) The names that we use to refer to the six assembled people are just for our expository benefit. The names carry no inherent meaning related to the *Friends and Strangers* problem. You can repeat our argument while choosing arbitrary replacements for P_1, P_2, P_3, P_5, with no change to the logical outcome.
>
> You can also interchange the *know* and *don't-know* labels. The underlying logic will not change, although the conclusions regarding options A and B in the statement of the proposition will clearly "flip".

Having decided that P_5 *knows* P_1, P_2, and P_3, we now consider the implications of the possible relations between each of the three pairs of people chosen from $\{P_1, P_2, P_3\}$. There are two logical possibilities.

- Some two of P_1, P_2, P_3 know each other—say, with no loss of generality, P_1 and P_2. In this case, P_1, P_2, and P_5 form a trio of people who know one another (option A in the statement of the proposition).
- No two of P_1, P_2, P_3 know each other. In this case, P_1, P_2, and P_3 form a trio of people none of whom knows either of the others (option B in the statement of the proposition).

This disjunction completes the proof.

We close the proof by noting that nothing we have stated precludes the possibility that *both* option A *and* option B are true! □

2.2 Reasoning via Mathematical Analysis

2.2.1 Analyses via Linear Recurrences

Theorem 2.1 (The Master Theorem for Linear Recurrences). *Let the function F be specified by the following linear recurrence.*

$$F(n) \; = \; \begin{cases} aF(n/b) + c \text{ for } n \geq b \\ c \qquad \text{ for } n < b \end{cases} \tag{2.11}$$

Then the value of F on any argument n is given by

$$F(n) = (1 + \log_b n)c \qquad \text{if } a = 1$$

$$= \frac{1 - a^{\log_b n}}{1 - a} \; \approx \; \frac{1}{1 - a} \quad \text{if } a < 1 \tag{2.12}$$

$$= \frac{a^{\log_b n} - 1}{a - 1} \qquad \text{if } a > 1$$

2.2.2 Asymptotics

Asymptotics can be viewed as a language and a system of reasoning that allow one to talk in a *qualitative* voice about *quantitative* topics. We thereby generalize to arbitrary growth functions terms such as "linear", "quadratic", "exponential", and "logarithmic".

Such a language and system are indispensable if one needs to reason about computational topics over a range of situations, such as a range ("all existing"?) computer architectures and software systems. As two simple examples: (1) Carry-ripple adders perform additions in time linear in the lengths n of the summands (measured in number of bits) no matter what these lengths are. (2) Comparison-based sorting algorithms can sort lists of n keys in time proportional to $n \log n$, but no faster— where the base of the logarithm depends on the characteristics of the computing platform. More precise versions of the preceding statements require specication of the number n and other details, possibly down to the clock speeds of the host computer's circuitry.

2.2.2.1 The language of asymptotics

The language of asymptotics, which has its origins in the field of Number Theory in the late 19th century, builds on the following terminology, which is likely what one would cover in an early undergraduate course. More advanced aspects of the

language would likely by beyond the needs of most students of computing, aside from specialists in advanced courses. The basics of the language build on three primitive notations and notions. Standard sources, such as any text on algorithm design and analysis, flesh out the following ideas.

- *The big-O notation.* The assertion $f(x) = O(g(x))$ says, intuitively, that the function f grows no faster than function g. It is, thus, the asymptotic analogue of "less than".
  Formally: $f(x) = O(g(x))$
  means
  $(\exists c > 0)(\exists x^{\#})(\forall x > x^{\#})[f(x) \leq c \cdot g(x)]$
- *The big-Ω notation.* The assertion $f(x) = \Omega(g(x))$ says, intuitively, that the function f grows at least as fast as function g. It is, thus, the asymptotic analogue of "greater than".
  Formally: $f(x) = \Omega(g(x))$
  means
  $(\exists c > 0)(\exists x^{\#})(\forall x > x^{\#})[f(x) \geq c \cdot g(x)]$

- *The big-Θ notation.* The assertion $f(n) = \Theta(g(n))$ says, intuitively, that the function f grows at the same rate as does function g. It is, thus, the asymptotic analogue of "equal to".
  Formally: $f(x) = \Theta(g(x))$
  means
  $(\exists c_1 > 0)(\exists c_2 > 0)(\exists x^{\#})(\forall x > x^{\#})[c_1 \cdot g(x) \leq f(x) \leq c_2 \cdot g(x)]$

One renders the preceding intuitive explanations precise by pointing out that the three specifies relations (*a*) take hold *eventually*, i.e., only for large arguments to the functions f and g, and (*b*) hold up to an unspecified constant of proportionality.

2.2.2.2 The "uncertainties" in asymptotic relationships

The formal definitions of all three of our asymptotic relationships are bracketed by two important quantifiers:

$$\text{``}(\exists c > 0)\text{''} \quad \text{and} \quad \text{``}(\forall x > x^{\#})\text{''}.$$

The former, *uncertain-size* quantifier, asserts that asymptotic notions describe functional behavior "in the large". Thus, in common with more common qualitative descriptors of quantitative growth such as linear, quadratic, cubic, quartic, exponential, logarithmic, etc., asymptotic relationships give no infomation about constants of proportionality. *We are not saying that constant factors do not matter! We are, rather, saying that we want to discuss growth patterns in the large.*

The latter, *uncertain-time* quantifier asserts that asymptotic relationships between functions are promised to hold only "eventually", i.e., "for sufficiently large values of the argument x". Therefore, in particular, asymptotic notions cannot be employed

to discuss or analyze quantities that can never grow beyond a fixed finite value. The fact that all instances of a quantity throughout history have been below N is immaterial, as long as it is conceivable that an instance larger than N could appear at some time in the future.

These quantifiers in particular distinguishes claims of asymptotic relationship from the more familiar definite inequalities such as "$f(x) \leq g(x)$" or $f(x) \geq 7 \cdot g(x)$. In fact, it is often easier to think about our three asymptotic bounding assertions as establishing *envelopes* for $f(x)$:

- Say that $f(x) = O(g(x))$. If one draws the graphs of the functions $f(x)$ and $c \cdot g(x)$, then as one traces the graphs with increasing values of x, one eventually reaches a point $x^{\#}$ beyond which the graph of $f(x)$ never enters the territory *above* the graph of $c \cdot g(x)$.
- Say that $f(x) = \Omega(g(x))$. This situation is the up-down mirror image of the preceding one: just replace the highlighted "*above*" with "*below*."
- Say that $f(x) = \Theta(g(x))$. We now have a two-sided envelope: beyond $x^{\#}$, the graph of $f(x)$ never enters the territory *above* the graph of $c_1 \cdot g(x)$ and never enters the territory *below* the graph of $c_2 \cdot g(x)$.

In addition to allowing one to make familiar growth-rate comparisons such as "$n^{14} = O(n^{15})$" and "$1.001^n = \Omega(n^{1000})$," we can now also make assertions such as "$\sin x = \Theta(1)$," which are much clumsier to explain in words.

Beyond the big letters. There are "small"-letter analogues of the preceding "big"-letter asymptotic notations, but they are only rarely encountered in discourse about real computations (although they do arise in the analysis of algorithms).

2.2.2.3 Inescapable complications

The story we have told thus far is covered in many sources and courses. Two complications to the story are covered less faithfully, although lacking them, one cannot perform cogent asymptotic reasoning. Both complications involve the notion of *uniformity*.

1. *Multiple functions*. Say that we have four functions, f, g, h, k, and we know that both

$$f(n) = O(g(n)) \quad \text{and} \quad h(n) = O(k(n))$$

It is intuitive that

$$f(n) + h(n) = O(g(n) + k(n))$$

— but is it true?

In short, the answer is YES, but verifying that requires a bit of subtlety, because, absent hitherto undisclosed information, the proportionality constants $c_{f,g}$ and $N_{f,g}$ that witness the big-O relationship between functions f and g have no connection with the constants, $c_{h,k}, N_{h,k}$ that witness the analogous relationship between functions h and k. Therefore, in order to verify the posited relationship between functions $f + h$ and $g + k$, one much find witnessing constants $c_{f+h,g+k}$ and $N_{f+h,g+k}$.

Of course, this task requires only elementary reasoning and manipulation — but it must be done!

2. *Multivariate functions*. Finally, we discuss the scenario that almost automatically accompanies the transition from a focus on sequential, single-agent computing to a focus on PDC. Within this broadened context, most functions that describe a system have one or more variables that describe the computing system — its number of processors or of agents or the sizes of its memory modules or the communication-radii of its transponders or ..., in addition to the one or more variables that describe the data that the system is processing. Within such scenarios, every assertion of an asymptotic relationship, of the form

$$f(\mathbf{m};\mathbf{n}) = O(g(\mathbf{m};\mathbf{n}))$$

must explicitly specify the following information:

- which variables can grow without bound;
- among such unbounded variables, which participate in the posited asymptotic relation;
- for each participating unbounded variable x, what are the constants c_x and N_x that witness the posited asymptotic relationship(s).

Clearly the complexity of cogent asymptotic reasoning — hence also the complexity of teaching about such reasoning — gets much more complicated in the multivariate settings engendered by PDC. But, the benefits of being able to reason qualitatively about the quantitative aspects of computing increase at least commensurately!

Chapter 3
SETS, BOOLEAN ALGEBRA, AND LOGIC

3.1 Sets

3.1.1 Fundamental Set-Related Concepts

Sets are probably the most basic object of mathematical discourse. Sets exist to have *elements*, or *members*, the entities that *belong to* the set. The notion of set is surprisingly difficult to specify formally, so we just assume that *the reader knows what a set is and recognizes that some sets are finite, while others are infinite*. Speaking informally—a formal treatment will follow in later chapters—here are a few illustrative finite sets:

- the set of words in this book
  I do not know how big this set is, but I imagine that you as a reader have a better idea than I as an author.
- the set of characters in any JAVA program
  Note that while we are sure that this set is finite, we are not so confident about the number of seconds the program will run!
- the set consisting of *you*
  Paraphrasing the iconic television figure Mister Rogers, "You are unique." This set has just one element.
- the set of unicorns in New York City
  I will not argue with you about this, but I believe that this is the *empty set ∅*, which has zero members.

Some familiar infinite sets are:

- the set of *nonnegative integers*
- the set of *positive integers*
- the set of *all integers*
- the set of nonnegative *rational numbers*—which are quotients of integers
- the set of nonnegative *real numbers*—which can be viewed computationally as the set of numbers that admit infinite decimal expansions,

19

- the set of nonnegative *complex numbers*—which can be viewed as ordered pairs of real numbers,
- the set of *all* finite-length binary strings.

  A *binary string* is a sequence of 0s and 1s. When discussing computer-related matters, one often calls each 0 and 1 that occurs in a binary string a *bit* (for *binary digit*). The term "bit" leads to the term *bit string* as a synonym of *binary string*.

Despite this assumption, we begin the chapter by reviewing some basic concepts concerning sets and operations thereon.

As noted early, sets were created to contain members/elements. We denote the fact that element *t belongs to*, or, *is an element of* set T by the notation $t \in T$. A *subset* of a set T is a set S each of whose members belongs to T. The subset relation occurs in two forms, The *strong* form of the relation, denoted $S \subset T$, says that every element of S is an element of T, but *not* conversely; i.e., T contains (one or more) elements that S does not. The *weak* form of the relation, denoted $S \subseteq T$, is defined as follows:

$$[S \subseteq T] \quad \text{means:} \quad \left[either \quad [S = T] \quad or \quad [S \subset T] \right].$$

For any finite set S, we denote by $|S|$ the *cardinality* of S, which is the number of elements in S. Finite sets having three special cardinalities are singled out with special names. The limiting case of finite sets is the unique *empty set*, which we denote by $\emptyset$; thus, $\emptyset$ is characterized by the equation $|\emptyset| = 0$. (The empty set is often a limiting case of set-defined entities.) If $|S| = 1$, then we call S a *singleton*; and if $|S| = 2$, then we call S a *doubleton*.

It is often useful to have a convenient term and notation for *the set of all subsets of a set S*. This bigger set—it contains $2^{|S|}$ elements when S is finite—is denoted by $\mathscr{P}(S)$ and is called the *power set* of S.[1] Note carefully the two set-relations that we are talking about here:

A set T that is a subset *of set S is an* element *of the set* $\mathscr{P}(S)$.

You should satisfy yourself that the biggest and smallest elements of $\mathscr{P}(S)$ are, respectively, the set S itself and the empty set $\emptyset$.

3.1.2 Operations on Sets

Given two sets S and T, we denote by:

- $S \cap T$ the *intersection* of S and T: the set of elements that belong to *both S and T*.

$$[s \in S \cap T] \quad \text{means} \quad \left[[s \in S] \text{ and } [s \in T] \right]$$

- $S \cup T$ the *union* of S and T: the set of elements that belong to S, or to T, *or to both*. (Because of the "or both" qualifier, this operation is sometimes called *inclusive*

---

[1] The name "power set" arises from the relative cardinalities of S and $\mathscr{P}(S)$ for finite S.

union.)
$$[s \in S \cup T] \quad \text{means} \quad \Big[[s \in S] \textbf{ or } [s \in T] \textbf{ or } [s \in S \cap T]\Big]$$

- $S \setminus T$ is the *(set) difference* of S and T: the set of elements that belong to S but not to T.
$$[s \in S \setminus T] \quad \text{means} \quad \Big[[s \in S] \textbf{ and } [s \notin T]\Big]$$

(Particularly in the United States, one often encounters the notation "$S - T$" instead of "$S \setminus T$.")

We illustrate the preceding operations with the sets $S = \{a, b, c\}$ and $T = \{c, d\}$. For these sets:

$$S \cap T = \{c\},$$
$$S \cup T = \{a, b, c, d\},$$
$$S \setminus T = \{a, b\}.$$

In many set-related situations, the sets of interest will be subsets of some fixed "universal" set U.

We use the term "universal" as in "universe of discourse," not in the self-referencing sense of a set that contains all other sets as members, a construct (discussed by philosopher-logician Bertrand Russell) which leads to mind-bending paradoxes.

Given a universal set U and a *subset $S \subseteq U$*, we observe the set-inequalities

$$\emptyset \subseteq S \subseteq U.$$

When studying a context within which there exists a universal set U that contains all other sets of interest, we include within our repertoire of set-related operations also the operation of *complementation*

- $\overline{S} \stackrel{\text{def}}{=} U \setminus S$, the *complement* of S (relative to the universal set U).
  For instance, the set of odd positive integers is the complement of the set of even positive integers, relative to the set of all positive integers.

We note a number of basic identities involving sets and operations on them.

- $S \setminus T = S \cap \overline{T}$,
- If $S \subseteq T$, then

  1. $S \setminus T = \emptyset$,
  2. $S \cap T = S$,
  3. $S \cup T = T$.

Note, in particular, that[2]

$$[S = T] \text{ iff } \Big[[S \subseteq T] \text{ and } [T \subseteq S]\Big] \text{ iff } \Big[(S \setminus T) \cup (T \setminus S) = \emptyset\Big].$$

---

[2] "iff" abbreviates the common mathematical phrase, "if and only if."

The operations union, intersection, and complementation—and operations formed from them, such as set difference—are usually called the *Boolean (set) operations*, (named for the 19th-century English mathematician George Boole). There are several important identities involving the Boolean set operations. Among the most frequently invoked are the two "laws" attributed to the 19th-century French mathematician Auguste De Morgan:

$$\text{For all sets } S \text{ and } T\text{:} \quad \begin{cases} \overline{S \cup T} = \overline{S} \cap \overline{T}, \\[2mm] \overline{S \cap T} = \overline{S} \cup \overline{T}. \end{cases} \tag{3.1}$$

(Algebraic) Closure. We end this section with a set-theoretic definition that occurs in many contexts. Let $\mathscr{C}$ be any (finite or infinite) collection of sets, and let S and T be two elements of $\mathscr{C}$. (Note that $\mathscr{C}$ is a set whose elements are sets.) Think, e.g., of the concrete example of set intersection.

We say that $\mathscr{C}$ is *closed* under intersection if whenever sets S and T (which could be the same set) both belong to $\mathscr{C}$, the set $S \cap T$ also belongs to $\mathscr{C}$. By De Morgan's laws, $\mathscr{C}$'s closure under union implies also its closure under intersection.

3.2 Binary Relations

3.2.1 The Formal Notion of Binary Relation

We begin our discussion of relations by adding a new (binary) set operation to our earlier repertoire. Given (finite or infinite) sets S and T we denote by $S \times T$ the *direct product* of S and T, which is the set of all *ordered pairs* whose first coordinate contains an element of S and whose second coordinate contains an element of T. For example, if $S = \{a, b, c\}$ and $T = \{c, d\}$, then

$$S \times T = \{\langle a, c \rangle, \langle b, c \rangle, \langle c, c \rangle, \langle a, d \rangle, \langle b, d \rangle, \langle c, d \rangle\}$$

The direct-product operation on sets affords us a simple, yet powerful, formal notion of binary relation.

Given (finite or infinite) sets S and T, a *relation ρ on S and T* (in that order) is any subset

$$\rho \subseteq S \times T.$$

When $S = T$, we often call ρ a *binary relation on (the set) S* ("*binary*" because there are *two* copies of set S being related by ρ).

Relations are so common that we use them in every aspect of our lives without even noticing them. The relations "equal to," "less than," and "greater than or equal to" are simple examples of binary relations on the integers. These same three relations apply also to other familiar number systems such as the rational and real numbers; only "equal," though, holds (in the natural way) for the complex numbers.

Some subset of the three relations "is a parent of," "is a child of," and "is a sibling of" probably are binary relations on (the set of people constituting) your family. To mention just one relation with distinct sets S and T, the relation "A is taking course X" is a relation on

$$(\text{the set of all students}) \times (\text{the set of all courses}).$$

By convention, when dealing with a binary relation $\rho \subseteq S \times T$, we often write "$s\rho t$" in place of the more stilted notation "$\langle s,t \rangle \in \rho$." For instance we (almost always) write "$5 < 7$" in place of the strange-looking (but formally correct) "$\langle 5,7 \rangle \in <$."

The following operation on relations occurs in many guises, in almost all mathematical theories. Let ρ and ρ' be binary relations on a set S. The *composition* of ρ and ρ' (in that order) is the relation

$$\rho'' \stackrel{\text{def}}{=} \left\{ \langle s,t \rangle \in S \times S \mid (\exists t \in S) \Big[[s\rho t] \text{ and } [t\rho' u] \Big] \right\}.$$

Note that we have used both of our notational conventions for relations here. We also encounter here, for the first time in the text, but certainly not the last, a new notational convention: the common "shorthand" compound symbol "$\stackrel{\text{def}}{=}$": The sentence "$X \stackrel{\text{def}}{=} Y$" should be read "$X$ *is (or, equals), by definition*, Y."

> The operation of composition of relations is quite important in the study of "relational" databases.

It is important to be able to assert that elements $s,t \in S$ are *not* ρ-related, i.e., $\langle s,t \rangle \notin S \times S$. Several notations have been developed for this purpose.

Relation	Notation	Negation	Standard?
set membership	$\in$	$\notin$	yes
equality	$=$	$\neq$	yes
less than (strong)	$<$	$\not< $ or $\geq$	yes
less than (weak)	$\leq$	$\not\leq$ or $>$	yes
greater than (strong)	$>$	$\not>$ or $\leq$	yes
greater than (weak)	$\geq$	$\not\geq$ or $<$	yes
generic	ρ	$\sim\rho$ or $\widetilde{\rho}$	no

(3.2)

There are several special classes of binary relations that are so important that we must single them out immediately, in the following subsections.

3.2.2 Order Relations

A binary relation ρ on a set S is a *partial order relation*, or, more briefly, is a *partial order* if ρ is transitive. This means that, for all elements $s,t,u \in S$,

$$\text{if} \quad sRt \quad \text{and} \quad tRu \quad \text{then} \quad sRu. \tag{3.3}$$

The qualifier "partial" warns us that some pairs of elements of S do not occur in relation ρ. Number-related orders supply an easy illustrative example. Given any two distinct integers, m and n, one of them must be less than the other: either $m < n$, or $n < m$. In contrast, if we consider *ordered pairs* of integers, then there are pairs of pairs that are not related by the "less than" relation in any natural way. For instance, even though we may agree that, by a natural extension of the number-ordering relation "less than", $\langle 4, 17 \rangle$ is "less than" $\langle 22, 19 \rangle$, we might well not agree on which of $\langle 4, 22 \rangle$ and $\langle 19, 17 \rangle$ is less than the other—or, indeed, whether either is "less than" the other.

In many domains, order relations occur in two "flavors", *strong* and *weak*. For many such relations ρ—consider, e.g., "less than" on the integers—the weak version is denoted by underscoring the strong one's symbol. This will be our convention. Just as $\leq$ denotes the weak version of $<$, and $\geq$ denotes the weak version of $>$, we shall denote the weak version of a generic order ρ by $\underline{\rho}$.. Strong and weak versions of an order relation ρ (denoted, respectively, ρ and $\overline{\rho}$) are distinguished by their behavior under simultaneous membership. For illustration, instantiate the following template with ρ being "$<$" and with ρ being "$>$":

For a strong order ρ: **if** $[s \, \rho \, t]$, **then** $[t \, \widetilde{\rho} \, s]$
For the weak version $\underline{\rho}$ of ρ: **if** $[s \, \underline{\rho} \, t]$ **and** $[t \, \underline{\rho} \, s]$, **then** $[s = t]$.

3.2.3 Equivalence Relations

A binary relation R on a set S is an *equivalence relation* if it enjoys the following three properties:

1. R is *reflexive:* for all $s \in S$, we have sRs.
2. R is *symmetric:* for all $s, s' \in S$, we have sRs' whenever $s'Rs$.
3. R is *transitive:* for all $s, s', s'' \in S$, whenever we have sRs' and $s'Rs''$, we also have sRs''.

Sample familiar equivalence relations are:

- The equality relation, $=$, on a set S which relates each $s \in S$ with itself but with no other element of S.
- The relations $\equiv_{12}$ and $\equiv_{24}$ on integers, where[3]

  1. $n_1 \equiv_{12} n_2$ if and only if $|n_1 - n_2|$ is divisible by 12.
  2. $n_1 \equiv_{24} n_2$ if and only if $|n_1 - n_2|$ is divisible by 24.

  We use relation $\equiv_{12}$ (without formally knowing it) whenever we tell time using a 12-hour clock and relation $\equiv_{24}$ whenever we tell time using a 24-hour clock.

---

[3] As usual, $|x|$ is the *absolute value*, or, *magnitude* of the number x. That is, if $x \geq 0$, then $|x| = x$; if $x < 0$, then $|x| = -x$.

Closely related to the notion of an equivalence relation on a set S is the notion of a *partition* of S. A partition of S is a nonempty collection of subsets $S_1, S_2, \ldots$ of S that are

1. *mutually exclusive:* for distinct indices i and j, $S_i \cap S_j = \emptyset$;
2. *collectively exhaustive:* $S_1 \cup S_2 \cup \cdots = S$.

We call each set S_i a *block* of the partition.

One verifies the following Proposition easily.

Proposition 3.1 *A partition of a set S and an equivalence relation on S are just two ways of looking at the same concept.*

To verify this, we note the following.

Getting an equivalence relation from a partition. Given any partition $S_1, S_2, \ldots$ of a set S, define the following relation R on S:

sRs' if and only if s and s' belong to the same block of the partition.

Relation R is an equivalence relation on S. To wit, R is reflexive, symmetric, and transitive because collective exhaustiveness ensures that each $s \in S$ belongs to some block of the partition, while mutual exclusivity ensures that it belongs to only one block.

Getting a partition from an equivalence relation. To obtain the converse, focus on any equivalence relation R on a set S. For each $s \in S$, denote by $[s]_R$ the set

$$[s]_R \overset{\text{def}}{=} \{s' \in S \mid sRs'\};$$

we call $[s]_R$ *the equivalence class of s under relation R.*

The equivalence classes under R form a partition of S. To wit: R's reflexivity ensures that the equivalence classes collectively exhaust S; R's symmetry and transitivity ensure that equivalence classes are mutually disjoint.

The *index* of the equivalence relation R is its number of classes—which can be finite or infinite.

Let[4] $\equiv_1$ and $\equiv_2$ be two equivalence relations on a set S. We say that the relation $\equiv_1$ *is a refinement of* (or, *refines*) the relation $\equiv_2$ just when each block of $\equiv_1$ is a subset of some block of $\equiv_2$. We leave to the reader the simple verification of the following basic result.

Theorem 3.1. *The equality relation, $=$, on a set S refines every equivalence relation on S. In this sense, it is the finest equivalence relation on S.*

3.2.4 Functions

One learns early in school that a function from a set A to a set B is a rule that assigns a unique value from B to every value from A. Simple examples illustrate that this

---

[4] Conforming to common usage, we typically use the symbol $\equiv$, possibly embellished by a subscript or superscript, to denote an equivalence relation.

notion of function is more restrictive than necessary. Think, e.g., of the operation *division* on integers. We learn that division, like multiplication, is a function that assigns a number to a given pair of numbers. Yet we are warned almost immediately not to "divide by 0": The quotient upon division by 0 is "undefined." So, division is not quite a function as envisioned our initial definition of the notion. Indeed, in contrast to an expression such as "$4 \div 2$," which should lead to the result 2 in any programming environment,[5] expressions such as "$4 \div 0$" will lead to wildly different results in different programming environments. Since "wildly different" is anathema in any mathematical setting, we deal with situations such as just described by broadening the definition of "function" in a way that behaves like our initial simple definition under "well-behaved" circumstances and that extends the notion in an intellectually consistent way under "ill-behaved" circumstances. Let us begin to get formal.

A *(partial) function from set S to set T* is a relation $F \subseteq S \times T$ that is *single-valued;* i.e., for each $s \in S$, there is *at most* one $t \in T$ such that sFt. We traditionally write "$F : S \to T$" as shorthand for the assertion, "F is a function from the set S to the set T"; we also traditionally write "$F(s) = t$" for the more conservative "sFt." (The single-valuedness of F makes the nonconservative notation safe.) We often call the set S the *source (set)*, or, the *domain* of function F, and we call set T the *target (set)* or, the *range* of function F. When there is always a (perforce, unique) $t \in T$ for each $s \in S$, then we call F a *total* function.

You may be surprised to encounter functions that are not total, because most of the functions you deal with daily are *total*. Our mathematical ancestors had to do some fancy footwork in order to make your world so neat. Their choreography took two complementary forms.

1. They expanded the target set T on numerous occasions. As just two instances:

   - They appended both 0 and the negative integers to the preexisting positive integers[6] in order to make subtraction a total function.
   - They appended the rationals to the preexisting integers in order to make division (by nonzero numbers!) a total function.

   The irrational algebraic numbers, the nonalgebraic real numbers, and the nonreal complex numbers were similarly appended, in turn, to our number system in order to make certain (more complicated) functions total.

2. They adapted the function. In programming languages, in particular, true undefinedness is anathema, so such languages typically have ways of making functions total, via devices such as "integer division" (so that odd integers can be "divided by 2") as well as various ploys for accommodating "division by 0."

The (20th-century) inventors of *Computation Theory* insisted on a theory of functions on nonnegative integers (or some transparent encoding thereof). The price for such "pureness" is that we must allow functions to be undefined on some arguments.

---

[5] We are, of course, ignoring demons such as round-off error.

[6] The great mathematician Leopold Kronecker said, "God made the integers, all else is the work of man"; Kronecker was referring, of course, to the *positive* integers.

Thus the theory renders such functions as "division by 2" and "taking square roots" as being *nontotal*: both are defined only on subsets of the positive integers (the even integers and the perfect squares, respectively).

Three special classes of functions merit explicit mention. For each, we give both a down-to-earth name and a more scholarly Latinate one.

A function $F : S \to T$ is:

1. *one-to-one* (or *injective*) if for each $t \in T$, there is at most one $s \in S$ such that $F(s) = t$;

   Example:

   - "multiplication by 2" is injective: If I give you an even integer $2n$, you can always respond by giving me n.
   - "integer division by 2" is not injective—because performing the operation on arguments $2n$ and $2n + 1$ yields the same answer (namely, n).

   An injective function F is called an *injection*.

   Importantly, each injection F has a *functional inverse*, which is commonly denoted F^{-1}. and which is defined as follows. For each $t \in T$:

   - If there is an $s \in S$ such that $F(s) = t$, then

$$F^{-1}(t) \; = \; s$$

   - If there is no $s \in S$ such that $F(s) = t$, then $F^{-1}(t)$ is not defined.

   Because F is an *injection*, there is at most $s \in S$ such that $F(s) = t$. In other words, an element $t \in T$ can occur in the range of F only because of a single element $s \in S$ in the domain of F. This means that the preceding definition of F^{-1} is a valid definition (it is "well-defined") and that F^{-1} is a (partial) function $F^{-1} : T \to S$ whose domain is the range of F.

2. *onto* (or *surjective*) if for each $t \in T$, there is at least one $s \in S$ such that $F(s) = t$;

   Example:

   - Two surjective functions on the nonnegative integers:
     – "subtraction of 1" is surjective, because "addition of 1" is a total function.
     – "taking the square root" is surjective because the operation of squaring is a total function.
   - Two functions on the nonnegative integers that are *not* surjective:
     – "addition of 1" is not surjective, because, e.g., 0 is not "1 greater" than any nonnegative integer.
     – "squaring" is not surjective, because, e.g., 2 is not the square of any integer. (We prove this as Proposition 4.3.)

   A surjective function F is called a *surjection*.

3. *one-to-one, onto* (or *bijective*) if for each $t \in T$, there is precisely one $s \in S$ such that $F(s) = t$.

A bijective function F is called a *bijection*. When F is a bijection from S onto T, we often write $F : S \leftrightarrow T$.

There is a marvelous theorem that must be mentioned here, even though it is beyond the scope of the book.[7] The theorem says that, given sets S and T: *if* there is an injection F_1 that maps elements of S one-to-one to elements of T *and* there is an injection F_2 that maps elements of T one-to-one to elements of S, *then* there is a single bijection F such that

- F maps elements of S one-to-one to elements of T;
- F^{-1}, the *functional inverse* of F, maps elements of T one-to-one to elements of S.

Theorem 3.2 (The Schröder-Bernstein Theorem). *For any sets S and T, if there is an* injection $F^{(S \to T)} : S \to T$ *and an* injection $F^{(T \to S)} : T \to S$, *then there is a* bijection $F^{(S \leftrightarrow T)} : S \to T$.

While the operation of *composition*, as introduced in Section 3.2.1, is important for general binary relations, it is a daily staple with relations that are functions! Let us be given two functions on a set S

$$F : S \to S \quad \text{and} \quad G : S \to S$$

The composition of F and G, *in that order*, is the function

$$F \circ G : S \to S$$

defined as follows.

$$\text{For each } s \in S \quad F \circ G(s) = G(F(s)). \tag{3.4}$$

The unexpected change in the orders of writing F and G on the two sides of equation (3.4) results from the existence of two historical schools that both contributed to the formulation of this material. One school cleaved to the tradition of *abstract algebra*; they wanted all expressions to be written with all operators—including the composition operator ∘—in infix notation. Another school, which could be called *applicative algebraic*, wanted to view functions as "applying" to their arguments. Both notations have significant advantages in certain contexts, so both have survived. It is a good idea for neophyte readers to be prepared to encounter both notations—but they have to keep their eyes open regarding the relative orders of F and G.

Before progressing with new material, it is worth taking a moment to verify that the important operation of composition behaves the way one would want and expect it to—by preserving the type of function being composed.

Proposition 3.2 *Let us be given functions $F : S \to S$ and $G : S \to S$ on the set S, together with their composition $F \circ G$, as defined in (3.4).*

---

[7] The theorem can be found in texts such as [5].

(a) *F ∘ G is a function on S.*
(b) *If F and G are injections, then so also is F ∘ G.*
(c) *If F and G are surjections, then so also is F ∘ G.*
(d) *If F and G are bijections, then so also is F ∘ G.*

Proof. We prove each of the four assertions by invoking the underlying definitions.
(a) Because F and G are functions on S: For each $s \in S$, there is at most one $t_1 \in S$ such that $F(s) = t_1$ and at most one $t_2 \in S$ such that $G(s) = t_2$. Hence, we identify three possibilities.

if F is defined at $s \in S$	**then** $F(s) \in S$ is unique
and if G is defined at $F(s) \in S$	**then** $G(F(s)) = F \circ G(s) \in S$ is unique
if F is not defined at $s \in S$	**then** $F \circ G$ is not defined at $s \in S$
if F is defined at $s \in S$	**then** $F(s) \in S$ is unique
and if G is not defined at $F(s) \in S$	**then** $F \circ G$ not defined at $s \in S$

Hence, for each $s \in S$, there is at most one $t \in S$ such that $F \circ G(s) = t$; in other words, $F \circ G$ is a function on S.

(b) Focus on any $s \in S$. Because F is an injection on S, there exists at most one $t \in S$ such that $F(t) = s$. Because G is an injection on S, there exists at most one $u \in S$ such that $G(u) = t$. Thus, there exists at most one $u \in S$ such that $F \circ G(u) = s$. Hence, $F \circ G(u)$ is an injection on S.

(c) Focus on any $s \in S$. Because F is a surjection on S, there exists $t \in S$ such that $F(t) = s$. Because G is a surjection on S, there exists $u \in S$ such that $G(u) = t$. This means, however, that $F \circ G(u) = s$. Because $s \in S$ was arbitrary, it follows that $F \circ G(u)$ is a surjection on S.

(d) If each of F and G is a bijection on S, then each is an injection on S, and each is a surjection on S. Then Part (b) tells us that $F \circ G$ is an injection on S, and Part (c) tells us that $F \circ G$ is a surjection on S. Hence, $F \circ G$ is a bijection on S.   □

3.3 Boolean Algebras

A *Boolean algebra* is a mathematical system **HERE

3.3.1 The Boolean Operations

3.3.2 The Axioms of Boolean Algebras

3.3.3 Two Special Boolean Algebras

3.3.3.1 The Algebra of Sets

3.3.3.2 Propositional Logic as an Algebra: The Propositional Calculus

A. The basic logical connectives. The Boolean set-related operations we discussed in
Section 3.1.2 have important analogues within the context of Propositional logic.
logical analogues of these operations for logical sentences and their logical *truth
values*, TRUE and FALSE, often denoted 1 and 0, respectively:

- logical not ($\sim$) The operation not is the logical analogue of the set-theoretic op-
  eration of complementation. Because always writing "not" makes logical expres-
  sions long and cumbersome, we usually use the prefix-operator $\sim$ to denote not;
  i.e., we write
  $$\sim P \quad \text{rather than the more cumbersome} \quad \text{not } P$$
  to denote the logical complementation of proposition *P*. Whichever notation we
  use, the defining properties of logical complementation are encapsulated in the
  following pair of equations

  $$[\sim \text{TRUE} = \text{FALSE}] \quad \text{and} \quad [\sim \text{FALSE} = \text{TRUE}].$$

- logical or ($\vee$) The operation or—which is also called *disjunction* or *logical
  sum*—is the logical analogue of the set-theoretic operation of union. For con-
  venience and brevity, we usually use the infix-operator $\vee$ to denote or in expres-
  sions. Whichever notation we use, the defining properties of logical disjunction
  are encapsulated as follows.

  $$[[P \vee Q] = \text{TRUE}] \quad \text{if, and only if,} \quad [P = \text{TRUE}] \text{ or } [Q = \text{TRUE}] \text{ or both.}$$

  Note that, as with union, logical or is *inclusive:* The assertion
  $$[P \vee Q] \text{ is TRUE}$$
  is true when *both P* and *Q* are true, as well as when only one of them is. Be-
  cause such inclusivity does not always capture one's intended meaning, another,
  exclusive version of disjunction also exists, as we see next.
- logical xor ($\oplus$) The operation *exclusive or*—which is also called xor—is a ver-
  sion of disjunction that does not allow both disjuncts to be true simultaneously.
  For convenience and brevity, we usually use the infix-operator $\oplus$ to denote xor in
  expressions. Whichever notation we use, the defining properties of exclusive or
  are encapsulated as follows.

$$[[P \oplus Q] = \text{TRUE}] \quad \text{if, and only if,} \quad [P = \text{TRUE}] \text{ or } [Q = \text{TRUE}] \text{ \textit{but not} both.}$$

To emphasize the distinction between $\vee$ and $\oplus$: The assertion
$$[P \oplus Q] \text{ is TRUE}$$
is *false* when *both P* and *Q* are true.

- logical and ($\wedge$) The operation and—which is also called *conjunction* or *logical product*—is the logical analogue of the set-theoretic operation of intersection. For convenience and brevity, we usually use the infix-operator $\wedge$ to denote and in expressions. Whichever notation we use, the defining properties of logical conjunction are encapsulated as follows.

$$[[P \wedge Q] = \text{TRUE}] \quad \text{if, and only if, \textit{both}} \quad [P = \text{TRUE}] \text{ and } [Q = \text{TRUE}]$$

- logical implication ($\Rightarrow$) The logical operation of implication, which is often called *conditional* and which we usually denote in expressions via the infix-operator $\Rightarrow$ differs from the other logical operations we have discussed in a way that the reader must always keep in mind. In contrast to not, or, and and, whose formal versions pretty much coincide with their informal versions, the formal version of implication, being formal, fixed, and precise, carries connotations that we do not always associate with the informal word "implies." The formal version of implication is defined as follows.

$$[[P \Rightarrow Q] = \text{TRUE}] \quad \text{if, and only if,} \quad [[\sim P] = \text{TRUE}] \text{ (inclusive) or } [Q = \text{TRUE}]$$

This definition means, in particular, that

- If proposition *P* is false, then it implies *every* proposition.
- If proposition *Q* is true, then it is implied by *every* proposition.

- logical equivalence ($\equiv$) The final logical operation in our toolbox is variously called *equivalence,* as in:
  Proposition *P* is (logically) equivalent to Proposition *Q*
  and *biconditional*. We usually denote the operation in expressions via the infix-operator $\equiv$. The operation is defined as follows.

$$[[P \equiv Q] = \text{TRUE}] \quad \text{if, and only if } [[P \Rightarrow Q] = \text{TRUE}] \quad \text{and} \quad [[Q \Rightarrow P] = \text{TRUE}]$$

We often use the term "connective" rather than "operation" to refer to what we have here called the logical operations of the Propositional Calclulus. This is because one often feels that logical propositions are static statements rather than active prescriptions for computations.

B. The logical connectives via truth tables If one is willing to view statements in the Propositional Calclulus as prescriptions for computations, then one can encapsulate the definitions of the basic logical connectives via functions that map logical expressions int the truth values true, or 1, and false, or 0. The following tables reproduce the definitions of Subsection A within this computational framework.

P	Q	$P \vee Q$	$P \oplus Q$	$P \wedge Q$	$P \Rightarrow Q$	$P \equiv Q$
0	0	0	0	0	1	1
0	1	1	1	0	1	0
1	0	1	1	0	0	0
1	1	1	0	1	1	1

P	$\sim P$
0	1
1	0

(3.5)

Note how the truth-value entries in the righthand truth table of (3.5) reinforce the lessons of Subsection A. This awareness is especially essential with the implication-related operations ($\Rightarrow$ and $\equiv$) that have informal counterparts in "real-life" reasoning. Think about whether your intuition agrees with the formal specifications of these operations.

C. The (Boolean) algebra of logical operations   The operations of the Propositional Calclulus give rise to a Boolean algebra that manipulates (logical) propositions in much the way that the set-oriented Boolean algebra of Section 3.3.3.1 manipulates sets.

D. Logic via truth values As demonstrated in Subsection C, the Propositional Calculus is a Boolean Algebra when one interprets the logical connectives of Subsection A as operations on logical expressions that evaluate to either TRUE (or, 1) or FALSE (or, 0). But, the calculus forms a very special genre of Boolean Algebra, namely, a *free Boolean Algebra*.

The meaning of the qualifier *free* is easy to state but not so easy to understand. Here is a very informal try.

Any sort of algebra is defined as a set of objects accompanied by a set of basic operations, wherein the operations obey certain "self-evident" axioms.[8] In rough, but evocative, terms, the algebra is *free* if its operations do not relate to one another in any way that is not mandated by the axioms.

An easy illustration of this decidedly *not easy* concept is the following. There is a (quite important) genre of algebra called a *semi-group*. A semi-group $\mathscr{S}$ is specified via a set (of objects) S, together with a binary operation on S, which we denote $\oplus$. For the (algebraic) structure $\mathscr{S} = \langle S, \oplus \rangle$ to be a semi-group, it must obey the following two axioms.

1. $\mathscr{S}$ must be *closed* under operation $\oplus$; i.e., for all $s_1, s_2 \in S$, we must have

$$(s_1 \oplus s_2) \in S.$$

2. The operation $\oplus$ must obey the *associative law;* i.e., for all $s_1, s_2, s_3 \in S$, we must have

$$(s_1 \oplus s_2) \oplus s_3 \;=\; s_1 \oplus (s_2 \oplus s_3)$$

Now, it is easy to verify—we shall discuss this in Chapter 4—that the (algebraic) structure formed by the integers coupled with the operation of addition forms a semi-group (which the notation of Chapter 4 would denote $\langle \mathbb{Z}, + \rangle$). But this semi-group is *not* free, because integer addition obeys laws beyond just closure and associativity, namely:

1. Addition of integers is *commutative*.
2. Integer addition has an *identity*, namely 0.

---

[8] Indeed, that is what the word "axiom" means.

3. Every integer $z \in \mathbb{Z}$ has an *additive inverse* (namely, $-z$).

So, it really is meaningful that the Boolean Algebra built upon the Porpositional calculus *is* free.

For us, the impact of the "freeness" of the Propositional algebra, *qua* Boolean Algebra, is manifest in the following *meta-theorem*[9] which is discussed and proved in [90].

Theorem 3.3. *A propositional expression $E(P, Q, \ldots, R)$ is a theorem of Propositional logic if, and only if, it to* TRUE *under all truth assignments to the propositions $P, Q, \ldots, R$.*

Proving Theorem 3.3 is beyond the scope of this book, but we now present a few illustrative instantiations, annotated to suggest their "real-world" messages. Each example is accompanied by a "real-life" interpretation and a verifying truth table.

The law of double negation
This is a formal analogue of the homely adage that "a double negative is a positive."

Proposition 3.3 *For any proposition P,*

$$P \;\equiv\; \sim [\sim P]$$

Proof via truth table.

P	$\sim P$	$\sim [\sim P]$
0	1	0
1	0	1

(3.6)

Note that columns 1 and 3 of truth table (3.6) are identical. By Theorem 3.3, this fact verifies Proposition 3.3, the law of double negation. $\square$

The law of contraposition
This is a very exciting example! Let us immediately convert this to a mathematical statement about the Boolean Algebra of Propositional logic and then prove the statement. We shall then contemplate the implications of this law for *logic* rather than *mathematics*.

Proposition 3.4 *For any propositions P and Q,*

$$[[P \Rightarrow Q] \;\equiv\; [\sim Q \Rightarrow \sim P]]$$

Proof via truth table.

---

[9] A *theorem* exposes some truth within the mathematical structure being discussed. A *meta-theorem* exposes some truth about the way the discussion can proceed.

P	$\sim P$	Q	$\sim Q$	$P \Rightarrow Q$	$\sim Q \Rightarrow \sim P$
0	1	0	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	1

(3.7)

Note that columns 5 and 6 of truth table (3.7) are identical. By Theorem 3.3, this fact verifies Proposition 3.4, the law of contraposition.   □

Now let us reconsider the whole concept of contraposition, including this law, in the light of *logic and reasoning*.

asserts that the assertion

"Proposition Q implies Proposition Q"

is *logically equivalent* to the assertion

"the negation of Proposition Q implies the negation of Proposition P".

Think about this! In any system of reasoning in which a given proposition is either true or false

De Morgan's Laws:

Proposition 3.5 *For any propositions P and Q:*

- $[P \wedge Q] \equiv \sim [[\sim P] \vee [\sim Q]]$
- $[P \vee Q] \equiv \sim [[\sim P] \wedge [\sim Q]]$

Proof via truth table.

P	$\sim P$	Q	$\sim Q$	$[P \wedge Q]$	$[\sim P] \vee [\sim Q]$	$[P \vee Q]$	$[\sim P] \wedge [\sim Q]$
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	1
1	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0

(3.8)

Note that columns 5 and 6 of truth table (3.8) are mutually complementary, as are columns 7 and 8. If we negate (or, complement) the entries of columns 6 and 8, then we can invoke Theorem 3.3 to verify proposition 3.5, which encapsulates De Morgan's laws for Propositional logic.   □

The distributive laws for Propositional logic

In numerical arithmetic, multiplication distributes over addition, but not conversely, so we have a single distributive law for arithmetic (see Section 4.2.2). In contrast, each of logical multiplication and logical addition distributes over the other, so we have two distributive laws for Propositional logic.

- $P \vee [Q \wedge R] \equiv [P \vee Q] \wedge R$
- $P \wedge [Q \vee R] \equiv [P \wedge Q] \vee R$

P	Q	R	$[P\vee Q]$	$[P\wedge Q]$	$[Q\wedge R]$	$[Q\vee R]$	$P\vee[Q\wedge R]$	$[P\vee Q]\wedge[P\vee R]$	$P\wedge[Q\vee R]$	$[P\wedge Q]\vee[P\wedge R]$
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0
0	1	1	1	0	1	1	1	1	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	1	0	1	0	1	1	1	1
1	1	0	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

$$(3.9)$$

Note that columns 8 and 9 of truth table (3.9) are identical, as are columns 10 and 11. By Theorem 3.3 this fact verifies the distributive laws for Propositional logic.

3.3.4 Connecting Mathematical Logic with Logical Reasoning

3.3.4.1 A formal notion of *implication*, and its implications

In everyday discourse, we all employ an intuitive notion of *implication*. When we make the assertion

Proposition *A implies* Proposition *B*

what we usually have in mind is

If Proposition *A* is true, then Proposition *B* istrue.

But this, or any, homespun meaning of the word/concept "implies" raises many questions.

- What if Proposition *A* is *not* true? Are there any inferences we can draw?
- Is there any relation between the assertion

Proposition *A implies* Proposition *B*

and its *converse* assertion

Proposition *B implies* Proposition *A*?

- If we know that Proposition *B* is true, shouldn't it be "implied by every other proposition—perhaps even a *false* one?

This section is devoted to discussing the *formal* notion of implication that was adopted by mathematical philosophers in the 19th century. The *advantage* of having such a formal notion is that it will answer all questions of the sort we have just posed. The *disadvantage* of having such a formal notion is that the way in which the notion answers some of our questions may be rather counter to one's untutored intuition.

Chapter 4
ARITHMETIC

Many, perhaps most, of us take for granted the brilliant notations that have been developed for the myriad arithmetic constructs that we use daily. Our mathematical ancestors have bequeathed us notations that are not only perspicuous but also convenient for computing and for discovering and verifying new mathematical truths. This chapter is dedicated to sharing this legacy with the reader.

4.1 Numbers and Numerals

Every reader will be familiar with the notion of *number* and with the familiar strings, called *numerals*, that name numbers within positional number systems.

> Numbers and numerals embody what is certainly the most familiar instance of a very important dichotomy that pervades our intellectual lives: the distinction between objects and their names:
>
> *Numbers are objects. Numerals are the names we use to refer to and manipulate numbers.*
>
> This is a crucially important distinction! You can "touch" a numeral: break it into pieces, combine two (or more) numerals via a large range of operations. Numbers are intangible abstractions: you cannot compute with them.

In our daily commerce, we typically deal with numerals formed within a *base-b positional number system,* i.e., by strings of *digits*, often embellished with other symbols, such as a *radix point*. We describe such systems in detail in Section 4.1.2.A. For now, we settle for a few examples: 123456789 (base 10, or, *decimal*), 10111.001 (base 2, or, *binary*), or 0.1267×8^{24} (base 8, or, *octal*).[1]

---

[1] The use of a period as the radix point is a US convention; in much of Europe, a comma denotes the radix point.

4.1.1 Numbers

We begin our study of arithmetic notions with a short taxonomy of numbers, the objects that arithmetic was invented to explicate and exploit. Although we assume that the reader is familiar with the most common classes of numbers, we do spend some time highlighting important features of each class, partly, at least, in the hope of heightening the reader's interest in this most basic object of mathematical discourse.

We present the four most common classes of numbers in what is almost certainly the chronological order of their discovery/invention.

> Did humans invent these classes of numbers to fill specific needs, or did we just discover their pre-existing selves as needs prompted us to search for them? The great German mathematician Leopold Kronecker, as cited on page 477 of [3], shared his viewpoint on this question: "God made the integers; all else is the work of man."

A pleasing narrative can be fabricated to account for our multi-class system of numbers. In the beginning, the story goes, we needed to count things (sheep, bottles of oil, weapons, . . .), and the positive *integers* were discovered. As accounting practices matured, we needed to augment this class with both zero (0) and the negative integers—and the full class of integers was born. As society developed, we had to start sharing subdivisible materials, so we needed to invent the *rational numbers*. Happily for the mathematically inclined, the rational numbers could be developed in a way that allowed one to view an integer as a special type of rational.

> This quest for a single encompassing framework rather than a set of isolated concepts is a hallmark of mathematical thinking.

As the ancient Greeks (so the story goes) were inventing geometry and its place within architecture, they encountered the uncomfortable fact that the lengths of portions of eminently buildable structures were not "measurable," by which they meant "not rational." The poster child for this phenomenon was the hypotenuse of the isosceles right triangle with unit-length legs. As a response to this discomfort, the *real numbers* were invented. Once again, happily, one could develop the real numbers in a way that allowed one to view a rational number as a special type of real number. Time went on, and mathematics matured. Polynomials and their roots[2] were discovered, together with the next source of discomfort. To understand this discomfort, one must note that, since the invention of the real numbers, every polynomial $P_m(x) \stackrel{\text{def}}{=} x^2 - m$, where m is a nonnegative real number, had two roots, denoted, respectively, $+\sqrt{m}$ and $-\sqrt{m}$. But—here is the source of discomfort—*the polynomial $P_m(x)$ had no roots when m was negative, even a negative integer*. The response this time resided in the invention of a new *imaginary* number, called i, that was a root of the polynomial $P_{-1}(x) = x^2 + 1$; i was often defined via the equation, $i = \sqrt{-1}$; easily, $-i$ is also a root of $P_{-1}(x)$. By combining the imaginary number i with the real number system, the *complex numbers* were born.

---

[2] A number r is a *root* of a polynomial $P(x)$ if $P(r) = 0$.

The reader is possibly anticipating a new "discomfort" leading to an new augmentation of our number system, but, no, the story is now complete, in the sense expressed in the *Fundamental Theorem of Algebra*, which asserts:

Theorem 4.1 (The Fundamental Theorem of Algebra). *Every polynomial of degree n with complex coefficients has n roots over the complex numbers.*

We return to the later to the Theorem and its applications and implications. For now, though, it completes our historical tour, so we can finally begin to get acquainted with our four classes of numbers.

4.1.1.1 The integers

The most basic class of numbers are the *integers* (or, *whole numbers*, or, *counting numbers*). Integers are certainly the numbers that our prehistoric ancestors employed in the earliest days of our species.

A. Integers and their number line. We survey a number of the most important properties of the set $\mathbb{Z}$ that comprises *all integers* (the positive and negative integers and zero (0)) and the set $\mathbb{N}$ that comprises the *nonnegative integers* (the positive integers and 0). We estimate a property's importance from the vantage points of both mathematics and its manifold applications.

Several essential properties of $\mathbb{N}$ and $\mathbb{Z}$ are consequences of the sets' behavior under their natural order relations: strong ($<$) and weak ($\leq$) and their converses. Indeed, order within a number system is one of one's biggest friends when reasoning about the numbers within the system.

- The set $\mathbb{Z}$ is *totally ordered*, also termed *linearly ordered*.

  This fact is embodied in the *Trichotomy Laws for integers*.

  The Trichotomy laws for integers.
  (a) *For each integer $a \in \mathbb{Z}$, precisely one of the following is true.*

     (1) *a* equals 0: $a = 0$ (2) *a* is *positive*: $a > 0$ (3) *a* is *negative*: $a < 0$

  Consequently, $\mathbb{Z}$ can be visualized via the (2-way infinite) number line: $\ldots, -2, -1, 0, 1, 2, \ldots$. Analogously, $\mathbb{N}$ can be visualized via the (1-way infinite) number line: $0, 1, 2, \ldots$.

  (b) *For any integers $a, b \in \mathbb{Z}$, precisely one of the following is true.*

  $$(1)\ a = b \qquad (2)\ a < b \qquad (3)\ a > b$$

- The set $\mathbb{N}$ is *well-ordered*.

  The Well-ordering law for nonnegative integers.
  Every subset of $\mathbb{N}$ has a smallest element (under the ordering $<$).

- The set $\mathbb{Z}$ is *discrete*.

  The discreteness of the integers.
  For every integer $a \in \mathbb{Z}$, there is no integer between a and $a+1$; i.e., there is no $b \in \mathbb{Z}$ such that $a < b < a+1$.

- The set $\mathbb{Z}$ obeys the *"Between" Laws*

  The "Between" laws for integers.
  For any integers $a,b \in \mathbb{Z}$, there are finitely many $c \in \mathbb{Z}$ such that $a < c < b$.

  Any such $c \in \mathbb{Z}$ is *between a* and *b*, whence the name of the law.

B. Prime numbers We single out a subclass of the positive integers whose mathematical importance has been recognized for millennia but which have found important applications (e.g., within the domain of computer security) mainly within the past several decades.

Positive integer p *divides* positive integer n (or, *is a divisor of n*) if there is a positive integer q such that $n = p \cdot q$. Equivalently, we say that *n is divisible by p*. Thus, every positive integer n is divisible by $p = 1$ ($q = n$) and by $p = n$ ($q = 1$).

The class of integers we single out is defined by its divisibility characteristics.

An integer $p > 1$ is *prime* if its *only* positive integer divisors are 1 (which divides every integer) and itself (which is always a divisor).

> We usually use the shorthand assertion, "*p* is a prime," instead of the longer, but equivalent, "*p* is a prime integer."

A very important way to classify a positive integer n is to list the primes that divide it, coupling each such prime p with its *multiplicity*, i.e., the number of times that p divides n. Let $p_1, p_2, \ldots, p_r$ be the distinct primes that divide n, and let each p_i divide n with multiplicity m_i. The *prime factorization* of n is the product $p_1^{m_1} \times p_2^{m_2} \times \cdots \times p_r^{m_r}$; note that this product satisfies the equation

$$n = p_1^{m_1} \times p_2^{m_2} \times \cdots \times p_r^{m_r} \tag{4.1}$$

When writing an integer n's prime factorization, it is traditional to list the primes $p_1, p_2, \ldots, p_r$ in increasing order, i.e., so that $p_1 < p_2 < \cdots < p_r$.
A positive integer n is totally characterized by its canonical prime factorization, as attested to by the following classical theorem, which has been known for millennia and has been honored with the title *The Fundamental Theorem of Arithmetic*. We state the Theorem in two equivalent ways which suggest different ways of thinking about the result.

Theorem 4.2 (The Fundamental Theorem of Arithmetic).
(Traditional formulation.) *The canonical prime factorization of every positive integer is unique.*
(Alternative formulation.) *Let $n \in \mathbb{N}^+$ be a positive integer, and let $\widehat{P}_n$ denote the ordered sequence of prime numbers that are no larger than n:*

$$\widehat{P}_n \;=\; \langle P_1,\, P_3,\, \ldots,\, P_{r-1},\, P_r \rangle$$
$$\textit{where: } P_1 \;=\; 2$$
$$\textit{each } \; P_i \;<\; P_{i+1}$$
$$P_r \;\leq\; n.$$

There exists a unique sequence of nonnegative *integers,* $\langle a_1, a_2, \ldots, a_r \rangle$ *such that*

$$n \;=\; \prod_{i=1}^{r} P^{a_i} \;=\; P^{a_1} \cdot P^{a_2} \cdot \;\cdots\; \cdot P^{a_{r-1}} \cdot P^{a_r}$$

A simple, yet important, corollary of Theorem 4.2 is the following result, whose proof we leave to the reader.

Proposition 4.1 *Every positive integer is divisible by at least one prime number.*

The proof of Theorem 4.2 goes beyond the scope of this book. We do discuss a number of applications of this fundamental result. We have already seen one application in Proposition 2.4.

One very important application of Theorem 4.2 is as a mechanism for *encoding* sequences of positive integers as single integers! This works as follows. Consider the (infinite) ordered sequence of *all primes:*

$$(P_1 = 2), (P_2 = 3), (P_3 = 5), \ldots$$

Let

$$\bar{s} \;=\; \langle m_1, m_2, \ldots, m_k \rangle \tag{4.2}$$

be an arbitrary sequence of positive integers. Then Theorem 4.2 assures us that the (single) positive integer

$$\iota(\bar{s}) \;\stackrel{\text{def}}{=}\; P_1^{m_1} \cdot P_2^{m_2} \cdot \;\cdots\; \cdot P_k^{m_k}$$

is a (uniquely decodable) integer-representation of sequence $\bar{s}$.

We return to this idea of encoding-via-integers in a later chapter.

4.1.1.2 The rational numbers

A. Inventing the rational numbers. Each augmentation of our number system throughout history has been a response to a deficiency with the then-current system. The deficiency that instigated the introduction of the rational numbers was the frequency with which a given integer q does not divide another given integer p. This fact meant that the operation of multiplication could often not be "undone", or "inverted", because division, the operation that would accomplish the undoing/inverting, would be a *partial operation:* There are pairs of integers neither of which divides the other. The defining characteristic of the expanded system is that, within it, every nonzero number divides every number.

B. The rationals as ordered pairs of integers. The set $\mathbb{Q}$ of *rational* numbers consists of the number 0, plus the ratios p/q of all nonzero integers:

$$\mathbb{Q} \overset{\text{def}}{=} \{0\} \cup \{p/q \mid p,q \in \mathbb{Z} \setminus \{0\}\}$$

Each element of $\mathbb{Q}$ is called a *rational* number; each *nonzero* rational number p/q is often called a *fraction*, especially when $q > p$.

An alternative, mathematically more advanced, way of defining the set $\mathbb{Q}$ is to view it as the smallest set of numbers that contains the integers, i.e., the set $\mathbb{Z}$, and is *closed under the operation of dividing any number by any nonzero number.* The word "closed" here means that, given any two numbers in $\mathbb{Q}$, call them r and $s \neq 0$, their quotient r/s belongs to $\mathbb{Q}$.

Numerous notations have been developed for "naming" rational numbers in terms of the integers they are "built from." Most of these notations continue our custom of employing the symbol "0" for the number 0. For the nonzero elements of $\mathbb{Q}$, we traditionally employ some notation for the operation of division and denote the

$$p/q \quad \text{or} \quad \frac{p}{q} \quad \text{or} \quad p \div q \tag{4.3}$$

The integer p in any of the expressions in (4.3) is the *numerator* of the fraction; the integer q is the *denominator*.

C. Comparing the rationals with the integers. There are many ways to compare the sets $\mathbb{Z}$ and $\mathbb{Q}$ via properties that enhance our understanding of the two sets.

i. Every integer is a rational. Obviously, every integer $n \in \mathbb{Z}$ can be viewed as a rational number, namely, the rational p/q whose numerator is $p = n$ and whose denominator is $q = 1$; i.e., $n = n/1$. Easily, this encoding preserves the special character of the numbers 0 and 1, because $0/1 = 0$ and $1/1 = 1$.

ii. The rational and integer number lines. The rational numbers share some, but not all, of the number-line laws enumerated in Section 4.1.1.1.A. We mirror for $\mathbb{Q}$ that section's discussion of $\mathbb{Z}$.

- *The Trichotomy laws for rational numbers.*
  (a) *For each rational $a \in \mathbb{Q}$, precisely one of the following is true.*

  (1) a equals 0: $a = 0$ (2) a is *positive*: $a > 0$ (3) a is *negative*: $a < 0$

  Consequently, $\mathbb{Q}$ can be visualized via the (2-way infinite) number line.

  (b) *For any rationals $a, b \in \mathbb{Q}$, precisely one of the following is true.*

  $$(1) \, a = b \qquad (2) \, a < b \qquad (3) \, a > b$$

- *The set $\mathbb{Q}$ is* not *well-ordered.*
  For illustration: The set

$$S = \{a \in \mathbb{Q} \mid 0 < a \leq 1\}$$

has no smallest element. If you give me a rational $p \in S$ that you claim is the smallest element of the set, then I shall give you $p/2$ as a smaller one.

- *The set $\mathbb{Q}$ does* not *obey the "Between" laws.*
  In fact, $\mathbb{Q}$ violates the "Between" laws in a very strong way: *For any two unequal rationals, a and b > a, there are infinitely many rationals between a and b.*
  One can specify such an infinite set for the pair a,b in myriad ways. Here is a simple such set, call it $S_{a,b}$.

$$S_{a,b} = \left\{ \frac{a+b}{k} \ \middle| \ k \in \mathbb{Z} \right\}. \tag{4.4}$$

iii. The relative "sizes" of $\mathbb{Z}$ and $\mathbb{Q}$: Are there more rationals than integers?
Consider the following facts.

- Every integer is a rational number, as attested to by the "encoding"

$$\text{Encode} \quad n \in \mathbb{Z} \quad \text{by} \quad \frac{n}{1} \in \mathbb{Q}. \tag{4.5}$$

- There are infinitely many non-integer rational numbers between every pair of adjacent integers, as attested to by every set $S_{n,n+1}$ as defined in (4.4).

Represented symbolically, we thus have $\mathbb{Z} \subset \mathbb{Q}$.

The preceding "argument" makes an intuitive case that
  "there are more rational numbers than integers."
We put this assertion in quotes because, absent a formal notion of "*more*" for infinite sets, the assertion is meaningless! Is there an acceptable formal notion that would allow us to verify or refute the assertion? The 19th-century mathematician/logician Georg Cantor addressed this question, and related ones, in his groundbreaking study of the relative "sizes" of infinte sets [9, 10]. We adapt enough of his formulation to suggest how such issues can be dealt with mathematically. We shall take a longer look at Cantor's work in Section 4.1.1.4

Let us take our lead from finite sets. Is there a notion of "bigger" for finite sets that can be extended to infinite sets?

We begin with a set A of apples and a set O of oranges, together with challenge of determining which set is bigger.

If sets A and O are both finite, then we can just *count* the number of apples in A, call it a, and the number of oranges in O, call it o, and then compare the sizes of the (nonnegative) integers a and o. The Trichotomy Laws for integers (Section 4.1.1.1.A) guarantee that we shall be able to settle the question. *But* we cannot count the elements in an infinite set, so this approach fails us when we can infinitely much fruit!

Here is another approach that works for finite sets and that promises to extend to infinite sets. Let us assume that we can "prove"—we shall explain the word imminently—the following.

For every apple that we extract from set *A for the first time*, we can extract an orange from set *O for the first time*. It will then follow (at least in the finite case), that

> *There are at least as many oranges as apples!*

This is really promising, because there is another way to describe the fruit-matching process that readily extends to infinite sets.

> *There is an injection,[3] call it f, from O to A.*

In more formal terms: *Every time you pull an apple α from set A, I pull the orange $f^{-1}(\alpha)$ from O.*

Inspired by this formulation using injections—and by the work of Cantor—we craft the following definition.

Given sets A and O (finite or infinite), we say

> Set *O* is at least as big as set *A*

precisely when there is an injection from O to A.

Finally, back to numbers!

Proposition 4.2 (a) *There exists an injection from $\mathbb{Z}$ to $\mathbb{Q}$. Therefore, $\mathbb{Q}$ is at least as big as $\mathbb{Z}$.*
(b) *There exists an injection from $\mathbb{Q}$ to $\mathbb{Z}$. Therefore, $\mathbb{Z}$ is at least as big as $\mathbb{Q}$.*

Proof. (a) We have already established part (a), via the injection from $\mathbb{Z}$ into $\mathbb{Q}$ implicit in (4.5).

(b) We proceed in two steps.

1. We define an injection f_1 that maps $\mathbb{Q}$ one-to-one into the set $\mathbb{Q}^+$ of *nonnegative* rationals. We specify f_1 as follows.

$$f_1(p/q) \;=\; \begin{cases} 2p/q & \text{if } p/q \geq 0 \\ (2p+1)/q & \text{if } p/q < 0. \end{cases}$$

   We leave to the reader the easy proof that f_1 is an injection from $\mathbb{Q}$ into $\mathbb{Q}^+$.
2. We define an injection from $\mathbb{Q}^+$ into $\mathbb{Z}$.
   Toward this end, we represent each rational number $r \in \mathbb{Q}^+$ by a pair of integers $p > 0$ and $q > 0$ such that

   - $r = p/q$
   - p and q are *relatively prime* in the sense that no integer $n > 1$ divides both p and q. If such an n existed, then we could divide both p and q by it, with the assurance that

     $$\frac{p/n}{q/n} \;=\; \frac{p}{q}.$$

   Inspired by Theorem 4.2, we now consider the function

   $$f_2(p/q) \;\stackrel{\text{def}}{=}\; 2^p 3^q.$$

---

[3] Recall from Section 3.2.4 that "*injection*" is synonymous with "*one-to-one function*".

Thus defined, f_2 is a function from $\mathbb{Q}^+$ into $\mathbb{Z}$. Moreover, Theorem 4.2 guarantees that f_2 maps $\mathbb{Q}^+$ *one-to-one* into $\mathbb{Z}$.

Since the composition of two injections is also an injection (see Proposition 3.2), the result is proved. $\square$

4.1.1.3 The real numbers

A. Invemting the real numbers. Each subsequent augmentation of our system of numbers inevitably gets more complicated than the last: one solves the easy problems first. The deficiency in the system of real numbers harkens back to historical time, roughly $2\frac{1}{2}$ millennia ago. The ancient Egyptians were prodigious builders who mastered truly sophisticated mathematics in order to engineer their temples and pyramids. The ancient Greeks perpetuated this engineering tradition, but they added to it the "soul" of mathematics.

Numbers were (literally) sacred objects to the Greeks, and they invented quite "modern" (to our perspective) ways of thinking about mathematical phenomena in order to understand *why* certain facts were true, in addition to knowing *that* they were true. One intellectual project in this spirit had to do with the way they designed contructions. They were attracted to geometric contructions that could be accomplished using only *straight-edges and compasses*. And—most relevant to our story—they preferred that the relative lengths of linear sections of their structures be *commensurable*, in the following sense. *Integers* $x, y \in \mathbb{N}$ *are* commensurable *if there exist* $a, b \in \mathbb{N}$ *such that*

$$ax \;=\; by \quad \text{or, equivalently,} \quad x \;=\; \frac{a}{b}y.$$

The desire to employ only commensurable pairs of integers, at least in moderately simple constructions, was shown to be impossible when one considered *the diagonal of the square with unit-length sides* or, equivalently, *the hypotenuse of the isosceles right triangle with unit-length legs*. In both situations, one encountered the unit lengths of the sides or legs of the structures, together with the *noncommensurable* length of the diagonal or hypotenuse, which, in current terminology, is $\sqrt{2}$. The Greek mathematicians, as reported by the renowned mathematician Euclid,[4] proved, using current terminology, that $\sqrt{2}$ is not rational. (We rephrase the proof imminently, in Proposition 4.3.) The conclusion from this proof is that the number system based on the rational numbers was inadequate. In response, they augmented this system by introducing *surds* or, as we more commonly term them, *radicals*, beginning a trajectory that culminated in the real number system. Since our intention has been to justify the journey along that trajectory, we leave our historical digression and turn to our real focus, the set $\mathbb{R}$ of *real numbers*.

B. The real numbers via their numerals. For any integer $b > 1$, the real numbers are the numbers that can be named by em infinite strings built out of the digits

---

[4] who wrote extensively on this and related subjects.

$\{0, 1, \ldots, b-1\}$;[5] the resulting strings are called *b-ary numerals*. There are a couple of ways to form *b*-ary numerals; we shall discuss some of the most common ones in Section 4.1.2. For now, we define real numbers as those that can be represented by a base-*b* numeral, for some integer $b > 1$. Such a numeral has the form

$$\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0 . \beta_0 \beta_1 \beta_2 \cdots \tag{4.6}$$

and represents the (real) number

$$\underline{\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0 . \beta_0 \beta_1 \beta_2 \cdots} \overset{\text{def}}{=} \sum_{i=0}^{n} \alpha_i \cdot b^i \; + \; \sum_{j \geq 0} \beta_j \cdot b^{-j}.$$

By prepending a "negative sign" (or, "minus sign") $-$ to a numeral or a number, one renders the thus-embellished entity as negative.

The fact that every rational number (hence, also, every integer) is also a real number is manifest in the fact that integers and rationals can also be written as *b*-ary numerals as in (4.6). But with rational and integers, we are able to insist that their numerals have special forms. We cite without proof the following classical results from the theory of arithmetic. The result for integers is easily phrased.

Theorem 4.3. *Integers as real numbers A real number is an integer if, and only if, it can be represented by a* finite-length *numeral.*

The result for rationals needs an introductory definition. An *infinite* sequence of numbers Σ is *ultimately periodic* if there exist two *finite* sequences of numbers, Γ and Δ, such that Σ can be written in the following form (spaces added to enhance legibility):

$$\Sigma \; = \; \Gamma \, \Delta \, \Delta \, \Delta \, \ldots \, \Delta \, \ldots$$

The intention here is that the finite sequence Δ is repeated *ad infinitum*.

Theorem 4.4. *Rationals as real numbers A real number is rational if, and only if, it can be represented by a numeral that is* ultimately periodic.

A clarification. Note that two types of sequences of 0s do not affect the value of the number represented by a numeral: (1) an *initial* sequence of 0s to the *left* of the radix point and of all non-0 digits; (2) a *terminal* sequence of 0s to the *right* of the radix point and of all non-0 digits.

One consequence of this fact is that we lose no generality by insisting that every numeral have the following form:

> a finite sequence of digits, followed by a radix point, followed by an infinite sequence of digits

Integers can then be singled out via numerals that have only 0s to the right of the radix point.

---

[5] This is not the traditional way that a mathematician would define the class of real numbers, but it is correct and adequate for thinking about the class.

Theorems 4.3 and 4.4 show us that the three sets of numbers we have defined are a nested progression of successively more inclusive sets, in the sense that *every integer is a rational number* and *every rational number is a real number*. Those interested in the (philosophical) foundations of mathematics might quibble about the verb "is" in the highlighted sentences, but for all practical purposes, we can accept the sentences as written.

C. Not all real numbers are rational. We close this section by verifying the earlier-mentioned assertion about the non-commensurability of the length of the diagonal of a square with the (common) length of its sides—or, equivalently, the leg-length of an isosceles right triangle with the length of its hypotenuse. The following result is a simple application of Theorem 4.2. Its proof only suggests the range of the theorem's myriad applications.

Proposition 4.3 *The real number $\sqrt{2} = 2^{1/2}$ is not rational.*

Proof. We prove the result by contradiction, a proof technique described in Chapter 3.3.4.

Let us assume, for contradiction, that $\sqrt{2}$ is rational. By definition, then $\sqrt{2}$ can be written as a fraction

$$\sqrt{2} = \frac{a}{b}$$

for positive integers a and b. In fact, we can also insist that a and b *share no common prime factor*. For, if a and b shared the prime factor p, then we would have $a = p \times c$ and $b = p \times d$. In this case, though, we would have

$$\sqrt{2} = \frac{a}{b} = \frac{p \times c}{p \times d} = \frac{c}{d}.$$

by cancellation of the common factor p. We can eliminate further common prime factors if necessary until, finally, we find a fraction for $\sqrt{2}$ whose numerator and denominator share no common prime factor. This must occur eventually because each elimination of a common factor leaves us with smaller integers, so the iterative elimination of common factors must terminate.

Let us say that, finally,

$$\sqrt{2} = \frac{k}{\ell} \tag{4.7}$$

where k and ℓ share no common prime factor. Let us square both expressions in (4.7) and multiply both sides of the resulting equation by ℓ^2. We thereby discover that

$$2\ell^2 = k^2. \tag{4.8}$$

This rewriting exposes the fact that k^2 is *even*, i.e., *divisible by* 2. But, Theorem 4.2 tells us that *if k^2 is divisible by 2, then so also is k*! This means that $k = 2m$ for some positive integer m, whoch allows us to rewrite (4.8) in the form

$$2\ell^2 = k^2 = (2m)^2 = 4m^2. \tag{4.9}$$

Hence, we can divide the first and last quantities in (4.9) by 2, to discover that

$$\ell^2 \;=\; 2m^2.$$

Repeating the invocation of Theorem 4.2 now tells us that the integer ℓ must be even.

We now see that *both k and ℓ are even, i.e., divisible by* 2. This contradicts our assumption that k and ℓ share no common prime divisor!

Since every step of our argument is ironclad—except for our assumption that $\sqrt{2}$ is rational, we conclude that that assumption is false! The Proposition is verified! □

The proof of Proposition 4.3 is a classical (and early) example of *proof by contradiction*, as discussed in Section 3.3.4.

4.1.1.4 The complex numbers

A. The basics of the complex numbers. $\mathbb{C}$ denotes the complex numbers

Each complex number $\kappa = a + bi \in \mathbb{C}$ has a *real part*—the part that *does not* involve the imaginary unit i—and an *imaginary part*—the part that *does* involve i. To be explicit: the real part of our number κ, is $\mathrm{Re}(\kappa) = a$; the *imaginary part* of our number κ, is $\mathrm{Im}(\kappa) = b$. The notation $\mathrm{Re}(\kappa)$ and $\mathrm{Im}(\kappa)$ is common but not universal.

Using the basic arithmetic laws that we have discussed thus far, plus the defining equation, $i^2 = -1$, of the imaginary unit i, we find that the *product* of two complex numbers, $a + bi \in \mathbb{C}$ and $c + di \in \mathbb{C}$ is the complex number, call it κ,

$$\kappa \;=\; (a+bi) \cdot (c+di) \;=\; (ac-bd) + (ad+bc)i. \tag{4.10}$$

We note that a "direct" implementation of complex multiplication, i.e., one that implements (4.10) literally, requires four real multiplications—namely, ac, bd, ad, bc.

During the 1960s, people first began to pay close attention to the costs associated with various ways of achieving computational results. They sought—and found—a number of procedures that replaced computations involving k real multiplications (a relatively expensive operation) and ℓ real additions (a relatively inexpensive operation) by computations that achieved the same result but used fewer multiplications and not too many more additions. Complex multiplication was one of the operations they studied. Here is the result.

Proposition 4.4 *One can compute the product of two complex numbers using* three *real multiplications rather than four.*

Proof. Although implementing (4.10) "directly" correctly produces the product $\kappa = (a+bi) \cdot (c+di)$, there is another implementation that is *more efficient*. Specifically, the following recipe computes κ using only *three* real multiplications instead of the four real multiplications of the "direct" implementation. We begin to search for this

recipe by noting that our immediate goal is to compute both $\mathrm{Re}(\kappa) = ac - bd$ and $\mathrm{Im}(\kappa) = ad + bc$. We can accomplish this by computing the *three* real products

$$(a+b)\cdot(c+d); \quad ac; \quad bd \tag{4.11}$$

and then noting that

$$\begin{aligned}\mathrm{Re}(\kappa) &= (a+b)\cdot(c+d) - ac - bd,\\ \mathrm{Im}(\kappa) &= ac - bd\end{aligned} \tag{4.12}$$

We thereby achieve the result of the complex multiplication described in (4.10) while using only *three* real multiplications.

Of course, a full reckoning of the costs of the two implementations we have discussed exposes the fact that the implementation that invokes (4.11) and (4.12) uses *three* real additions rather than the *two* real additions of the "direct" implementation. But this entire exercise was predicated on the observation that each real addition is much less costly than a real multiplication, so trading one multiplication for one addition is an unqualified "win". □

B. Pairing functions: ordering $\mathbb{C}$ and $\mathbb{N} \times \mathbb{N}$. As noted in Section 4.1.1.1.A, "order within a number system is one of one's biggest friends when reasoning about the numbers within the system." This section is devoted to showing that, even when a number system lacks intrinsic desirable ordering properties, we can sometimes endow the system with "inherited access" to such properties by devising a *bijection* that *encodes* the system's numbers as nonnegative integers. We focus here on two important number systems, the rational and complex numbers, $\mathbb{Q}$ and $\mathbb{C}$, whose ordering properties are much weaker than those of the nonnegative integers $\mathbb{N}$: Neither $\mathbb{Q}$ nor $\mathbb{C}$ is *well-ordered*; $\mathbb{C}$ is not even *totally* ordered.

> *Well-ordering* is an especially welcome property because it enables algorithms that are structured as a linear recursion that "counts down" from an argument *n*. Well-ordering guarantees that there is a "bottom" that will terminate the downward recursion.

Two simplifications facilitate our quest for bijective encodings

$$\varepsilon_1 : \mathbb{Q} \leftrightarrow \mathbb{N} \quad \text{and} \quad \varepsilon_2 : \mathbb{C} \leftrightarrow \mathbb{N}$$

- We abstract both $\mathbb{Q}$ and $\mathbb{C}$ as the set $\mathbb{N} \times \mathbb{N}$. Since both $\mathbb{Q}$ and the *complex integers*—complex numbers $a + bi$ where $a, b \in \mathbb{N}$—can be encoded as ordered pairs of nonnegative integers, this is a natural abstraction.
  Of course, using this abstraction to encode $\mathbb{Q}$ into $\mathbb{N}$ ignores common (integer) divisors of a fraction's numerator and denominator, whose elimination preserve the fraction's value. A very simple algorithm would compensate for this.
- We slightly change our agenda and focus on bijections between $\mathbb{N}^+ \times \mathbb{N}^+$ and $\mathbb{N}^+$, where $\mathbb{N}^+$ is the set of *positive* integers.
  Focusing on *positive* integers rather than *nonnegative* integers somewhat simplifies certain mathematical expressions.

The structures of the domain $\mathbb{N} \times \mathbb{N}$ and the range $\mathbb{N}$ of the bijective encodings of interest have led to the name *pairing functions* for these bijections.

We now embark on a short guided tour of the world of pairing functions.

i. The Diagonal pairing function $\mathscr{D}$. Pairing functions first appeared in the literature early in the 19th century. Perhaps the simplest and "prettiest" such function (since it is a *polynomial*) appears, pictorially, in an 1821 work by the great French mathematician Augustin Cauchy [11]. This *diagonal* pairing function was formally specified a half-century later by the German logician Georg Cantor, whose studies [9, 10] revolutionized how we think about infinite sets.

$$\mathscr{D}(x,y) \;=\; \binom{x+y-1}{2} + y \;=\; \frac{(x+y)\cdot(x+y-1)}{2} + y \qquad (4.13)$$

($\mathscr{D}$ of course has a twin that exchanges x and y). $\mathscr{D}$'s mapping of $\mathbb{N}^+ \times \mathbb{N}^+$ onto $\mathbb{N}^+$, as depicted in Fig. 4.1, exposes that we can view $\mathscr{D}$ as:

$$
\begin{array}{rrrrrrrrl}
1 & 3 & 6 & 10 & \boxed{15} & 21 & 28 & 36 & \cdots \\
2 & 5 & 9 & \boxed{14} & 20 & 27 & 35 & 44 & \cdots \\
4 & 8 & \boxed{13} & 19 & 26 & 34 & 43 & 53 & \cdots \\
7 & \boxed{12} & 18 & 25 & 33 & 42 & 52 & 63 & \cdots \\
\boxed{11} & 17 & 24 & 32 & 41 & 51 & 62 & 74 & \cdots \\
16 & 23 & 31 & 40 & 50 & 61 & 73 & 86 & \cdots \\
22 & 30 & 39 & 49 & 60 & 72 & 85 & 99 & \cdots \\
29 & 38 & 48 & 59 & 71 & 84 & 98 & 113 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Fig. 4.1 *The diagonal pairing function $\mathscr{D}$. The shell $x+y=6$ is highlighted.*

1. partitioning $\mathbb{N}^+ \times \mathbb{N}^+$ into "diagonal shells" defined as

$$\{\langle x,y\rangle \mid x+y=2\}, \;\; \{\langle x,y\rangle \mid x+y=3\}, \;\; \{\langle x,y\rangle \mid x+y=4\}, \;\; \ldots$$

This is accomplished by the following subexpression in (4.13).

$$\frac{1}{2}(x+y)\cdot(x+y-1) \;=\; \binom{x+y-1}{2}$$

2. "climbing up" these shells in order.
   This is accomplished by the additional subexpression "$+y$" in (4.13).

Understanding $\mathscr{D}$'s structure leads to an inductive verification of $\mathscr{D}$'s bijectiveness.[6]

---

[6] A computationally more satisfying proof of $\mathscr{D}$'s bijectiveness appears in [23], along with an explicit recipe for computing its inverse.

```
**************
```
DO THE INDUCTION
```
**************
```

The fact that $\mathscr{D}$ is a *polynomial* in x and y raises the natural (to a mathematician!) question of whether there are other polynomial pairing functions. This question remains largely open, but there are a few nontrivial pieces of an answer.

1. There is no *quadratic* polynomial pairing function other than $\mathscr{D}$ (and its twin) [27, 57].
2. No *cubic* or *quartic* polynomial is a pairing function [58].
3. The development in [58] excludes large families of higher-degree polynomials from being pairing functions; e.g., a *super-quadratic* polynomial whose coefficients are all positive cannot be a pairing function.

ii. A methodology for constructing pairing functions. The shell-oriented strategy that underlies the diagonal pairing function $\mathscr{D}$ can be adapted to incorporate shell-"shapes" that are inspired by a variety of computational situations—and can be applied to great computational advantage in such situations. We describe how such adaptation can be effected, and we describe a few explicit shapes and situations. We invite the reader to craft others.

Procedure PF-Constructor($\mathscr{A}$)
/*Construct a shape-inspired pairing function (PF) $\mathscr{A}$*/

Step 1.   Partition the set $\mathbb{N}^+ \times \mathbb{N}^+$ into finite sets called *shells*. Order the shells linearly in some way: many natural shell-partitions carry a natural order.

As noted above, Shell c of the diagonal pairing function $\mathscr{D}$ is the following subset of $\mathbb{N}^+ \times \mathbb{N}^+$: $\{\langle x,y \rangle \mid x+y = c\}$. The parameter c orders $\mathscr{D}$'s shells.

Step 2.   Construct a pairing function from the shells as follows.

Step 2a.   Enumerate $\mathbb{N}^+ \times \mathbb{N}^+$ shell by shell, honoring the ordering of the shells; i.e., list the pairs in shell #1, then shell #2, then shell #3, etc.

Step 2b.   Enumerate each shell in some systematic way, e.g., "by columns": Enumerate the pairs $\langle x,y \rangle$ in each shell in increasing order of y and, for pairs having equal y values, in decreasing order of x.

Proposition 4.5 *Any function* $\mathscr{A} : \mathbb{N}^+ \times \mathbb{N}^+ \leftrightarrow \mathbb{N}^+$ *that is designed via Procedure* PF-Constructor *is a bijection.*

Proof. (Sketch) Step 1 of Procedure PF-Constructor constructs a partial order on $\mathbb{N}^+ \times \mathbb{N}^+$, in which: (*a*) each shell is finite; (*b*) there is a linear order on the shells. Step 2 extends this partial order to a linear order, by honoring the inherent ordering of shells and imposing a linear order within each shell. The function constructed via the Procedure is: *injective* because the disjoint shells are enumerated sequentially; *surjective* because the enumeration within each shell begins immediately after the enumeration within the preceding shell, with no gap.   □

We have noted how to use Procedure PF-Constructor to construct pairing function $\mathscr{D}$. We now use the Procedure to design two other useful pairing functions.

iii. The Square-shell pairing function $\mathscr{S}$. One computational situation where pairing functions can be useful involves storage-mappings for arrays/tables that can expand and/or contract dynamically. In conventional systems, when one expands an $n \times n$ table into an $(n+1) \times (n+1)$ table, one allocates a new region of $(n+1)^2$ storage locations and copies the current table from its n^2-location region to the new region. Of course, this is very wasteful: one is moving $\Omega(n^2)$ items to make room for the anticipated $2n+1$ new items. On any given day, the practical impact of this waste depends on current technology. But, this is a mathematics text, not an engineering one, so we are exploring whether *in principle* we can avoid the waste. The answer is "YES". If we employ a pairing function $\varepsilon : \mathbb{N}^+ \times \mathbb{N}^+ \leftrightarrow \mathbb{N}^+$ to allocate storage for tables, then to expand a table from dimensions $n \times n$ to $(n+1) \times (n+1)$, we need move only $O(n)$ items to accommodate the *new* table entries; the current entries need not be moved. For square tables, the following *Square-shell* pairing function $\mathscr{S}$ manages the described scenario perfectly. After describing $\mathscr{S}$, we comment on managing tables of other shapes.

1	4	9	16	25	36	49	64	$\cdots$
2	3	8	15	24	35	48	63	$\cdots$
5	6	7	14	23	34	47	62	$\cdots$
10	11	12	13	22	33	46	61	$\cdots$
17	18	19	20	21	32	45	60	$\cdots$
26	27	28	29	30	31	44	59	$\cdots$
37	38	39	40	41	42	43	58	$\cdots$
50	51	52	53	54	55	56	57	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Fig. 4.2 *The square-shell pairing function $\mathscr{S}$. The shell $\max(x,y) = 5$ is highlighted.*

$$\begin{aligned} \mathscr{S}(x,y) &= m^2 + m + y - x + 1 \\ &\text{where} \quad m \overset{\text{def}}{=} \max(x-1, y-1). \end{aligned} \tag{4.14}$$

One sees in Fig. 4.2 that $\mathscr{S}$ follows the prescription of Procedure PF-Constructor: (1) it maps integers into the "square shells" defined by: $m = 0$, $m = 1, \ldots$. (2) it enumerates the entries in each shell in a counterclockwise direction. (Of course, $\mathscr{S}$ has a twin that enumerates the shells in a clockwise direction.)

> Using somewhat more complicated instantiations of Procedure PF-Constructor, the study in [83] adapts the square-shell pairing function $\mathscr{S}$ to: (*a*) accommodate, with no wastage, arrays/tables of any fixed aspect ratio $an \times bn$ ($a, b \in \mathbb{N}$); (*b*) accommodate, with only $O(n)$ wastage, arrays/tables whose aspect ratios come from a fixed finite set of candidates—i.e., $(a_1 n \times b_1 n)$ or $(a_2 n \times b_2 n)$ or ... or $(a_k n \times b_k n)$.

iv. The Hyperbolic-shell pairing function $\mathscr{H}$. We have just seen, in paragraphs (ii) and (iii), that when the growth patterns of one's arrays/tables is very constrained, one

can use pairing functions as storage mappings with very little wastage. In contrast, if one employs a pairing function such as $\mathscr{D}$ without consideration of its wastage, then a storage map would show some $O(n)$-entry tables being "spread" over $\Omega(n^2)$ storage locations. In the worst-case, $\mathscr{D}$ spreads the n-position $1 \times n$ array/table over $> \frac{1}{2}n^2$ addresses: $\mathscr{D}(1,1) = 1$ and $\mathscr{D}(1,n) = \frac{1}{2}(n^2 + n)$. This degree of wastefulness can be avoided via careful analysis, coupled with the use of Procedure PF-Constructor. The target commodity to be minimized is the *spread* of a PF-based storage map, which we define as follows.

Note that an ordered pair of integers $\langle x, y \rangle$ appears as a position-index within an n-position table if, and only if, $xy \leq n$. Therefore, we define the spread of a PF-based storage map $\mathscr{M}$ via the function

$$\mathbf{S}_{\mathscr{M}}(n) \overset{\text{def}}{=} \max\{\mathscr{M}(x,y) \mid xy \leq n\}. \tag{4.15}$$

$\mathbf{S}_{\mathscr{M}}(n)$ is the largest "address" that PF $\mathscr{M}$ assigns to any position of a table that has n or fewer positions.

Happily, the tools we have developed enable us to design a pairing function that (to within constant factors) has minimum worst-case spread. This is the *Hyperbolic-shell pairing function* $\mathscr{H}$ of (4.16) and Fig. 4.3.[7]

Let $\delta(k)$ be the number of divisors of the integer k.

$$\mathscr{H}(x,y) = \sum_{k=1}^{xy-1} \delta(k) \ + \text{ the position of } \langle x,y \rangle \text{ among 2-part} \tag{4.16}$$

$$\text{factorizations of the number } xy, \text{ in}$$
$$\text{reverse lexicographic order}$$

1	3	5	8	10	14	16	$\cdots$
2	7	13	19	26	34	40	$\cdots$
4	12	22	33	44	56	69	$\cdots$
6	18	32	48	64	81	99	$\cdots$
9	25	43	63	86	108	130	$\cdots$
11	31	55	80	107	136	165	$\cdots$
15	39	68	98	129	164	200	$\cdots$
17	47	79	116	154	193	235	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Fig. 4.3 *The hyperbolic pairing function* $\mathscr{H}$. *The shell* $xy = 6$ *is highlighted.*

Proposition 4.6 ([83]) **(a)** *The hyperbolic function* $\mathscr{H}$ *is a pairing function.*
 (b) *The spread of* $\mathscr{H}$ *is given by*

---

[7] Details appear in [82, 83].

$$\mathbf{S}_{\mathcal{H}}(n) \;=\; O(n\log n),$$

(c) *No pairing function has better compactness (in the worst case) by more than a constant factor.*

Proof. **(a)** The fact that $\mathcal{H}$ is a pairing function follows from Proposition 4.5.

(b) The pairing function $\mathcal{H}$ maps integers along the "hyperbolic shells" defined by $xy = 1$, $xy = 2$, $xy = 3,\ldots$. Hence, when an integer n is "placed" into the table of values of $\mathcal{H}$, the number of occupied slots is within n of

$$\sum_{i=1}^{n-1} |\{\langle x, y \rangle \mid xy < i\}|$$

Elementary calculations show that this sum is $O(n\log n)$.

(c) The optimality of $\mathcal{H}$ in compactness (up to constant factors) is seen via the following argument. The set of tables that have n or fewer positions are those of aspect ratios $a_i \times b_i$, where $a_i b_i \leq n$. As one sees from Fig. 4.4 (generalized to arbitrary n), the union of the positions of all these arrays is the set of integer lattice points under the hyperbola $xy = n$. It is well-known—cf. [73]—that this set of points has cardinality $\Theta(n\log n)$. Since every table contains position $\langle 1, 1 \rangle$, it follows that, for every n, some table containing n or fewer positions is spread over $\Omega(n\log n)$ "addresses."   □
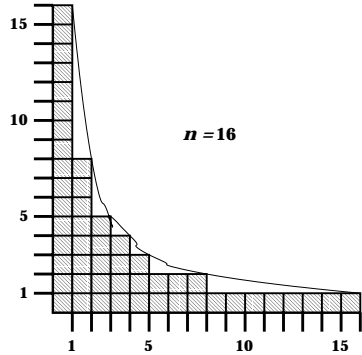
Fig. 4.4 *The aggregate set of positions of tables having* 16 *or fewer position.*

4.1.2 Numerals

We can identify distinct families of *operational* numerals, i.e., numerals that allow one to do things such as perform arithmetic (add, multiply, etc.).

THOUGHTS***** series expansions, strings created by a positional number system, and hybrids built on positional systems such as "scientific notation". **************

Of course, we are all familar with certain numbers that have *non-operational* names that we use all the time. Notable among these are (using terminology that anticipates future sections and chapters):

- π: the ratio of the circumference of a circle to its diameter; $\pi \approx 3.141592653\ldots$
- e: Euler's constant; the base of "natural" logarithms: $e \approx 2.718281828\ldots$
- i: the name of the number whose square is -1; i is appended to the real numbers in order to "complete" them to the complex numbers, within which system every polynomial of degree n has n roots.

A. Positional number systems. The most common way of forming numerals is via strings over a *number base*. We begin with an integer $b > 1$ that will serve as our base, and we define the set $B_b = \{0, 1, \ldots, b - 1\}$ of *digits in base b*. To aid legibility, *within the context of base-b positional numerals*, we denote the digit $b - 1$ as a single character, $\bar{b}$. We then form base-b numerals in the following way. This formation builds on *geometric sums*, a mathematical structure that we shall learn to manipulate, evaluate, and compute with in Section **??**.

A base-b numeral is a string having three sections.

1. The numeral begins with its *integral part*, which is a finite string of digits from B_b: $\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0$.
   The base-b number represented by the numeral's integral part is[8]

   $$\underline{\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0} \;\overset{\text{def}}{=}\; \sum_{i=0}^{n} \alpha_i \cdot b^i$$

2. The numeral continues with a single occurrence of the *radix point* "."
3. The numeral ends with its *fractional part*, which is a string—*finite or infinite*—of digits from B_b: $\beta_0 \beta_1 \beta_2 \cdots$.
   The base-b number represented by the numeral's fractional part is

   $$\underline{.\beta_0 \beta_1 \beta_2 \cdots} \;\overset{\text{def}}{=}\; \sum_{j \geq 0} \beta_j \cdot b^{-j}$$

In summary, then, the base-b number represented by the numeral $\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0 . \beta_0 \beta_1 \beta_2 \cdots$ is

$$\underline{\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0 . \beta_0 \beta_1 \beta_2 \cdots} \;\overset{\text{def}}{=}\; \sum_{i=0}^{n} \alpha_i \cdot b^i + \sum_{j \geq 0} \beta_j \cdot b^{-j}.$$

By prepending a "negative sign" (or, "minus sign") $-$ to a numeral or a number, one renders the thus-embellished entity as negative.

---

[8] Our underlined notation for the numerical value of a numeral is not common, but we find it convenient.

B. Scientific notation. The finite numerals in subsection A are all "exact" in the sense that changing any digit changes the value of the named number. We turn now to a class of numerals that abjure this "exactness" for the sake of expedience. There are a few reasons that one might be willing to do this.

- WHY SCIENTIFIC NOTATION

4.2 Arithmetic and Its Laws

Numbers are *adjectives*—you have five apples and three oranges—but in contrast to adjectives that are purely descriptive—the red ball, the big dog—numbers can be *manipulated*, using the tools of *arithmetic*.

4.2.1 The Tools of Arithmetic

The basic tools of arithmetic reside in a small set of operations, together with two special integers that play important roles with respect to the operations. Since these entities are so tightly intertwined, we discuss them simultaneously.

Two special integers The integers zero (0) and one (1), play special roles within all four of the classes of numbers we have described.

The operations of arithmetic Arithmetic on the four classes of numbers that we have described is built upon a rather small repertoire of operations. When we say that an operation produces a number "of the same sort", we mean that it produces

- an integer result from integer arguments;
- a rational (number) result from rational (number) arguments;
- a real (number) result from real (number) arguments;
- a complex (number) result from complex (number) arguments;

The fundamental operations on numbers are, of course, familiar to the reader. Our goal in discussing them is to stress the laws that govern the operations. Along the way, we also introduce a few operations that are less familiar but no less important.

4.2.1.1 Unary (single-argument) operations

A. Negating and reciprocating numbers. *i. The operation of* negation:

- is a *total function* on the sets $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$. It replaces a number a by its *negative*, a number of the same sort, denoted $-a$.
- is a *partial function* on the nonnegative subsets of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$. It replaces a number a by its negative, $-a$, whenever both a and $-a$ belong to the nonnegative subset being operated on.

Zero (0) is the unique *fixed point* of the operation, meaning that 0 is the unique number a such that $a = -a$.

ii. The operation of reciprocation:

- is a *total function* on the sets $\mathbb{Q}, \mathbb{R}, \mathbb{C}$, which replaces each number a by its *reciprocal*, a number of the same sort, denoted $1/a$ or $\dfrac{1}{a}$. We shall employ whichever notation enhances legibility.
- is *undefined* on every integer a except for 1.

B. Floors, ceilings, magnitudes.
i. The operations of taking floors and ceilings are total operations on the sets $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$.

- The *floor* of a number a, also called *the integer part* of a, denoted $\lfloor a \rfloor$, is the largest integer that does not exceed a; i.e.,:

$$\lfloor a \rfloor \overset{\text{def}}{=} \max_{b \in \mathbb{N}} \left[b \leq a \right]$$

- The *ceiling* of a number a of a, denoted $\lceil a \rceil$, is the smallest integer that is not smaller than a:

$$\lceil a \rceil \overset{\text{def}}{=} \min_{b \in \mathbb{N}} \left[b \geq a \right]$$

Thus, the operations of taking floors and ceilings are two ways to *round* rationals and reals to their "closest" integers.

ii. The operations of taking absolute values/magnitudes: Let a be a real number. The *absolute value*, or, *magnitude*, of a, denoted $|a|$ equals either a or $-a$, whichever is positive. For a complex number a, the definition of $|a|$ is more complicated: it is a measure of a's "distance" from the "origin" complex number $0 + 0 \cdot i$. In detail:

$$|a| = \begin{cases} a & \text{if } [a \in \mathbb{R}] \text{ and } [a \geq 0] \\ -a & \text{if } [a \in \mathbb{R}] \text{ and } [a < 0] \\ \sqrt{b^2 + c^2} & \text{if } [a \in \mathbb{C}] \text{ and } [a = (b + ci)] \end{cases}$$

C. Factorials (of nonnegative integers). The *factorial* of a nonnegative integer $n \in \mathbb{N}$, which is commonly denoted $n!$, is the function defined via the following recursion.

$$\text{FACT}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{FACT}(n-1) & \text{if } n > 0 \end{cases} \tag{4.17}$$

By "unwinding" the recursion in (4.17), one finds that, for all $n \in \mathbb{N}$,

$$n! = \text{FACT}(n) = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1 \tag{4.18}$$

A 3-step inductive argument validates this "unwinding":

1. If $n = 0$, then $\text{FACT}(n) = 1$, by definition (4.17).

2. Assume, for induction, that the expansion in (4.18) is valid for a given $k \in N$:

$$\text{FACT}(k) \ = \ k \cdot (k-1) \cdot (k-2) \cdots 2 \cdot 1$$

3. Then:

$$\begin{aligned}
\text{FACT}(k+1) &= (k+1) \cdot \text{FACT}(k) && \text{by (4.17)} \\
&= (k+1) \cdot k \cdot (k-1) \cdot (k-2) \cdots 2 \cdot 1 && \text{by induction}
\end{aligned}$$

4.2.1.2 Binary (two-argument) operations

A. Addition and Subtraction. The operation of *addition* is a *total function* that replaces any two numbers a and b by a number of the same sort. The resulting number is the *sum of a and b* and is denoted $a + b$.

The operation of *subtraction* is a *total function* on the sets $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$, which replaces any two numbers a and b by a number of the same sort. The resulting number is the *difference of a and b* and is denoted $a - b$. On the nonnegative subsets of the sets $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$—such as $\mathbb{N}$, which is the largest nonnegative subset of $\mathbb{Z}$—subtraction is a *partial function*, which is defined only when $a \geq b$.

Subtraction can also be defined as follows. For any two numbers a and b, *the difference of a and b is the sum of a and the negation of b*; i.e.,

$$a - b \ = \ a + (-b)$$

The special role of 0 *under addition and subtraction.* The number 0 is the *identity* under addition and subtraction. This means that, for all numbers a,

$$a + 0 \ = \ a - 0 \ = \ a.$$

The special role of 1 *under addition and subtraction.* For any integer a, there is no integer between a and $a+1$ or between $a-1$ and a. For this reason, on the sets $\mathbb{Z}$ and $\mathbb{N}$, one often singles out the following special cases of addition and subtraction, especially in reasoning about situations that are indexed by integers. Strangely, these operations have no universally accepted notations.

- The *successor* operation is a *total function* on both $\mathbb{N}$ and $\mathbb{Z}$, which replaces an integer a by the integer $a + 1$.
- The *predecessor* operation is a *total function* on $\mathbb{Z}$, which replaces an integer a by the integer $a - 1$. It is a *partial function* on $\mathbb{N}$, which is defined only when the argument a is positive (so that $a - 1 \in \mathbb{N}$).

The operations of addition and subtraction are said to be *inverse operations* of each other because each can be used to "undo" the other:

$$a \ = \ (a+b) - b \ = \ (a-b) + b$$

B. Multiplication and Division. The operation of *multiplication* is a *total function* that replaces any two numbers a and b by a number of the same sort. The resulting number is the *product of a and b* and is denoted either $a \cdot b$ or $a \times b$. We shall usually favor the former notation, except when the latter enhances legibility.

The operation of *division* is a *partial function* on all of our sets of numbers. Given two numbers a and b, the result of dividing a by b—*when that result is defined*—is the *quotient of a by b* and is denoted by one of the following three notations: a/b, $a \div b$, $\frac{a}{b}$. The *quotient of a by b* is defined precisely when *both*

(1) $b \neq 0$: one can never divide by 0

and

(2) there exists a number c such that $a = b \cdot c$.

Assuming that condition (1) holds, *condition (2) always holds when a and b belong to $\mathbb{Q}$ or $\mathbb{R}$ or $\mathbb{C}$.*

Division can also be defined as follows. For any two numbers a and b, *the quotient of a and b is the product of a and the reciprocal of b* (assuming that the latter exists); i.e.,

$$a/b = a \cdot (1/b).$$

Computing reciprocals of nonzero numbers in $\mathbb{Q}$ and $\mathbb{R}$ is standard high-school level fare; computing reciprocals of nonzero numbers in $\mathbb{C}$ requires a bit of calculational algebra which we do not cover. For completeness, we note that the reciprocal of the *nonzero* complex number $a + bi \in \mathbb{C}$ is the complex number $c + di$ where

$$c = \frac{a}{a^2 + b^2} \quad \text{and} \quad d = \frac{-b}{a^2 + b^2}.$$

The special role of 1 *under multiplication and division.* The number 1 is the *identity* under the operations of multiplication and division. This means that, for all numbers a,

$$a \cdot 1 = a \cdot (1/1) = a.$$

The special role of 0 *under multiplication and division.* The number 0 is the *annihilator* under multiplication. This means that, for all numbers a

$$a \cdot 0 = 0.$$

The operations of multiplication and division are said to be *inverse operations* because, when both operations can be applied, each can be used to "undo" the other:

$$a = (a \cdot b) \div b = (a \div b) \cdot b.$$

C. Binomial coefficients and Pascal's triangle. We close our catalogue of arithmetic operations with a binary operation on[9] $\mathbb{N} \times \mathbb{N}$.

---

[9] In advanced contexts, one encounters binomial coefficients with non-integer arguments.

Let n and $k \leq n$ be nonnegative integers (i.e., elements of $\mathbb{N}$). The binomial coefficient $\binom{n}{k}$ is the number

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k(k-1)(k-2)\cdots 1} \tag{4.19}$$

Many of the secrets of these wonderful numbers—including the fact that they are *integers*—can be deduced from the following results.

Proposition 4.7 *For all $n,k \in \mathbb{N}$ with $k \leq n$:*
(a) *The symmetry rule:*

$$\binom{n}{k} = \binom{n}{n-k} \tag{4.20}$$

(b) *The addition rule:*

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1} \tag{4.21}$$

Proof. (*a*) We verify equation (4.20) by (4.19) plus the commutativity of multiplication (see Section 4.2.2),

$$\begin{aligned}
\binom{n}{k} &= \frac{n!}{k!(n-k)!} \\
&= \frac{n!}{(n-k)!k!} \\
&= \binom{n}{n-k}
\end{aligned}$$

(*b*) We verify equation (4.21) by explicitly adding the fractions exposed by (4.19):

$$\begin{aligned}
\binom{n}{k} + \binom{n}{k+1} &= \frac{n!}{k!(n-k)!} + \frac{n!}{(k+1)!(n-k-1)!} \\
&= n! \cdot \frac{(k+1)+(n-k)}{(k+1)!(n-k)!} \\
&= \frac{(n+1)!}{(k+1)!(n-k)!} \\
&= \binom{n+1}{k+1} \qquad\qquad \square
\end{aligned}$$

Extrapolating from Proposition 4.7, we now present *Pascal's triangle*, named in honor of the French polymath Blaise Pascal. Fig. 4.5 provides a "prefix" of this famed array of integers, for $n,k \leq 5$. The formation rule of the array is:

Formation rule for Pascal's triangle: *The entry at (row $n+1$, column $k+1$) is the sum of the entries at (row n, column k) and (row n, column $k+1$).*

$\binom{n}{k}$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	...
$n=1$	1	1					...
$n=2$	1	2	1				...
$n=3$	1	3	3	1			...
$n=4$	1	4	6	4	1		...
$n=5$	1	5	10	10	5	1	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Fig. 4.5 A "prefix" of Pascal's Triangle, for $n, k \le 5$.

□

If you compare the formation rule for Pascal's triangle with equation (4.21), then you may anticipate the following result.

Proposition 4.8 *The entries of Pascal's triangle are the binomial coefficients. Specifically, for all n,k, the entry at (row n, column k) of the Triangle is* $\binom{n}{k}$.

Proof. We note by observation and direct calculltion (see Fig. 4.5) that the proposition is true for $n = 1$ and $k \in \{0, 1\}$. A simple double induction
****
induction on n, then for each value of n on $k \le n$
SHOULD WE SPELL THIS OUT IN DETAIL? GIVE AS AN EXERCISE?
****
verifies that every binomial coefficient appears in the Triangle and every Triangle entry is a binomial coefficient.   □

Proposition 4.9 *Every binomial coefficient is an integer.*

Proof. Since every entry in Pascal's triangle is obtained from integers via repeated additions, this result follows from Proposition 4.8.   □

Binomial coefficients are indispensable when studying myriad topics related to *counting*, such as:

- what are the relative likelihoods of various 5-card deals from a fair 52-card deck?
- What is the likelihood of observing 15 HEADs and 25 TAILs in 40 flips of a fair coin?
- What are the comparative operation-count costs of Merge-Sort and Quick-Sort when sorting n keys; cf. [20]?

We shall, therefore, see a lot more about binomial coefficients in Section 4.3.1 and Chapter 9. With each subsequent encounter, our respect for these numbers will grow.

4.2.2 The Laws of Arithmetic, and applications

The student should understand the following laws of arithmetic on the reals, rationals, and reals—and be able to employ them cogently in rigorous argumentation.

A. The commutative law.   For all numbers x and y:

$$\textit{for addition:} \qquad x+y \;=\; y+x$$
$$\textit{for multiplication:} \quad x\cdot y \;=\; y\cdot x$$

B. The associative law For all numbers x, y, and z,

$$(x+y)+z \;=\; x+(y+z) \quad \textbf{and} \quad x\cdot(y\cdot z)(x\cdot y)\cdot z \;=\; x\cdot(y\cdot z).$$

This allows one, for instance, to write strings of additions or of multiplications without using parentheses for grouping.

C. The distributive law. For all numbers x, y, and z,

$$x\cdot(y+z) \;=\; (x\cdot y)+(x\cdot z). \tag{4.22}$$

One commonly articulates this law as, "*Multiplication distributes over addition.*"

One of the most common uses of the distributive law reads equation (4.22) "backwards," thereby deriving a formula for *factoring* complex expressions that use both addition and multiplication.

Easily, addition does *not* distribute over multiplication; i.e., in general, $x+y\cdot z \neq (x+y)\cdot(x+z)$. Hence, when we see "$x+y\cdot z$", we know that the multiplication is performed before the addition. In other words, *Multiplication takes priority over addition.* This priority permits us to write the righthand side of (4.22) without parentheses, as in

$$x\cdot(y+z) \;=\; x\cdot y+x\cdot z.$$

Via multiple invocations of the preceding laws, we can derive a recipe for multiplying complicated expressions. We illustrate this via the "simplest" complicated expression, $(a+b)\cdot(c+d)$.

Proposition 4.10 *For all numbers a,b,c,d:*

$$(a+b)\cdot(c+d) \;=\; a\cdot c+a\cdot d+b\cdot c+b\cdot d \tag{4.23}$$

Proof. Note first that because multiplication takes priority over addition, the absence of parentheses in expressions such as (4.10) does not jeopardize unambiguity. Our proof of the proposition invokes the laws we have just enunciated multiple times.

$$
\begin{aligned}
(a+b)\cdot(c+d) &= (a+b)\cdot c \;+\; (a+b)\cdot d && \text{distributive law}\\
&= c\cdot(a+b) \;+\; d\cdot(a+b) && \text{commutativity of multiplication } (2\times)\\
&= c\cdot a+c\cdot b+d\cdot a+d\cdot b && \text{distributive law } (2\times)\\
&= a\cdot c+b\cdot c+a\cdot d+b\cdot d && \text{commutativity of multiplication } (4\times)\\
&= a\cdot c+a\cdot d+b\cdot c+b\cdot d && \text{commutativity of addition}
\end{aligned}
$$

□

We close our short survey of the laws of arithmetic with the following important two-part law.

- *The law of inverses.*

  - Every number x has an *additive inverse*, i.e., a number y such that $x+y = 0$. This inverse is x's *negative* $-x$.
  - Every *nonzero* number $x \neq 0$ has a *multiplicative inverse*, i.e., a number y such that $x\cdot y = 1$. This inverse is x's *reciprocal*, $1/x$.

We close this section with another of our "fun" propositions.

Proposition 4.11 *Let n be any number that has a 2-digit decimal of the form $\delta 5$, where $\delta \in \{0,1,2,3,4,5,6,7,8,9\}$, so that*

$$
n \;=\; 10\cdot\delta + 5
$$

Then

$$
n^2 \;=\; 100\cdot\delta\cdot(\delta+1) + 25.
$$

In other words, one obtains a base-10 numeral for n^2 by multiplying δ by $\delta+1$ and appending 25 to the product.

Examples of Proposition 4.11 include $25^2 = 625$ (because $2\cdot 3 = 6$) and $75^2 = 5625$ (because $7\cdot 8 = 56$).

Proof. (for general δ). We invoke Proposition 4.10 and the distributive law.

$$
\begin{aligned}
n^2 &= (10\cdot\delta + 5)^2 && \text{Given}\\
&= 100\cdot\delta^2 \;+\; 100\cdot delta \;+\; 25 && \text{the proposition}\\
&= 100\cdot(\delta^2 \;+\; \delta) \;+\; 25 && \text{factoring: distributive law}\\
&= 100\cdot\delta\cdot(\delta+1) \;+\; 25 && \text{factoring: distributive law}
\end{aligned}
$$

□

4.2.3 Rational Arithmetic: A Worthwhile Exercise

In Section 4.1.1.2 we defined the rational numbers and reviewed why they were needed to compensate for the general lack of multiplicative inverses in the integers. But we did not review how to perform arithmetic on the elements of the set $\mathbb{Q}$. We

correct this shortcoming now. Of course, the reader will have encountered rational arithmetic long ago—but we are now reviewing the topic in order to provide the reader with a set of worthwhile exercise to reinforce the mathematical thinking whose presentation is our main goal.

The rational numbers build their rules for arithmetic upon the corresponding rules for integers. For all p/q and r/s in $\mathbb{Q}$:

Addition:	$\dfrac{p}{q} + \dfrac{r}{s} = \dfrac{p \cdot s + r \cdot q}{q \cdot s}$
Subtraction:	$\dfrac{p}{q} + \dfrac{r}{s} = \dfrac{p}{q} + \dfrac{(-r)}{s}$
Multiplication:	$\dfrac{p}{q} \cdot \dfrac{r}{s} = \dfrac{p \cdot r}{r \cdot s}$
Division:	$\dfrac{p}{q} \div \dfrac{r}{s} = \dfrac{p}{q} \cdot \dfrac{s}{r}$

It is worth verifying that rational arithmetic as thus defined behaves in the required manner; in particular that rational arithmetic:

- works correctly when the argument rational numbers are, in fact, integers, i.e., when $q = s = 1$ in the preceding table.
- treats the number 0 appropriately, i.e., as an additive identity and a multiplicative annihilator; cf., Sections 4.2.1 and 4.2.2.
- obeys the required laws; cf., Section 4.2.2.
  Verifying the distributivity of rational multiplication over rational addition will be a particularly valuable exercise because of the required amount of manipulation.

4.3 Basic Algebraic Concepts and Their Manipulations

4.3.1 Powers and polynomials

4.3.1.1 Raising a number to a power.

A conceptually powerful notational construct is the operation of *raising a number to a power:* For real numbers a and b, the *bth power* of a, denoted a^b is defined by the system of equations

$$\text{for all numbers } a > 0 \quad a^0 = 1$$

$$\text{for all numbers } a, b, c \quad a^b \cdot a^c = a^{b+c}.$$

(4.24)

This deceptively simple definition has myriad consequences which we often take for granted.

- For all numbers $a > 0$, the number $a^0 = 1$.
  This follows (via cancellation) from (4.24) via the fact that

$$a^b \cdot a^0 \; = \; a^{b+0} \; = \; a^b \; = \; a^b \cdot 1.$$

- For all numbers $a > 0$, the number $a^{1/2}$ is the *square root* of a, i.e., $a^{1/2}$ is the (unique, via cancellation) number b such that $b^2 = a$. Another common notation for The number $a^{1/2}$ is $\sqrt{a}$.
  This follows from (4.24) via the fact that

$$a \; = \; a^1 \; = \; a^{(1/2)+(1/2)} \; = \; a^{1/2} \cdot a^{1/2} \; = \; \left(a^{1/2} \right)^2.$$

- For all numbers $a > 0$ and b, the number a^{-b} is the *multiplicative inverse* of a^b, meaning that $a^b \cdot a^{-b} = 1$
  This follows from (4.24) via the fact that

$$a^b \cdot a^{-b} \; = \; a^{(b+(-b))} \; = \; a^0 \; = \; 1$$

When the power b is a positive integer, then definition (4.24) can be cast in the following attractive inductive form:

$$\begin{aligned} \text{for all numbers } a > 0 \qquad\qquad & a^0 = 1 \\[2mm] \text{for all numbers } a \text{ and integers } b \quad & a^{b+1} = a \cdot a^b. \end{aligned} \tag{4.25}$$

Summing up, we now know about powers that are integral or fractional, positive, zero, or negative

4.3.1.2 Polynomials and their roots.

We want students to master the notions of polynomials and their associated notions, such as degrees and coefficients, and computations therewith, including polynomial summation and multiplication. While polynomial multiplication is often considered "non-elementary", it must be mastered in order to fully understand positional number systems; it is also essential, e.g., when discussing a range of topics relating to, say, fault tolerance and encryption).

Theorem 4.5 (The Fundamental Theorem of Algebra). *Every degree-n univariate polynolmial with complex coefficients has n complex roots*

4.3.1.3 The Binomial Theorem.

Perhaps the simplest bivariate polynomials are the ones in the following family.

$$\text{For } n \in \mathbb{N}^+, \qquad P_n(x,y) \overset{\text{def}}{=} (x+y)^n. \tag{4.26}$$

There are lessons to be learned from the structure of these polynomials, so let us begin to expand them using the arithmetic techniques we have learned earlier.

$$
\begin{aligned}
P_1(x,y) &= (x+y)^1 = x+y \\
P_2(x,y) &= (x+y)^2 = (x+y)\cdot(x+y) \\
&\qquad = x^2 + 2xy + y^2 \\
P_3(x,y) &= (x+y)^3 = (x+y)\cdot(x^2+2xy+y^2) \\
&\qquad = (x^3 + 2x^2y + xy^2) + (x^2y + 2xy^2 + y^3) \\
&\qquad = x^3 + 3x^2y + 3xy^2 + y^3
\end{aligned}
$$

Let us stop to review what we are seeing. We have remarked before that doing mathematics can sometimes involve a wonderfully exciting (quite sophisticated) pattern-matching game. So, let us pattern-match!

1. The coefficients of the expanded $P_1(x,y)$ are $\langle 1, 1 \rangle$.
2. The coefficients of the expanded $P_2(x,y)$ are $\langle 1, 2, 1 \rangle$.
3. The coefficients of the expanded $P_3(x,y)$ are $\langle 1, 3, 3, 1 \rangle$.

There is a pattern emerging here. Can you spot it? Where have we seen a pattern of tuples that begins in the same manner? As a rather broad hint, look at Fig. 4.5! Could the coefficients of each P_n possibly be the successive binomial coefficients

$$\binom{n}{0}, \ \binom{n}{1}, \ \cdots, \ \binom{n}{n-1}, \ \binom{n}{n}$$

Let us use induction to explore this possibility by expanding a generic P_n with symbolic "dummy" coefficients and see what this says about P_{n+1}. To this end, let $a_{n,n-r}$ denote the coefficient of $x^{n-r}y^r$ in the expansion of $P_n(x,y)$. Using our "dummy" coefficients, we have

$$
\begin{aligned}
P_n(x,y) = x^n \ + \ \cdots \ + \ a_{n,n-r}x^{n-r}y^r \ + \ a_{n,n-r-1}x^{n-r-1}y^{r+1} \\
+ \ a_{n,n-r-2}x^{n-r-2}y^{r+2} \ + \ \cdots \ + \ y^n
\end{aligned}
$$

Continuing with this symbolic evaluation, we have:

$$
\begin{aligned}
x\cdot P_n(x,y) = x^{n+1} \ + \ \cdots \ + \ a_{n,n-r}x^{n-r+1}y^r \ + \ a_{n,n-r-1}x^{n-r}y^{r+1} \\
+ \ a_{n,n-r-2}x^{n-r-1}y^{r+2} \ + \ \cdots \ + \ xy^n
\end{aligned}
\tag{4.27}
$$

and

$$y \cdot P_n(x,y) \;=\; x^n y \;+\; \cdots \;+\; a_{n,n-r} x^{n-r} y^{r+1} \;+\; a_{n,n-r-1} x^{n-r-1} y^{r+2}$$
$$+\; a_{n,n-r-2} x^{n-r-2} y^{r+3} \;+\; \cdots \;+\; y^{n+1} \tag{4.28}$$

Because

$$P_{n+1}(x+y) \;=\; (x+y) \cdot P_n(x,y) \;=\; x \cdot P_n(x,y) \;+\; y \cdot P_n(x,y),$$

the "dummy" coefficient $a_{n-r+1,r}$ of $x^{n-r+1} y^r$ in $P_{n+1}(x+y)$ is the sum of the following coefficients in $P_n(x,y)$:

- the coefficient $a_{n,n-r}$ of $x^{n-r} y^r$ and   • the coefficient $a_{n,n-r+1}$ of $x^{n-r+1} y^{r-1}$

By induction, then, for all $n, r \in \mathbb{N}$ with $r \le n$,

$$a_{n,r} + a_{n,r+1} \;=\; a_{n+1,r+1}$$

Combining this equation with the observed initial conditions

$$a_{1,0} \;=\; a_{1,1} \;=\; 1$$

we see that each coefficient $a_{n,r}$ is actually the binomial coefficient $\binom{n}{r}$. This observation is attributed to the renowned English mathematician/physicist Isaac Newton and is enshrined in Newton's famous *Binomial Theorem*.

Theorem 4.6 (The Binomial Theorem). *For all $n \in \mathbb{N}$,*

$$(x+y)^n \;=\; \sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i.$$

4.3.2 Exponentials and Logarithms

This section introduces the fundamentals of two extremely important classes of functions which are functional inverses of each other, in the following sense. Functions f and g are *functional inverses* of each other if for all arguments x

$$f(g(x)) \;=\; x. \tag{4.29}$$

4.3.2.1 Basic definitions

A. Exponential functions. A function f is *exponential* if there is a positive number b such that, for all x,

$$f(x) \;=\; b^x. \tag{4.30}$$

The number b is the *base* of $f(x)$. The basic arithmetic properties of exponential functions are derivable from (4.24), so we leave these details to the reader and turn immediately to the functional inverses of exponential functions..

B. Logarithmic functions. Given an integer $b > 1$ (mnemonic for "base"), the *base-b logarithm* of a real number $a > 0$ is denoted $\log_b a$ and defined by the equation

$$a = b^{\log_b a}. \tag{4.31}$$

Logarithms are partial functions: $\log_b a$ is not defined for non-positive arguments.

The base $b = 2$ is so prominent in the contexts of computation theory and information theory that we commonly invoke one of two special notations for $\log_2 a$: (1) we often elide the base-2 subscript and write $\log a$; (2) we employ the specialized notation $\ln a$. Notationally:

$$\log_2 a \overset{\text{def}}{=} \log a \overset{\text{def}}{=} \ln a$$

We leave to the reader the easy verification, from (4.31), that the *base-b logarithmic function*, defined by

$$f(x) = \log_b x \tag{4.32}$$

is the functional inverse of the base-b exponential function.

4.3.2.2 Fun facts about exponentials and logarithms

Definition (4.31) exposes and—even more importantly—explains myriad facts about logarithms that we often take for granted.

Proposition 4.12 *For any base $b > 1$, for all numbers $x > 0$, $y > 0$,*

$$\log_b(x \cdot y) = \log_b x + \log_b y$$

Proof. Definition (4.31) tells us that $x = b^{\log_b x}$ and $y = b^{\log_b y}$. Therefore,

$$x \cdot y = b^{\log_b x} \cdot b^{\log_b y} = b^{\log_b x + \log_b y},$$

by the laws of powers. Taking base-b logarithms of the first and last terms in the chain yields the claimed equation.   □

Many students believe that the following result is a *convention* rather than a consequence of the basic definitions. *The logarithm of* 1 *to any base is* 0.

Proposition 4.13 *For any base $b > 1$,*

$$\log_b 1 = 0$$

Proof. We note the following chain of equalities.

$$b^{\log_b x} = b^{\log_b(x \cdot 1)} = b^{(\log_b x) + (\log_b 1)} = b^{\log_b x} \cdot b^{\log_b 1}$$

Hence, $b^{\log_b 1} = 1$. If $\log_b 1$ did not equal 0, then $b^{\log_b 1}$ would exceed 1. $\quad\square$

Proposition 4.14 *For all bases $b > 1$ and all numbers x, y,*

$$x^{\log_b y} = y^{\log_b x}$$

Proof. We invoke (4.31) twice to remark that

$$\left[x^{\log_b y} = b^{(\log_b x)\cdot(\log_b y)} \right] \quad \text{and} \quad \left[y^{\log_b x} = b^{(\log_b y)\cdot(\log_b x)} \right]$$

The commutativity of addition completes the verification. $\quad\square$

Proposition 4.15 *For any base $b > 1$,*

$$\log_b(1/x) = -\log_b x$$

Proof. This follows from the fact that $\log_b 1 = 0$, coupled with the product law for logarithms.

$$\log_b x + \log_b(1/x) = \log_b(x \cdot (1/x)) = \log_b 1 = 0$$

$\square$

Proposition 4.16 *For any bases $a, b > 1$,*

$$\log_b x = (\log_b a) \cdot (\log_a x). \tag{4.33}$$

Proof. We begin by noting that, by definition, Note that

$$x = b^{\log_b x} = a^{\log_a x}. \tag{4.34}$$

Let us take the base-b logarithm of the second and third expressions in (4.34) and then invoke the product law for logarithms. From the second expression in (4.34), we find that

$$\log_b \left(b^{\log_b x} \right) = \log_b x. \tag{4.35}$$

From the third expression in (4.34), we find that

$$\log_b \left(a^{\log_a x} \right) = (\log_b a) \cdot (\log_a x). \tag{4.36}$$

We know from (4.34) that the righthand expressions in (4.35) and (4.36) are equal, whence (4.33). $\quad\square$

If we set $x = b$ in (4.33), then we find the following marvelous equation.

Proposition 4.17 *For any integers $a, b > 1$,*

$$(\log_b a) \cdot (\log_a b) = 1 \quad \text{or, equivalently,} \quad \log_b a = \frac{1}{\log_a b}. \tag{4.37}$$

4.3.2.3 Exponentials and logarithms within information theory

The student should recognize and be able to reason about the following facts. If one has an alphabet of a letters/symbols and must provide distinct string-label "names" for n items, then at least one string-name must have length no shorter than $\lceil \log_a n \rceil$.

Proposition 4.18 *Say that one must assign distinct labels to n items, via strings over an alphabet of a letters. Then at least one string-label must have length no shorter than $\lceil \log_a n \rceil$.*

Proof. Let *Sigma* be an alphabet of a letters/symbols. For each integer $k \geq 0$ (i.e., for each $k \in \mathbb{N}$), let $\Sigma^{(k)}$ denote the set of all length-k strings over Σ. The bound of Proposition 4.18 follows by counting the number of strings of various lengths over Σ, because each such string can label at most one item. Let us, therefore, inductively evaluate the cardinality $|\Sigma^{(k)}|$ of each set $\Sigma^{(k)}$.

- $|\Sigma^{(0)}| = 1$
  This is because the null-string ε is the unique string in $\Sigma^{(0)}$, i.e., $\Sigma^{(0)} = \{\varepsilon\}$.
- $|\Sigma^{(k+1)}| = |Sigma| \cdot |\Sigma^{(k)}|$.
  This reckoning follows from the following recipe for creating all strings over Σ of length $k + 1$ from all strings of length k.

$$\Sigma^{(k+1)} \;=\; \{\sigma x \mid \sigma \in \Sigma, x \in \Sigma^{(k)}\}$$

  This recipe is correct because

  - Each string in $\Sigma^{(k+1)}$, as constructed, has length $k + 1$.
    This is because the recipe adds a single symbol to a length-k string.
  - For each string $x \in \Sigma^{(k)}$, there are $|\Sigma|$ distinct strings in $\Sigma^{(k+1)}$, as constructed.
    This is because each string in $\Sigma^{(k+1)}$ begins with a distinct symbol from Σ.
  - $\Sigma^{(k+1)}$, as constructed, contains all strings of length $k + 1$ over Σ.
    This is because for each $\sigma \in \Sigma$ and each $x \in \Sigma^{(k)}$, the string σx is in $\Sigma^{(k+1)}$, as constructed.

We thus have the following recurrence.

$$|\Sigma^{(0)}| = 1$$
$$|\Sigma^{(k+1)}| = |\Sigma| \cdot |\Sigma^{(k)}| \quad \text{for } k \geq 0$$

Using the Master Theorem, we thus find explicitly that
For each $\ell \in \mathbb{N}$,

$$|\Sigma^{(\ell)}| \;=\; \frac{|\Sigma|^{\ell+1} - |\Sigma|}{|\Sigma| - 1} \;\leq\; c \cdot |\Sigma|^{\ell}$$

for some constant c. In order for this quantity to reach $n \in \mathbb{N}$, we must have

$$\ell \;>\; d \cdot \log_{|\Sigma|} n$$

for some small constant d. $\square$

**HERE
Focus on Say, inductively, that there are ℓ_k

Proposition 4.19 *The number of distinct strings of length k over an alphabet of a letters is a^k.*

4.4 Congruences and Modular Arithmetic

4.5 Numbers and Numerals

4.5.1 Number vs. Numeral: Object vs. Name

4.5.2 Geometric series and positional number systems

The relation between simple geometric series and numeration within positional number systems – including changing bases in such systems.

4.6 Useful Nonalgebraic Notions

4.6.1 Nonalgebraic Notions Involving Numbers

If the intended curriculum will approach more sophisticated application areas such as robotics or data science or information retrieval or data mining (of course, at levels consistent with the students' preparation), then one would do well to insist on familiarity with notions such as:

Chapter 5
SUMMATION

5.1 Manifesto

The fundamental goal of this book / chapter / section is to endow the reader with an operational level of conceptual and methodological understanding of the discrete mathematics that is used to study and understand computing. We construe an "operational" level of understanding to be one that enables the reader to "do" mathematics.

Somewhat surprising to the non-mathematician, a large portion of "doing" mathematics, the widely touted "queen of the sciences", is pattern-matching—albeit of a monumentally sophisticated variety. Mathematicians are trained to understand pieces of reality to a depth that allows them to understand how apparently unrelated concepts *A* and *B* can be conceptualized via the same abstract representation, and to analyze (computational, in our bailiwick) advantages to exploiting such representations.

Toward the end of guiding the reader through this forest of abstractions, we categorize our targets in three ways

1. fundamental sums
   of intrinsic intererest
   e.g.: arithmetic sums, geometric sums, mathematically "smooth" sums
2. fundamental techniques
   that work in many distinct situations
   e.g.: using integration for summation; grouping/replication; verification via induction
3. fundamental representations
   that enable one to study the same phenomenon in many seemingly unrelated ways
   e.g.: numbers, Riemann sums, slices of pie, tokens arranged in stylized ways

5.2 Introduction

The operation of *summation*—adding up aggregates of numbers—is of fundamental importance in the world of digital computing. While we humans are able to deal handily with abstractions such as "smoothness" and "continuity", we must employ sophisticated *discretizations* of these concepts in order to enlist the aid of digital computers in dealing with such abstractions. Summations provide a very useful discretization of "continuous" or "smooth" phenomena that are typically dealt with by humans with the aid of the calculus that was invented by Newton and Leibnitz for such dealings.

This chapter is dedicated to exploring how to employ summations as a computational tool. We deal throughout with *series*, i.e., (possibly infinite) sequences of numbers

$$a_1, a_2, \ldots$$

whose sum

$$a_1 + a_2 + \cdots$$

is of interest.

As a simple illustration, consider the following puzzle. You are presented with a traditional 8×8 chess board, plus an offer of a one-time gift of *one million euros*. In place of this one-time gift, you can opt for the money amassed in the following way.
I will go along the chessboard row by row. I will place 1 euro in the first square I encounter, 4 ($= 2^2$) euros in the second, 9 ($= 3^2$) euros in the third , ..., 4096 ($= 64^2$) euros in the last square.
WHICH CHOICE DO YOU MAKE?

By the end of this chapter, you will be able to determine in minutes that the *alternative* offer (the one that uses the chessboard) would net you a sum of 8,390,656 euros—clearly the preferable choice!

5.3 Summing Structured Series

5.3.1 Arithmetic Sums and Series

We define arithmetic sequences and learn how to calculate their sums.

An *n*-term arithmetic sequence:
$$a, \ a+b, \ a+2b, \ a+3b, \ \ldots, a+(n-1)b$$

The corresponding arithmetic series:
$$a + (a+b) + (a+2b) + (a+3b) + \cdots + (a+(n-1)b)$$
$$= \ an + b \cdot (1 + 2 + \cdots + n - 1)$$

(5.1)

We can, thus, sum the arithmetic series in (5.1) by determining the sum of the first m positive integers; $m = n - 1$ in (5.1). We use this result as an opportubnity to introduce important notation.

Proposition For all $n \in \mathbb{N}$,

$$S_1(n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} i \stackrel{\text{def}}{=} 1 + 2 + \cdots + (n-1) + n$$

$$= \frac{1}{2}n(n+1) \tag{5.2}$$

$$= \binom{n+1}{2}.$$

Proof. The *constructive* proof[1] of summation (5.2) that we present now employs a device known to the eminent German mathematician Karl Friedrich Gauss as a pre-teenager.

Write $S_1(n)$ "forwards": $\quad \sum_{i=1}^{n} = 1 + \quad 2 \quad + \cdots + (n-1) + n$

Write $S_1(n)$ "in reverse": $\quad \sum_{i=1}^{n} = n + (n-1) + \cdots + \quad 2 \quad + 1$

$$\tag{5.3}$$

Now add the two representations of $S_1(n)$ in (5.3) *columnwise*. Because each of the n column-sums equals $n+1$, we find that $2S_1(n) = n(n+1)$, which we easily rewrite as in (5.2) (after multiplying both sides of the equation by 2).

It follows that our original series in (5.1) sums as follows.

$$a + (a+b) + (a+2b) + (a+3b) + \cdots + (a+(n-1)b) = an + b \cdot \binom{n}{2}.$$

We can build on Proposition 5.3.1 to craft *two* "constructive" proofs—i.e., proof that explicitly calculate the summation—that each perfect square, say, m^2, is the sum of the first m odd integers, $1, 3, 5, \ldots, 2m - 1$. These proofs complement the "guess-and-verify" inductive proof of the same result in Proposition 5.4.3.

Proposition For all $n \in \mathbb{N}^+$,

$$\sum_{k=1}^{n} (2k - 1) = 1 + 3 + 5 + \cdots + (2n - 1) = n^2. \tag{5.4}$$

That, is, the nth perfect square is the sum of the first n odd integers.

Before presenting our two proofs of this result, we note that the notation in (5.4) is perfectly general: every positive odd integer m can be written in the form $2n - 1$ for some positive integer n.

---

[1] The proof is *constructive* in that it actually derives an answer. This is in contrast to, say, the inductive proof of Proposition 5.4.3, which just verifies a "guessed" answer.

Proof. (*Argument #1.*) By direct calculation, we have

$$\sum_{k=1}^{n} (2k-1) = 2\sum_{k=1}^{n} k \ - n$$
$$= 2\frac{n(n+1)}{2} - n \quad \text{by Proposition 5.3.1}$$
$$= (n^2 + n) - n$$
$$= n^2.$$

Proof. (*Argument #2.*) Let us adapt Gauss's "trick" to this problem. Let us denote the target sum $\sum_{k=1}^{n} (2k-1)$ by $S(n)$.

$$S_n \text{ "forwards": } S(n) = \quad 1 \quad + \quad 3 \quad + \cdots + (2n-3) + (2n-1)$$
$$\tag{5.5}$$
$$S_n \text{ "in reverse": } S(n) = (2n-1) + (2n-3) + \cdots + \quad 3 \quad + \quad 1$$

Now add the two representations of $S(n)$ in (5.5) *columnwise*. Because each of the n column-sums equals $2n$, we find that

$$2S(n) \ = \ 2\sum_{k=1}^{n} (2k-1) \ = \ 2n^2. \tag{5.6}$$

We thus derive the desired summation (5.4) when we divide both sides of equation (5.6) by 2.

5.3.2 *Geometric Sums and Series*

We define geometric sequences and learn how to calculate their sums.

> An *n*-term geometric sequence:
> $a, \ ab, \ ab^2, \ \dots, ab^{n-1}$
>
> $$\tag{5.7}$$
>
> The corresponding geometric series:
> $a + ab + ab^2 + \cdots + ab^{n-1} \ = \ a(1 + b + b^2 + \cdots + b^{n-1})$

It is clear just from this definition that we can sum the series in (5.7) by summing just the sub-series

$$S_b(n) \ \overset{\text{def}}{=} \ 1 + b + b^2 + \cdots + b^{n-1}. \tag{5.8}$$

As we develop a solution technique for sub-series (5.8), we remark that when $b < 1$, an even simpler solution technique will show that the infinite series

$$\sum_{i=0}^{\infty} b^i \ = \ 1 + b + b^2 + \cdots$$

sums to $\dfrac{1}{1-b}$.

Toward the end of developing our first method for summing the sub-series (5.8), we note the following exploitable fact.

> This fact can be viewed as one of the "patterns" we referred to in Section 5.1 as we mentioned how mathematicians operate.

$$S_b(n) = 1 + b + b^2 + \cdots + b^{n-1}$$

$$= 1 + b \cdot S_b(n) - b^n \qquad = b \cdot S_b(n) + (1 - b^n) \qquad (5.9)$$

$$= b^{n-1} + \frac{1}{b} \cdot S_b(n) - \frac{1}{b} \qquad = \frac{1}{b} \cdot S_b(n) + \frac{b^n - 1}{b}$$

We proceed as follows. Write $S_b(n)$ so that its terms are in *decreasing* order. We thereby isolate two cases.

1. Say first that $b > 1$. In this case, we write the series in the form

$$S_b^{b>1}(n) = b^{n-1} + b^{n-2} + \cdots + b^2 + b + 1,$$

and we note that

$$S_b^{b>1}(n) = b^{n-1} + \frac{1}{b} \cdot S_b^{b>1}(n) - \frac{1}{b}.$$

In other words, we have

$$\left(1 - \frac{1}{b}\right) S_b^{b>1}(n) = b^{n-1} - \frac{1}{b},$$

or equivalently,

$$S_b^{b>1}(n) = \frac{b^n - 1}{b - 1}. \qquad (5.10)$$

2. Alternatively, if $b < 1$, then we write the series in the form

$$S_b^{b<1}(n) = 1 + b + b^2 + b^3 + \cdots + b^{n-1}.$$

and we note that

$$S_b^{b<1}(n) = 1 + b \cdot S_b^{b<1}(n) - b^n.$$

In other words,

$$(1 - b)S_b^{b<1}(n) = 1 - b^n$$

or equivalently,

$$S_b^{b<1}(n) = \frac{1 - b^n}{1 - b}. \qquad (5.11)$$

Note that $S_b^{b>1}(n)$ and $S_b^{b<1}(n)$ actually have the same form. We have chosen to write them differently to stress their *approximate* values, which are useful in "back-of-the-envelope" calculations: For very large values of n, we have

$$S_b^{b>1}(n) \approx \frac{b^n}{b-1} \quad \text{while} \quad S_b^{b<1}(n) \approx \frac{1}{1-b}. \tag{5.12}$$

We now exploit our ability to sum geometric sums to illustrate a somewhat surprising, nontrivial fact about integers that are "encoded" in their positional numerals. We hope that this "fun" result will inspire the reader to seek kindred numeral-encoded properties of numbers.

Proposition *An integer n is divisible by an integer m if, and only if, m divides the sum of the digits in the base-$(m+1)$ numeral for n.*

The most familiar instance of this result is phrased in terms of our traditional use of base-10 (decimal) numerals.

An integer n is divisible by 9 if, and only if, the sum of the digits of n's base-10 numeral is divisible by 9.

Proof. (*Argument for general base b*). Of course, we lose no generality by focusing on numerals without leading 0's, for adding leading 0's does not alter a numeral's sum of digits.

To enhance legibility, let $b = m+1$, so that we are looking at the base-b numeral for n. Say that

$$n = \delta_k \cdot b^k + \delta_{k-1} \cdot b_{k-1} + \cdots + \delta_1 \cdot b + \delta_0,$$

so that the sum of the digits of n's base-b numeral is

$$s_b(n) \stackrel{\text{def}}{=} \delta_k + \delta_{k-1} + \cdots + \delta_1 + \delta_0.$$

We next calculate the difference $n - s_b(n)$. We proceed as follows, digit by digit.

$$
\begin{array}{rclclclclc}
n & = & \delta_k \cdot b^k & + & \delta_{k-1} \cdot b^{k-1} & + \cdots + & \delta_1 \cdot b & + \delta_0 \\
s_b(n) & = & \delta_k & + & \delta_{k-1} & + \cdots + & \delta_1 & + \delta_0 \\
\hline
n - s_b(n) = & & \delta_k \cdot (b^k - 1) & + & \delta_{k-1} \cdot (b^{k-1} - 1) & + \cdots + & \delta_1 \cdot (b-1) &
\end{array}
\tag{5.13}
$$

We now revisit summation (5.10). Because b is a positive integer, so that $1 + b + \cdots + b^{a-2} + b^{a-1}$ is also a positive integer, we adduce from the summation that *the integer $b^a - 1$ is divisible by $b - 1$.*

We are almost home. Look at the equation for $n - s_b(n)$ in the system (5.13). As we have just seen, every term on the righthand side of that equation is divisible by $b - 1$. It follows therefore, that the lefthand expression, $n - s_b(n)$, is also divisible by $b - 1$. An easy calculation, which we leave to the reader, now shows that this final fact means that n is divisible by $b - 1$ if, and only if, $s_b(n)$ is.

5.3.3 Deriving and Solving Linear Recurrences

We have discussed already discussed the use of linear recurrences in Section We now derive the mathematics underlying this important topic.

By the time the reader has reached this paragraph, she has the mathematical tools necessary to prove and apply what is called *The Master Theorem for Linear Recurrences* [20]. This level of mathematical preparation should be adequate for most early-undergrad courses on data structures and algorithms, as well for for analyzing a large fraction of the algorithms that she is likely to encounter in daily activities.

Theorem[The Master Theorem for Linear Recurrences] *Let the function F be specified by the following linear recurrence.*

$$F(n) = \begin{cases} aF(n/b) + c & \text{for } n \geq b \\ c & \text{for } n < b \end{cases} \tag{5.14}$$

Then the value of F on any argument n is given by

$$F(n) = (1 + \log_b n)c \qquad \text{if } a = 1$$

$$= \frac{1 - a^{\log_b n}}{1 - a} \approx \frac{1}{1 - a} \quad \text{if } a < 1 \tag{5.15}$$

$$= \frac{a^{\log_b n} - 1}{a - 1} \qquad \text{if } a > 1$$

Proof. In order to discern the recurring pattern in (6.1), let us begin to "expand" the specified computation by replacing occurrences of $F(\bullet)$ as mandated in (6.1).

$$\begin{aligned} F(n) &= & aF(n/b) + c \\ &= & a\left(aF(n/b^2) + c\right) + c & = & a^2 F(n/b^2) + (1 + a)c \\ &= a^2\left(aF(n/b^3) + c\right) + (1 + a)c & = a^3 F(n/b^3) + (1 + a + a^2)c \\ & \vdots & & \vdots \\ &= \left(1 + a + a^2 + \cdots + a^{\log_b n}\right)c \end{aligned} \tag{5.16}$$

The segment of (6.3) "hidden" by the vertical dots betokens an induction that is left to the reader. Equations (5.10) and (5.11) now enable us to demonstrate that (6.2) is the case-structured solution to (6.1).

5.4 On Summing "Smooth" Series

We use the problem of summing the first *n* integers as a running example.

5.4.1 Approximate Sums via Integration

This section illustrates a powerful strategy for obtaining nontrivial upper and lower bounds on summations, by finding continuous *envelopes* that bound the discrete summations both above and below. The areas under the enveloping continuous functions—which we can calculate via integration—provide the desired bounds on the summations.

The continuous functions that enable this method are obtained by the following stratagem, which we illustrate for

- the *harmonic* summation $S_{(harmonic)}(n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} 1/i$ in Fig. 5.1.

○ **Rectangles represent sum**

○ **Integral under curve (a)**
  upper–bounds the area
  of the rectangles

○ **Integral under curve (b)**
  lower–bounds the area
  of the rectangles
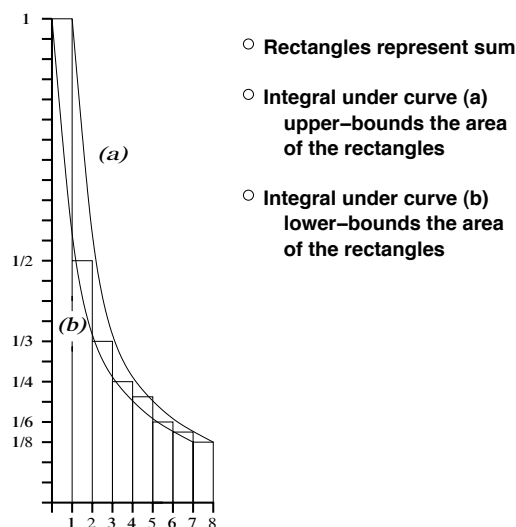
Fig. 5.1 The summation $S_{(harmonic)}(n) = \sum_{i=1}^{n} 1/i$ represented by the aggregate area of a sequence of unit-width rectangles, and bounded above an below by and "enveloping" pair of integrals. The integral that provides the upper bound on the sum yields the area under the righthand continuous curve (*a*), namely, $\int_{1}^{n} \frac{1}{x} dx$. The integral that provides the lower bound yields the area under the lefthand continuous curve (*b*), namely, $\int_{0}^{n-1} \frac{1}{x+1} dx$.

- the summation $S_2(n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} i^2$ in Fig. 5.2.

The strategem operates as follows.

1. Represent the summands seriatim as abutting unit-width rectangles. In Fig. 5.2, for instance, these rectangles have heights 1, 4, 9, 16, and 25. If we were to extend the figure, then next rectangle would have height 36.
2. Construct the continuous curve that will yield (via integration) the upper bound on the summation so that it passes through the upper lefthand corners of the unit-
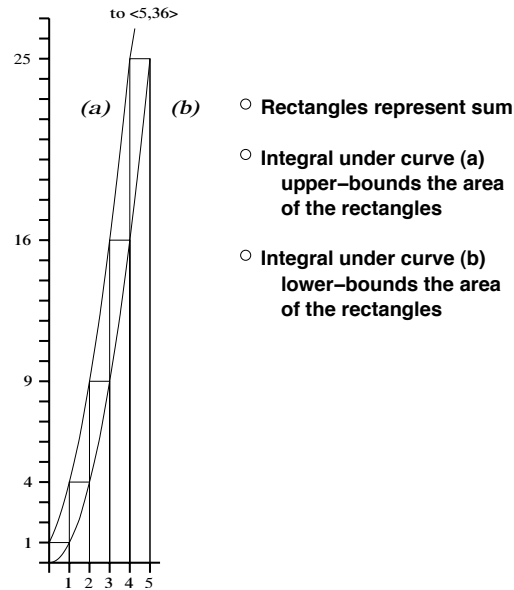
Fig. 5.2 The summation $S_2(n) = \sum_{i=1}^{n} i^2$ represented by the aggregate area of a sequence of unit-width rectangles, and bounded above an below by and "enveloping" pair of integrals. The integral that provides the upper bound on the sum yields the area under the lefthand continuous curve (*a*), namely, $\int_0^n (x+1)^2 dx$. The integral that provides the lower bound yields the area under the righthand continuous curve (*b*), namely, $\int_1^n x^2 dx$.

width rectangles specified by the summation. When constructed appropriately, the area subtended by the abutting rectangles lie completely under this curve.

3. Construct the continuous curve that will yield (via integration) the lower bound on the summation so that it passes through the upper righthand corners of the unit-width rectangles specified by the summation. When constructed appropriately, this curve lies completely within the area subtended by the abutting rectangles.

5.4.1.1 Summing prefixes of the harmonic series.

TO BE DONE

5.4.1.2 Summing successive powers.

We illustrate the technique of bounding summations via integrals by focusing on the summations

$$S_k(n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} i^k,$$

for arbitrary positive integers k. The technique generalizes far beyond the summations $S_k(n)$, but this single illustration should give the reader a firm basis from which to extrapolate to more general summations.

We begin by describing the technique in detail for the summation $S_2(n) = \sum_{i=1}^{n} i^2$. We shall then indicate how to extrapolate from this case to general summations $S_k(n)$.

We obtain our upper bound on $S_2(n)$ by evaluating the integral that yields the area $\overline{A}_2(n)$ under the lefthand continuous curve (a) in Fig. 5.2, namely,

$$\overline{A}_2(n) = \int_0^n (x+1)^2 dx = \int_0^n x^2 dx + 2\int_0^n x dx + \int_0^n dx \qquad (5.17)$$
$$= \frac{1}{3}n^3 + n^2 + n + O(1).$$

We obtain our lower bound on $S_2(n)$ by evaluating the integral that yields the area $\underline{A}_2(n)$ under the righthand continuous curve (b) in the the figure, namely,

$$\underline{A}_2(n) = \int_1^n x^2 dx = \frac{1}{3}n^3 + O(1). \qquad (5.18)$$

Combining these bounds, we finally have the following two-sided bound on $S_2(n)$.

$$\frac{1}{3}n^3 + O(1) \leq \sum_{i=1}^{n} i^2 \leq \frac{1}{3}n^3 + n^2 + n + O(1) \qquad (5.19)$$

For summations $S_k(n)$ with $k > 2$, we follow the roadmap of the case $k = 2$, invoking as a technical tool the following special case of Newton's *Binomial Theorem*.[2]

The Restricted Binomial Theorem. *For all positive integers k,* $(x+1)^k =$ $\sum_{i=0}^{k} \binom{k}{i} x^{k-i}.$[3]

The extrapolation from $S_2(n)$ to general $S_k(n)$ begins with the straightforward exercise of crafting analogues of Fig. 5.2 for arbitrary summations $S_k(n)$. Parallelling the reasoning that led us to the relations (5.17) and (5.18) we obtain:

- an upper bound on summation $S_k(n)$ by evaluating the integral that yields the area $\overline{A}_k(n)$ under the continuous curve that passes through the upper lefthand corners of the unit-width rectangles specified by summation $S_k(n)$.

---

[2] The general form of the Binomial Theorem expands the polynomial $(x+y)^k$ rather than $(x+1)^k$.

[3] As usual, the *binomial coefficients* are defined and denoted as follows.

$$\binom{k}{i} = \frac{k!}{i!(k-i)!} = \frac{k(k-1)(k-2)\cdots(k-i+1)}{i(i-1)(i-2)\cdots 1}$$

$$\overline{A}_k(n) = \int_0^n (x+1)^k dx = \int_0^n \left(\sum_{i=0}^k \binom{k}{i} x^{k-i} \right) dx \tag{5.20}$$

$$= \sum_{i=0}^k \frac{1}{k-i+1} \binom{k}{i} n^{k-i+1} + O(1)$$

This is a proper upper bound because the region defined by this curve totally contains the region subtended by the rectangles.

- a lower bound on summation $S_k(n)$ by evaluating the integral that yields the area $\underline{A}_k(n)$ under the continuous curve that passes through the upper righthand corners of the unit-width rectangles specified by summation $S_k(n)$.

$$\underline{A}_k(n) = \int_1^n x^k dx = \frac{1}{k+1} n^{k+1} + O(1). \tag{5.21}$$

This is a proper lower bound because the region subtended by the rectangles totally contains the below this curve

Using this strategy, one finds that for any positive integer k, summation $S_k(n)$ enjoys the following two-sided bound:

$$\frac{1}{k+1} n^{k+1} + O(1) \leq \sum_{i=1}^n i^k \leq \sum_{i=0}^k \frac{1}{k-i+1} \binom{k}{i} n^{k-i+1} + O(1) \tag{5.22}$$

It follows that $S_k(n)$ enjoys the following asymptotic behavior.

$$S_k(n) = \sum_{i=1}^n i^k \approx \frac{1}{k+1} n^{k+1}. \tag{5.23}$$

We now discuss how to "expand" bounds such as (5.22) into explicit expressions for the summations $S_k(n)$.

5.4.2 *On Using* **Undetermined Coefficients** *to Obtain Explicit Sums*

This section introduces the *Method of Undetermined Coefficients*, a tool that can sometimes refine approximations to summations to exact explicit expressions. We illustrate the tool on the summations $S_k(n)$ that we studied in Section 5.4.1; we use the tool to convert bounds on the $S_k(n)$, such as (5.23), to explicit expressions for the summations.

We illustrate the method by deriving explicit expressions for two rather simple summations: the sum $S_1(n)$ of the first n positive integers and the sum $S_2(n)$ of the first n perfect squares.

Expressing $S_1(n)$ explicitly. Of course, we have already determined $S_1(n)$ in Section 5.3.1, by other means. We provide this alternative derivation as a gentle introduction to the method of undetermined coefficients.

Proposition. *For all $n \in \mathbb{N}$,*

$$S_1(n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} i^2 = 1 + 2 + \cdots + n = \frac{1}{2}n^2 + \frac{1}{2}n$$

Proof. We can reason from the case $k = 1$ of (5.22) that

$$S_1(n) = \frac{1}{2}n^2 + cn$$

for some positive constant c, the eponymous *undetermined coefficient* for this case. We can discover the value of c by evaluating $S_1(n)$ at the value $n = 1$. Any value of n will work; using the *smallest* one simplifies our calculation.

Because $S_1(1) = 1$, we have

$$S_1(1) = 1 = \frac{1}{2} + c,$$

so that $c = 1/2$; in other words,

$$S_1(n) = \frac{1}{2}\left(n^2 + n\right) = \frac{n(n+1)}{2}.$$

We verify this expression by induction in Proposition 5.4.3.

Expressing $S_2(n)$ explicitly. A modestly more complicated undetermined-coefficient calculation allows us to evaluate the sum $S_2(n)$ of the first n squares.

Proposition. *For all $n \in \mathbb{N}$,*

$$S_2(n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} i^2 \stackrel{\text{def}}{=} 1 + 4 + \cdots + (n-1)^2 + n^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \qquad (5.24)$$

$S_2(n)$ is often expressed in a more aesthetic form:

$$S_2(n) = \frac{1}{6}n(n+1)(2n+1) = \frac{2n+1}{3} \cdot \binom{n}{2}.$$

Proof. We can reason from the case $k = 2$ of (5.22) that

$$\sum_{i=0}^{n} k^2 = \frac{1}{3}n^3 + c_2 n^2 + c_1 n. \qquad (5.25)$$

for some positive numbers c_2 and c_1, the eponymous *undetermined coefficient* for this case.

We begin our determination of the constants c_1 and c_2 by instantiating the polynomial in (5.25) with the smallest two values of n, namely, $n = 1, 2$. Of course,

any two values will work, but using the *smallest* ones simplifies calculations. These instantiations leave us with the following pair of linear equations.

$$n = 1 : \sum_{i=0}^{1} k^2 = 1 = 1/3 + c_2 + c_1$$
$$n = 2 : \sum_{i=0}^{2} k^2 = 5 = 8/3 + 4c_2 + 2c_1$$

By elementary arithmetic, these equations simplify to yield the pair

$$c_2 + c_1 = 2/3$$
$$2c_2 + c_1 = 7/6$$

These equations reveal that

$$2/3 - c_2 = 7/6 - 2c_2$$

so that

$$c_2 = 1/2$$

which means that

$$c_1 = 1/6.$$

We have, thus, derived equation (5.24).

With more (calculational) work, but no new (mathematical) ideas, one can derive explicit expressions for the sum of the first n kth powers, i.e., the quantity $S_n^{(k)}$, for any positive integer k.

5.4.3 Validating Approximate Sums via Induction

We illustrate the proof technique of (Finite) Induction by proving the correctness of two familiar summation formulas: (1) the sum of the first n positive integers and (2) the sum of the first n odd positive integers.

For all $n \in \mathbb{N}$,

$$S_n \stackrel{\text{def}}{=} \sum_{i=1}^{n} i \stackrel{\text{def}}{=} 1 + 2 + \cdots + (n-1) + n$$

$$= \frac{1}{2}n(n+1) \qquad (5.26)$$

$$= \binom{n+1}{2}.$$

Proof. For every positive integer m, let $\mathbf{P}(m)$ be the proposition

$$1 + 2 + \cdot + m = \binom{m+1}{2}.$$

Let us proceed according to the standard format of an inductive argument.

1. Because $\binom{2}{2} = 1$, proposition $\mathbf{P}(1)$ is true.

2. Let us assume, for the sake of induction, that proposition $\mathbf{P}(m)$ is true for all positive integers strictly smaller than n. In particular, then, $\mathbf{P}(n-1)$ is true.

3. Consider now the summation

$$1 + 2 + \cdots + (n-1) + n.$$

Because $\mathbf{P}(n-1)$ is true, we know that

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

By direct calculation, we see that

$$\binom{n}{2} + n = \frac{n(n-1)}{2} + n$$
$$= \frac{n^2 - n + 2n}{2}$$
$$= \frac{n^2 + n}{2}$$
$$= \binom{n+1}{2}$$

Because n is an arbitrary positive integer, we conclude that $\mathbf{P}(n)$ is true whenever

- $\mathbf{P}(1)$ is true
- *and* $\mathbf{P}(m)$ is true for all $m < n$.

By the Principle of (Finite) Induction, then, we conclude that $\mathbf{P}(n)$ is true for all $n \in \mathbb{N}^+$.

We turn now to our second summation, which asserts that each perfect square of a positive integer, say, n^2, is the sum of the first n odd integers, $1, 3, 5, \ldots, 2n-1$. This proof complements the constructive proofs of the same result in Proposition 5.3.1.

For all $n \in \mathbb{N}^+$,

$$\sum_{k=1}^{n} (2k-1) = 1 + 3 + 5 + \cdots + (2n-1) = n^2.$$

That, is, the nth perfect square is the sum of the first n odd integers.

Verification. For every positive integer m, let $\mathbf{P}(m)$ be the proposition

$$m^2 = 1 + 3 + 5 + \cdots + 2m - 1.$$

Let us proceed according to the standard format of an inductive argument.

1. Because $1 \cdot 1 = 1$, proposition $\mathbf{P}(1)$ is true.

2. Let us assume, for the sake of induction, that proposition $\mathbf{P}(m)$ is true for all positive integers strictly smaller than n. In particular, then, $\mathbf{P}(n-1)$ is true.

3. Consider now the summation

$$1 + 3 + 5 + \cdots + 2n - 3 + 2n - 1$$

Because $\mathbf{P}(n-1)$ is true, we know that

$$1 + 3 + 5 + \cdots + 2n - 3 + 2n - 1 \;=\; (n-1)^2 + 2n - 1.$$

By direct calculation, we see that

$$(n-1)^2 + 2n - 1 \;=\; (n^2 - 2n + 1) + (2n - 1) \;=\; n^2.$$

Because n is an arbitrary positive integer, we conclude that $\mathbf{P}(n)$ is true whenever

- $\mathbf{P}(1)$ is true
- *and* $\mathbf{P}(m)$ is true for all $m < n$.

By the Principle of (Finite) Induction, then, we conclude that $\mathbf{P}(n)$ is true for all $n \in \mathbb{N}^+$. $\square$

Chapter 6
Deriving and Solving Linear Recurrences

While we have discussed already discussed the use of linear recurrences in Section 2.2.1, we derive the underlying mathematics in this section

By the time the reader has reached this paragraph, she has the mathematical tools necessary to prove and apply what is called *The Master Theorem for Linear Recurrences* [20]. This level of mathematical preparation should be adequate for most early-undergrad courses on data structures and algorithms, as well for for analyzing a large fraction of the algorithms that she is likely to encounter in daily activities.

Theorem 6.1 (The Master Theorem for Linear Recurrences). *Let the function F be specified by the following linear recurrence.*

$$F(n) = \begin{cases} aF(n/b) + c \ \textit{for } n \geq b \\ \quad c \qquad \textit{for } n < b \end{cases} \tag{6.1}$$

Then the value of F on any argument n is given by

$$\begin{aligned} F(n) &= (1 + \log_b n)c & \textit{if } a = 1 \\[2mm] &= \frac{1 - a^{\log_b n}}{1 - a} \approx \frac{1}{1 - a} & \textit{if } a < 1 \\[2mm] &= \frac{a^{\log_b n} - 1}{a - 1} & \textit{if } a > 1 \end{aligned} \tag{6.2}$$

Proof. In order to discern the recurring pattern in (6.1), let us begin to "expand" the specified computation by replacing occurrences of $F(\bullet)$ as mandated in (6.1).

$$\begin{aligned} F(n) &= & aF(n/b) + c \\ &= & a\left(aF(n/b^2) + c\right) + c &= & a^2 F(n/b^2) + (1+a)c \\ &= a^2\left(aF(n/b^3) + c\right) + (1+a)c &= & a^3 F(n/b^3) + (1+a+a^2)c \\ & & \vdots & & \vdots \\ &= & \left(1 + a + a^2 + \cdots + a^{\log_b n}\right)c \end{aligned} \tag{6.3}$$

The segment of (6.3) "hidden" by the vertical dots betokens an induction that is left to the reader. Equations (5.10) and (5.11) now enable us to demonstrate that (6.2) is the case-structured solution to (6.1).   □

Chapter 7
COUNTING

7.1 Elementary counting

7.1.1 Binary Strings and Power Sets

Proposition 7.1 *For every integer $b > 1$, there are b^n b-ary strings of length n.*

Proof. The asserted numeration follows most simply by noting that there are always b times as many b-ary strings of length n as there are of length $n-1$. This is because we can form the set of b-ary strings of length n as follows. Take the set A_{n-1} of b-ary strings of length $n-1$, and make b copies of it, call them $A_{n-1}^{(0)}, A_{n-1}^{(1)}, \ldots, A_{n-1}^{(b-1)}$. Now, append 0 to every string in $A_{n-1}^{(0)}$, append 1 to every string in $A_{n-1}^{(1)}$, ..., append $\bar{b} = b-1$ to every string in $A_{n-1}^{(b-1)}$. The thus-amended sets $A_{n-1}^{(i)}$ are mutually disjoint (because of the terminal letters of their respective strings), and they collectively contain all b-ary strings of length n. $\square$

Proposition 7.2 *The power set $\mathscr{P}(S)$ of a finite set S contain $2^{|S|}$ elements.*

Proof. Let us begin by taking an arbitrary finite set S—say of n elements—and laying its elements out in a line. We thereby establish a correspondence between S's elements and positive integers: there is the first element, which we associate with the integer 1, the second element, which we associate with the integer 2, and so on, until the last element along the line gets associated with the integer n.

Next, let's note that we can specify any subset S' of S by specifying a length-n *binary (i.e., base-2) string*, i.e., a string of 0's and 1's. The translation is as follows. If an element s of S appears in the subset S', then we look at the integer we have associated with s (via our linearization of S), and we set the corresponding bit-position of our binary string to 1; otherwise, we set this bit-position to 0. In this way, we get a distinct subset of S for each distinct binary string, and a distinct binary string for each distinct subset of S.

Let us pause to illustrate our correspondence between sets and strings by focussing on the set $S = \{a,b,c\}$. Just to make life more interesting, let us lay S's elements out in the order b,a,c, so that b has associated integer 1, a has associated integer 2, and c has associated integer 3. We depict the elements of $\mathscr{P}(S)$ and the corresponding binary strings in the following table.

Binary string	Set of integers	Subset of S
000	$\emptyset$	$\emptyset$
001	$\{3\}$	$\{c\}$
010	$\{2\}$	$\{a\}$
011	$\{2,3\}$	$\{a,c\}$
100	$\{1\}$	$\{b\}$
101	$\{1,3\}$	$\{b,c\}$
110	$\{1,2\}$	$\{a,b\}$
111	$\{1,2,3\}$	$\{a,b,c\} = S$

Back to the Proposition: We have verified the following: *The number of length-n binary strings is the same as the number of elements in the power set of S!* The desired numeration thus follows by the ($b = 2$) instance of Proposition 7.1.   □

The binary string that we have constructed to represent each set of integers $N \subseteq \{0, 1, \ldots, n-1\}$ is called the *(length-n) characteristic vector of the set N*. Of course, the finite set N has characteristic vectors of all finite lengths. Generalizing this idea, *every* set of integers $N \subseteq \mathbb{N}$, whether finite or infinite, has an *infinite* characteristic vector, which is formed in precisely the same way as are finite characteristic vectors, but now using the set $\mathbb{N}$ as the base set.

within discrete frameworks, including introducing discrete probability/likelihood as a ratio:

$$\frac{\text{number of targeted events}}{\text{number of possible events}}$$

Chapter 8
GRAPH-THEORETIC ENTITIES

8.1 Basic Graph-Theoretic Topics

Graphs provide one of the richest technical and conceptual frameworks in the world of computing. They provide concrete representations of manifold data structures, hence must be well understood in preparation for a "Data Structures and Algorithms" course. They embody tangible abstractions of relationships of all sorts, hence must be well understood in order to discuss entities as varied as web-search engines and social networks with precision and rigor. As with most of the topics we are discussing, graph-oriented concepts must be taught "in layers". All students must be conversant with the use of graphs to represent and reason about a variety of complicated relationships, but the degree of sophistication that an individual student requires depends both on the abilities of the student and the range of applications that will appear in the student's program. The fundamental concepts in this essay should be understood by all students—although each can be developed with more texture and nuance within the context of specific application domains.

Many developments in computing technology over recent decades have made it imperative that graphs no longer be viewed by students as the static objects introduced, e.g., as abstractions of data struntures. Applications ranging from databases to web-search engines to social networks demand an appreciation of graphs as dynamic objects. This change in perspective affects many aspects of the mathematical prerequisites for any CS/CE program.

8.1.1 Fundamental Notions

The basic components of graphs are *nodes/vertices* (one encounters both terms in the literature) and the *edges* that interconnect them. When a graph is *undirected*, an edge connotes some sort of sibling-like relationship among nodes of "equal" status. When a graph is *directed* (sometimes referred to as a *digraph*), an edge connotes

an "unequal" relationship such as parenthood or priority or dependence; edges in directed graphs are often termed *arcs*. In many situations involving directed graphs, it is important to recognize The *dual* of a digraph G is a digraph obtained by reversing all of G's arcs. One sometimes encounters situations when arguments about, or operations on, a digraph G can be "translated" to arguments about, or operations on, G's dual with only clerical effort. A *subgraph* G' of a graph G is a graph whose nodes are a subset of G's and whose edges are a subset of G's that interconnect only nodes of G'. A *path* in an undirected graph is a sequence of nodes within which every adjacent pair is connected by an edge. A path is a *cycle* if all nodes in the sequence are distinct, except for the first and last, which are identical. Paths and cycles in directed graphs are defined similarly, except that every adjacent pair of nodes must be connected by an arc, and all arcs must "point in the same direction."

Getting formal. A *directed graph* (*digraph*, for short) $\mathscr{G}$ is given by a set of *nodes* $\mathscr{N}_\mathscr{G}$ and a set of *arcs* (or *directed edges*) $\mathscr{A}_\mathscr{G}$. Each arc has the form $(u \to v)$, where $u, v \in \mathscr{N}_\mathscr{G}$; we say that this arc goes *from u to v*. A *path* in $\mathscr{G}$ is a sequence of arcs that share adjacent endpoints, as in the following path from node u_1 to node u_n:

$$(u_1 \to u_2),\ (u_2 \to u_3),\ \ldots,\ (u_{n-2} \to u_{n-1}),\ (u_{n-1} \to u_n). \qquad (8.1)$$

It is sometimes useful to endow the arcs of a digraph with labels from an alphabet Σ. When so endowed, the path (8.1) would be written

$$(u_1 \overset{\lambda_1}{\to} u_2),\ (u_2 \overset{\lambda_2}{\to} u_3),\ \ldots,\ (u_{n-2} \overset{\lambda_{n-2}}{\to} u_{n-1}),\ (u_{n-1} \overset{\lambda_{n-1}}{\to} u_n),$$

where the λ_i denote symbols from Σ. If $u_1 = u_n$, then we call the preceding path a *cycle*.

An *undirected graph* is obtained from a directed graph by removing the directionality of the arcs; the thus-beheaded arcs are called *edges*. Whereas we say:

the *arc* (u, v) goes *from* node u *to* node v

we say:

the undirected edge $\{u, v\}$ goes *between* nodes u and v

or, more simply:

the undirected edge $\{u, v\}$ *connects* nodes u and v.

Undirected graphs are usually the default concept, in the following sense: *When $\mathscr{G}$ is described as a "graph," with no qualifier "directed" or "undirected," it is understood that $\mathscr{G}$ is an undirected graph.*

8.1.2 Graph evolution and decomposition

Any course on algorithms will discuss graphs, especially trees, that evolve over time. Such structures arise in the study of "classical" algorithmic topics such minimum-spanning-tree and branch-and-bound algorithms, as well as in the study of more modern topics such as social networks and internetworks. For the most part, the

mathematics already appearing in this essay suffices for these studies: the students must assimilate new algorithmic notions, not new mathematics. That said, students will be challenged to utilize the mathematical devices that they have (hopefully) mastered in new, more sophisticated, ways. Instructors who wish to lead their students to even a casual understanding of emerging modalities and platforms for computing are thereby challenged to teach required mathematical preliminaries using exemplars that include these new modalities and exemplars.

A fundamental variety of relevant notions within the study of graphs reside in the notions known in various contexts via terms such as *graph separators* or *graph bisection*. The key idea that underlies these notions is that certain graph-theoretic structures interconnect their graphs' constituent nodes more or less densely — and the type and level of interconnectivity has important algorithmic consequences. In such situations, the student must understand how the phenomenon/a modeled by the graphs of interest are elucidated by the way a graph can be broken into subgraphs by excising nodes or edges. When discussing communication-related structures, for instance, graph are often used to model the individual pairwise communications that must occur in order to accomplish the desired overall communication (such as a broadcast). There is often a provable tradeoff between the number of such pairwise communications and the overall time for the completion of the overall task. As another example, when discussing social networks, one can pose questions such as: which node in a network is best to connect to (when joining the network) in order to best facilitate one's interactions or influence within the community. The latter topic leads, e.g., to the study of *power-law* networks, a topic that would not be studied in depth in any early course; indeed, the structure of these networks is not yet well understood even in advanced settings.

8.1.3 Trees

Within the world of graphs, trees occupy a place of honor. In their undirected form, the are defined by the proerty of containing no cycles. They then provide the minimal interconnection structure that provides a path connecting every pair of nodes. Of course, trees appear also in directed form. Among directed trees, a very important subclass are those that have a *root*, i.e., a node that has a directed path to every other node of the tree. In applications, rooted trees represent hierarchical organizations (as do families when viewed generationally).

More formally: *rooted trees* are a class of *acyclic* digraphs. Paths in trees that start at the root are often called *branches*. The *acyclicity* of a tree $\mathscr{T}$ means that for any branch of $\mathscr{T}$ of the form (8.1), we cannot have $u_1 = u_n$, for this would create a cycle. Each rooted tree $\mathscr{T}$ has a designated *root node* $u_n \in \mathscr{N}_{\mathscr{T}}$ that resides at the end of a branch (8.1) that starts at $r_{\mathscr{T}}$ (so $u_1 = r_{\mathscr{T}}$) is said to reside at *depth $n-1$* in $\mathscr{T}$; by convention, $r_{\mathscr{T}}$ is said to reside at depth 0. $\mathscr{T}$'s root $r_{\mathscr{T}}$ has some number (possibly 0) of arcs that go from $r_{\mathscr{T}}$ to its *children,* each of which thus resides at depth 1 in $\mathscr{T}$; in turn, each child has some number of arcs (possibly 0) to its children, and

so on. (Think of a family tree.) For each arc $(u \rightarrow v) \in A_{\mathscr{T}}$, we call u a *parent* of v, and v a *child* of u, in $\mathscr{T}$; clearly, the depth of each child is one greater than the depth of its parent. Every node of $\mathscr{T}$ except for $r_{\mathscr{T}}$ has precisely one parent; $r_{\mathscr{T}}$ has no parents. A childless node of a tree is a *leaf*. The transitive extensions of the parent and child relations are, respectively, the *ancestor* and *descendant* relations. The *degree* of a node v in a tree is the number of children that the node has, call it c_v. If every nonleaf node in a tree has the same degree c, then we call c the *degree of the tree*.

8.2 Hypergraphs: an advanced topic

When studying certain computing-related topics, one will need a generalization of graphs called *hypergraphs*. A hypergraph has nodes that are analogous to the nodes of a graph, but in place of edges a hypergraph has *hyperedges*. Each hyperedge is a set of nodes whose size is not restricted to 2; thus, hypergraphs represent internode relationships that are not "binary". A simple example can be framed by describing *bus-connected* systems, such as occur in certain genres of digital circuits and certain message-passing systems. The underlying idea is that nodes that coreside in a hyperedge represent agents that "hear" all messages simultaneously, or equipotential points in a network. Because of their inherent complexity, hypergraphs as graph-theoretic objects are usually relegated to advanced courses, but specific concrete instantiations, exemplified by bus-oriented communication and digital circuits should be accessible even to early students.

Chapter 9
PROBABILITY AND STATISTICS

Elements of probability theory and statistics infuse every area of computing. The practicality of many algorithms that are experientially the most efficient for their target tasks depend on the *distribution* of inputs in "real" situations. Design methodologies for crucial complex circuits must acknowledge the *mean times to failure* of the critical components of the circuits. Sophisticated searching algorithms must take into account the relative *likelihoods* of finding one's goal in the various optional search directions. Analyzing and understanding large corpora of data requires a variety of methodologies that build on the concepts of *clustering* and/or *decomposition*.

A student needs at least an introduction to the foundations of probability and statistics to even understand, all the moreso to master, the terms highlighted in the preceding paragraph. We outline many of the key concepts that a student must be exposed to in the following subsections.

9.1 The Elements of Combinatorial Probability

Perhaps the easiest and most engaging way to introduce "probability via counting" is by calculating the comparative likelihoods of various deals in 5-card poker and of various rolls of a pair of dice. The arithmetic required for this discussion is elementary and the "application" to gambling of interest even to non-gamblers: "Why is such a deal in poker (say, a straight) worth more than another (say, three of a kind)?" One can also introduce in this setting concepts such as randomness, bias, etc., that are so important in the design of experiments and the analysis of their outcomes.

9.2 Toward a Basic Understanding of Statistics

Most students whose interest tend to the empirical will likely "do" statistics with
the aid of apps, rather than by explicitly writing programs that perform the required
calculations. That said, all students should understand the crucial notion of *random
variable* and should be conversant with the most common statistical distributions. "Conversant" in this context should include complete understandings of the
(low-numbered) moments of *at least* the *uniform* and *exponential* distributions. They
should know how to compute, say, the means and variances of various distributions
and, most importantly, they should *understand* the sense in which the variance of a
distribution give *important* information that is not available from the mean. All of
this is prerequisite to rigor in experimentation.

9.2.0.1 The Elements of Empirical Reasoning

Empirical reasoning does not convey the certitude that formal reasoning does. Students should understand how to craft experiments in a way that collects the "right"
data. The should then be able—perhaps just with statistical packages—to interpret
the results they collect and to understand what conclusions are justifiable. *It is essential that all students understand the ditinction between* positive correlation *and*
causation*!* (Most of the public would seem to flunk that test.)

In order to satisfy the preceding demands, students should understand enough
about statistics—including definitions and meanings related to distributions and
their moments—to understand what conclusions can be made based on experimental results, and to understand how to describe conclusions in a way that is supported
by the statistics.

9.3 Beyond the Basics

As students are introduced to modern topics within computing, whether at the level
of a Computing Literacy course or a post-core technical course, they will have to
master a variety of more specialized topics that combine pieces of the elements we
have discussed in this essay. While these topics are beyond the level of generality
aimed at in this essay, some may be appropriate prerequisites to programs that have
some specialized foci.

- Issues relating to *clustering* find application in applicatios as diverse as: *linear-algebraic computations*, *data mining*, *design and layout of digital circuitry*.
- Issues building on *graph separation/decomposition* are encountered when studying: *linear-algebraic computing*, *fault-tolerant design*, *load balancing*.
- Many issues relating to *fault/failure tolerance* and *data analytics* benefit from
  study using *random walks* (at least in one dimension).

- Many useful ideas regarding the *encoding and manipulation of data* can be gleaned from the elements of *information theory* and *computer arithmetic*.

The preceding list is really endless. Hopefully readers will be inspired by our few examples to compile a longer version that is appropriate for their particular environments.

References

1. F.E. Allen and J. Cocke (1976): A program data flow analysis procedure. *Comm. ACM 19* 137–147.
2. D.N. Arden (1960): *Theory of Computing Machine Design*. Univ. Michigan Press, Ann Arbor, pp. 1–35.
3. E.T. Bell (1986): *Men of Mathematics*. Simon and Schuster, New York.
4. F. Bernstein (1905): Untersuchungen aus der Mengenlehre. *Math. Ann. 61*, 117–155.
5. G. Birkhoff and S. Mac Lane (1953): *A Survey of Modern Algebra*, Macmillan, New York.
6. E. Bishop (1967): *Foundations of Constructive Analysis*, McGraw Hill, New York.
7. W.W. Boone, F.B. Cannonito, R.C. Lyndon (1973): *Word Problems: Decision Problem in Group Theory*, North-Holland, Amsterdam.
8. R. Buyya, D. Abramson, J. Giddy (2001): A case for economy Grid architecture for service oriented Grid computing. *10th Heterogeneous Computing Wkshp.*
9. G. Cantor (1874): Über eine Eigenschaft des Inbegriffes aller reellen algebraischen Zahlen. *J. Reine und Angew. Math. 77*, 258–262.
10. G. Cantor (1878): Ein Beitrag zur Begründung der transfiniter Mengenlehre. *J. Reine Angew. Math. 84*, 242–258.
11. A.L. Cauchy (1821): *Cours d'analyse de l'École Royale Polytechnique, 1ère partie: Analyse algébrique*. l'Imprimerie Royale, Paris. Reprinted: Wissenschaftliche Buchgesellschaft, Darmstadt, 1968.
12. N. Chomsky (1956): Three models for the description of language. *IRE Trans. Information Theory 2*, 113–124.
13. N. Chomsky (1959): On certain formal properties of grammars. *Inform. Contr. 2*, 137–167.
14. A. Church (1941): *The Calculi of Lambda-Conversion. Annals of Math. Studies 6*, Princeton Univ. Press, Princeton, NJ.
15. A. Church (1944): *Introduction to Mathematical Logic, Part I. Annals of Math. Studies 13*, Princeton Univ. Press, Princeton, NJ.
16. W. Cirne and K. Marzullo (1999): The Computational Co-Op: gathering clusters into a metacomputer. *13th Intl. Parallel Processing Symp.*, 160–166.
17. E.F. Codd (1970): *A relational model of data for large shared data banks. Comm. ACM 13*, 377–387.
18. S.A. Cook (1971): The complexity of theorem-proving procedures. *ACM Symp. on Theory of Computing*, 151–158.
19. I.M. Copi, C.C. Elgot, J.B. Wright (1958): Realization of events by logical nets. *J. ACM 5*, 181–196.
20. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein (2001): *Introduction to Algorithms (2nd ed.)*. MIT Press, Cambridge, MA.
21. H.B. Curry (1934): Some properties of equality and implication in combinatory logic. *Annals of Mathematics, 35*, 849–850.

22. H.B. Curry, R. Feys, W. Craig (1958): *Combinatory Logic. Studies in logic and the foundations of mathematics*. North-Holland, Amsterdam.

23. M. Davis (1958): *Computability and Unsolvability*. McGraw-Hill, New York.

24. J.R. Douceur (2002): The Sybil attack. *1st Intl. Wkshp. on Peer-to-Peer Systems*.

25. R.W. Floyd (1962): Algorithm 97: Shortest Path. *Comm. ACM 5*, 345.

26. R.W. Floyd (1967): Assigning meanings to programs. *Proc. Symposia in Applied Mathematics 19*, 19–32.

27. R. Fueter and G. Pólya (1923): Rationale Abzählung der Gitterpunkte. *Vierteljschr. Naturforsch. Ges. Zürich 58*, 380–386.

28. M.R. Garey and D.S. Johnson (1979): *Computers and Intractability*. W.H. Freeman and Co., San Francisco.

29. A. Gilat (2004): *MATLAB: An Introduction with Applications (2nd ed.)*. J. Wiley & Sons, New York.

30. K. Gödel (1931): Über Formal Unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme, I. *Monatshefte für Mathematik u. Physik 38*, 173–198.

31. B. Goldberg (1996): Functional programming languages, *ACM Computing Surveys 28*, 249–251.

32. O. Goldreich (2006): On teaching the basics of complexity theory. In *Theoretical Computer Science: Essays in Memory of Shimon Even. Springer Festschrift series, Lecture Notes in Computer Science 3895*, Springer, Heidelberg.

33. G.H. Golub, C.F. Van Loan (1996): *Matrix Computations* (3rd ed.) Johns Hopkins Press, Baltimore.

34. P.R. Halmos (1960): *Naive Set Theory*. D. Van Nostrand, New York.

35. D. Harel (1987): *Algorithmics: The Spirit of Computing*. Addison-Wesley, Reading, MA.

36. J. Hartmanis and R.E. Stearns (1966): *Algebraic Structure Theory of Sequential Machines*. Prentice Hall, Englewood Cliffs, NJ.

37. L.S. Heath, F.T. Leighton A.L. Rosenberg (1992): "Comparing queues and stacks as mechanisms for laying out graphs." *SIAM J. Discr. Math. 5*, 398–412.

38. F.C. Hennie (1966): On-line Turing machine computations. *IEEE Trans. Electronic Computers, EC-15*, 35–44.

39. F.C. Hennie and R.E. Stearns (1966): Two-tape simulation of multitape Turing machines. *J. ACM 13*, 533–546.

40. S. Homer and A.L. Selman (2001): *Computability and Complexity Theory*. Springer, New York

41. J.E. Hopcroft, R. Motwani, J.D. Ullman (2001): *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Addison-Wesley, Reading, MA.

42. J.E. Hopcroft and J.D. Ullman (1979): *Introduction to Automata Theory, Languages, and Computation* (1st ed.) Addison-Wesley, Reading, MA.

43. N. Immerman (1988): Nondeterministic space is closed under complementation. *SIAM J. Comput. 17*, 935–938.

44. *The Intel Philanthropic Peer-to-Peer program.* ⟨www.intel.com/cure⟩.

45. K.E. Iverson (1962): *A Programming Language*. J. Wiley & Sons, New York.

46. J. Jaffe (1978): A necessary and sufficient pumping lemma for regular languages. *SIGACT News*, 48–49.

47. R.M. Karp (1967): Some bounds on the storage requirements of sequential machines and Turing machines. *J. ACM 14*, 478–489.

48. R.M. Karp (1972): Reducibility among combinatorial problems. In *Complexity of Computer Computations* (R.E. Miller and J.W. Thatcher, eds.) Plenum Press, NY, pp. 85–103.

49. S.C. Kleene (1936): General recursive functions of natural numbers. *Math. Annalen 112*, 727–742.

50. S.C. Kleene (1952): *Introduction to Metamathematics*. D. Van Nostrand, Princeton, NJ.

51. S.C. Kleene (1956): Realization of events in nerve nets and finite automata. In *Automata Studies* (C.E. Shannon and J. McCarthy, Eds.) *[Ann. Math. Studies 34]*, Princeton Univ. Press, Princeton, NJ, pp. 3–42.

52. D. König (1936): *Theorie der endlichen und unendlichen Graphen.* Lipzig: Akad. Verlag.
53. D.E. Knuth (1973): *The Art of Computer Programming: Fundamental Algorithms* (2nd ed.) Addison-Wesley, Reading, MA.
54. E. Korpela, D. Werthimer, D. Anderson, J. Cobb, M. Lebofsky (2000): SETI@home: massively distributed computing for SETI. In *Computing in Science and Engineering* (P.F. Dubois, Ed.) IEEE Computer Soc. Press, Los Alamitos, CA.
55. P.J Landin (1964): The mechanical evaluation of expressions. *Computer J. 6*, 308–320.
56. L. Levin (1973): Universal search problems. *Problemy Peredachi Informatsii 9*, 265–266. Translated in, B.A. Trakhtenbrot (1984): A survey of Russian approaches to perebor (brute-force search) algorithms. *Annals of the History of Computing 6*, 384–400.
57. J.S. Lew and A.L. Rosenberg (1978): Polynomial indexing of integer lattices, I. *J. Number Th. 10*, 192–214.
58. J.S. Lew and A.L. Rosenberg (1978): Polynomial indexing of integer lattices, II. *J. Number Th. 10*, 215–243.
59. H.R. Lewis and C.H. Papadimitriou (1981): *Elements of the Theory of Computation*. Prentice-Hall, Englewood Cliffs, NJ.
60. P. Linz (2001): *An Introduction to Formal Languages and Automata* (3rd ed.) Jones and Bartlett Publ., Sudbury, MA.
61. A.A. Markov (1949): On the representation of recursive functions (in Russian). *Izvestiya Akademii Nauk S.S.S.R. 13*. English translation: Translation 54, Amer. Math. Soc., 1951.
62. W.S. McCulloch and W.H Pitts (1943): A logical calculus of the ideas immanent in nervous activity. *Bull. Mathematical Biophysics 5*, 115–133.
63. R. McNaughton and H. Yamada (1964): Regular expressions and state graphs for automata. In *Sequential Machines: Selected Papers* (E.F. Moore, ed.) Addison-Wesley, Reading, MA, PP. 157–176.
64. G.H. Mealy (1955): A method for synthesizing sequential circuits. *Bell Syst. Tech. J. 34*, 1045–1079.
65. K. Mehlhorn (1984): *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness.* Springer-Verlag, Berlin.
66. C.F. Miller (1991): Decision problems for groups – survey and reflections. In *Algorithms and Classification in Combinatorial Group Theory*, Springer, New York, pp. 1–60.
67. M. Minsky (1967): *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., Englewood Cliffs, NJ.
68. E.F. Moore (1956): Gendanken experiments on sequential machines. In *Automata Studies* (C.E. Shannon and J. McCarthy, eds.) *[Ann. Math. Studies 34]*, Princeton Univ. Press, Princeton, NJ, pp. 129–153.
69. B.M. Moret (1997): *The Theory of Computation*. Addison-Wesley, Reading, MA.
70. J. Myhill (1957): Finite automata and the representation of events. WADD TR-57-624, Wright Patterson AFB, Ohio, pp. 112–137.
71. T. Naur (2006): Letter to the Editor. *Comm. ACM 49*, 13.
72. A. Nerode (1958): Linear automaton transformations. *Proc. AMS 9*, 541–544.
73. I. Niven and H.S. Zuckerman (1980): *An Introduction to the Theory of Numbers.* (4th ed.) J. Wiley & Sons, New York.
74. *The Olson Laboratory Fight AIDS@Home project.* ⟨www.fightaidsathome.org⟩.
75. G.H. Ott, N.H. Feinstein (1961): Design of sequential machines from their regular expressions. *J. ACM 8*, 585–600.
76. C.H. Papadimitriou (1994): *Computational Complexity*. Addison-Wesley, Reading, MA.
77. M.O. Rabin (1963): Probabilistic automata. *Inform. Control 6*, 230–245.
78. M.O. Rabin (1964): The word problem for groups. *J. Symbolic Logic 29*, 205–206.
79. M.O. Rabin and D. Scott (1959): Finite automata and their decision problems. *IBM J. Res. Develop. 3*, 114–125.
80. H. Rogers, Jr. (1967): *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York. Reprinted in 1987 by MIT Press, Cambridge, MA.
81. A.L. Rosenberg (1971): Data graphs and addressing schemes. *J. CSS 5*, 193–238.

82. A.L. Rosenberg (1974): Allocating storage for extendible arrays. *J. ACM 21*, 652–670.

83. A.L. Rosenberg (1975): Managing storage for extendible arrays. *SIAM J. Comput. 4*, 287–306.

84. A.L. Rosenberg (2003): Accountable Web-computing. *IEEE Trans. Parallel and Distr. Systs. 14*, 97–106.

85. A.L. Rosenberg (2003): Efficient pairing functions—and why you should care. *Intl. J. Foundations of Computer Science 14*, 3–17.

86. A.L. Rosenberg (2006): State. In *Theoretical Computer Science: Essays in Memory of Shimon Even* (O. Goldreich, A.L. Rosenberg, A. Selman, eds.) *Springer Festschrift series, Lecture Notes in Computer Science 3895*, Springer, Heidelberg, pp. 375–398.

87. A.L. Rosenberg (2009): *The Pillars of Computation Theory: State, Encoding, Nondeterminism*. Universitext Series, Springer, New York

88. A.L. Rosenberg and L.S. Heath (2001): *Graph Separators, with Applications*. Kluwer Academic/Plenum Publishers, New York.

89. A.L. Rosenberg and L.J. Stockmeyer (1977): Hashing schemes for extendible arrays. *J. ACM 24*, 199–221.

90. J.B. Rosser (1953): *Logic for Mathematicians*. McGraw-Hill, New York.

91. *The RSA Factoring by Web Project.* ⟨http://www.npac.syr.edu/factoring⟩ (with Foreword by A. Lenstra). Northeast Parallel Architecture Center.

92. B. Russell, Bertrand (1903). Principles of Mathematics. Cambridge: Cambridge University Press.

93. W. Savitch (1969): Deterministic simulation of non-deterministic Turing machines. *1st ACM Symp. on Theory of Computing*, 247–248.

94. M. Schönfinkel (1924): Über die Bausteine der mathematischen Logik. *Math. Annalen 92*, 305–316.

95. A. Schönhage (1980): Storage modification machines. *SIAM J. Computing 9*, 490–508.

96. E. Schröder (1898): Über zwei Definitionen der Endlichkeit und G. Cantor'sche Sätze. *Nova Acta Academiae Caesareae Leopoldino-Carolinae (Halle a.d. Saale) 71*, 303–362.

97. E. Schröder (1898): Die selbständige Definition der Mächtigkeiten 0, 1, 2, 3 und die explicite Gleichzahligkeitsbedingung. *Nova Acta Academiae Caesareae Leopoldino-Carolinae (Halle a.d. Saale) 71*, 365–376.

98. M. Sipser (1997): *Introduction to the Theory of Computation*. PWS Publishing, Boston, MA.

99. R.E. Stearns, J. Hartmanis, P.M. Lewis, II (1972): Hierarchies of memory limited computations. *J. Symbolic Logic 37*, 624–625.

100. L.J. Stockmeyer (1973): Extendible array realizations with additive traversal. IBM Research Report RC-4578.

101. R. Szelepcsényi (1987): The method of forcing for nondeterministic automata. *Bull. EATCS 33*, 96–100.

102. R.E. Tarjan (1972): Sorting using networks of queues and stacks. *J. ACM 19*, 341–346.

103. M. Taufer, D. Anderson, P. Cicotti, C.L. Brooks (2005): Homogeneous redundancy: A technique to ensure integrity of molecular simulation results using public computing. *19th Intl. Parallel and Distributed Processing Symp.*

104. R.P. Tewarson (1973): *Sparse Matrices*. In *Mathematics in Science & Engineering*. Academic Press, New York.

105. A.M. Turing (1936): On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.* (ser. 2, vol. 42) 230–265; Correction *ibid.* (vol. 43) 544–546.

106. S. Warshall (1962): A theorem on Boolean matrices. *J. ACM 9*, 11-12.

107. Wikipedia: The Free Encyclopedia (2005): http://en.wikipedia.org/wiki/Pumping_lemma

108. R. Zach (1999): Completeness before Post: Bernays, Hilbert, and the development of propositional logic. *Bull. Symbolic Logic 5*, 331–366.

Index