chapter 1

Elementary Notions and Notations

'Excellent!' I cried. 'Elementary,' said he.

—Watson in *The Crooked Man*
by Arthur Conan Doyle (1859–1930)

To communicate, we sometimes need to agree on the meaning of certain terms. If the same idea is mentioned several times in a discussion, we often replace it with some shorthand notation. The choice of notation can help us avoid wordiness and ambiguity, and it can help us achieve conciseness and clarity in our written and oral expression.

Many problems of computer science, as well as other areas of thought, deal with reasoning about things and representing things. Since much of our communication involves reasoning about things, we'll begin the chapter with a short discussion about the notions of informal proof. The rest of the chapter is devoted to introducing the basic notions and notations for sets, tuples, graphs, and trees. The treatment here is introductory in nature, and we'll expand on these ideas in later chapters as the need arises.

chapter guide

Section 1.1 introduces some proof techniques that are used throughout the book. We'll practice each technique with a proof about numbers.

Section 1.2 introduces the basic ideas about sets. We'll see how to compare them and how to combine them, and we'll introduce some elementary ways to count them. We'll also introduce bags, which are like sets but that might contain repeated occurrences of elements, and we'll have a little discussion about why we should stick with uncomplicated sets.

Section 1.3 introduces some basic ideas about ordered structures and how to represent them. Tuples are introduced as a notation for ordered information. We'll introduce the notions and notations for lists, strings, and relations. We'll also see some elementary ways to count tuples.

Section 1.4 introduces the basic ideas about graphs and trees. We'll discuss ways to represent them, ways to traverse them, and we'll see a famous algorithm for constructing a spanning tree for a graph.

1.1 A Proof Primer

For our purposes an *informal proof* is a demonstration that some statement is true. We normally communicate an informal proof in an English-like language that mixes everyday English with symbols that appear in the statement to be proved. In the next few paragraphs we'll discuss some basic techniques for doing informal proofs. These techniques will come in handy in trying to understand someone's proof or in trying to construct a proof of your own, so keep them in your mental tool kit.

We'll start off with a short refresher on logical statements followed by a short discussion about numbers. This will give us something to talk about when we look at examples of informal proof techniques.

1.1.1 Logical Statements

For this primer we'll consider only statements that are either true or false. We'll start by discussing some familiar ways to structure logical statements.

Negation

If S represents some statement, then the *negation* of S is the statement "not S," whose truth value is opposite that of S. We can represent this relationship with a *truth table* in which each row gives a value for S and the corresponding value for not S:

S	not S
true	false
false	true

We often paraphrase the negation of a statement to make it more understandable. For example, to negate the statement "Earth is a star," we normally say, "Earth is not a star," or "It is not the case that Earth is a star," rather than "Not Earth is a star."

We should also observe that negation relates statements about *every* case with statements about *some* case. For example, the statement "Not every planet has a moon" has the same meaning as "Some planet does not have a moon." Similarly, the statement "It is not the case that some planet is a star" has the same meaning as "Every planet is not a star."

A	B	A and B	A or B
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Figure 1.1 Truth tables.

Conjunction and Disjunction

The *conjunction* of A and B is the statement "A and B," which is true when both A and B are true. The *disjunction* of A and B is the statement "A or B," which is true if either or both of A and B are true. The truth tables for conjunction and disjunction are given in Figure 1.1.

Sometimes we paraphrase conjunctions and disjunctions. For example, instead of "Earth is a planet and Mars is a planet," we might write "Earth and Mars are planets." Instead of "x is positive or y is positive," we might write "Either x or y is positive."

Conditional Statements

Many statements are written in the general form "If A then B," where A and B are also logical statements. Such a statement is called a *conditional statement* in which A is the *hypothesis* and B is the *conclusion*. We can read "If A then B" in several other ways: "A is a sufficient condition for B," or "B is a necessary condition for A," or simply "A implies B." The truth table for a conditional statement is contained in Figure 1.2.

Let's make a few comments about this table. Notice that the conditional is false only when the hypothesis is true and the conclusion is false. It's true in the other three cases. The conditional truth table gives some people fits because they interpret "If A then B" to mean "B can be proved from A," which assumes that A and B are related in some way. But we've all heard statements like "If the moon is made of green cheese, then $1 = 2$." We nod our heads and agree that the statement is true, even though there is no relationship between the hypothesis and conclusion. Similarly, we shake our heads and don't agree with a statement like "If $1 = 1$, then the moon is made of green cheese."

A	B	if A, then B
true	true	true
true	false	false
false	true	true
false	false	true

Figure 1.2 Truth table.

When the hypothesis of a conditional is false, we say that the conditional is *vacuously* true. For example, the statement "If $1 = 2$, then $39 = 12$" is vacuously true because the hypothesis is false. If the conclusion is true, we say that the conditional is *trivially* true. For example, the statement "If $1 = 2$, then $2 + 2 = 4$" is trivially true because the conclusion is true. We leave it to the reader to convince at least one person that the conditional truth table is defined properly.

The *converse* of "If A, then B" is "If B, then A." The converse does not always have the same truth value. For example, we know that the following statement about numbers is true.

$$\text{If } x > 0 \text{ and } y > 0, \text{ then } x + y > 0.$$

The converse of this statement is

$$\text{If } x + y > 0, \text{ then } x > 0 \text{ and } y > 0.$$

This converse is false. For example, let $x = 3$ and $y = -2$. Then the statement becomes "If $3 + (-2) > 0$, then $3 > 0$ and $-2 > 0$," which is false.

Equivalent Statements

Sometimes it's convenient to write a statement in a different form but with the same truth value. Two statements are said to be *equivalent* if they have the same truth value for any assignment of truth values to the variables that occur in the statements.

We can combine negation with either conjunction or disjunction to obtain the following pairs of equivalent statements.

"not (A and B)" is equivalent to "(not A) or (not B)."
"not (A or B)" is equivalent to "(not A) and (not B)."

For example, the statement "not ($x > 0$ and $y > 0$)" is equivalent to the statement "$x \leq 0$ or $y \leq 0$." The statement "not ($x > 0$ or $y > 0$)" is equivalent to the statement "$x \leq 0$ and $y \leq 0$."

Conjunctions and disjunctions distribute over each other in the following sense:

"A and (B or C)" is equivalent to "(A and B) or (A and C)."
"A or (B and C)" is equivalent to "(A or B) and (A or C)."

For example, the statement "$0 < x$ and ($x < 4$ or $x < 9$)" is equivalent to the statement "$0 < x < 4$ or $0 < x < 9$." The statement "$x > 0$ or ($x > 2$ and $x > 1$)" is equivalent to "($x > 0$ or $x > 2$) and ($x > 0$ or $x > 1$)."

The *contrapositive* of the conditional statement "If A, then B" is the equivalent statement "If not B, then not A." For example, the statement

$$\text{If } (x > 0 \text{ and } y > 0) \text{ then } x + y > 0$$

is equivalent to the statement

$$\text{If } x + y = 0, \text{ then } (x \leq 0 \text{ or } y \leq 0).$$

We can also express the conditional statement "If A, then B" in terms of the equivalent statement "(not A) or B." For example, the statement

$$\text{If } x > 0 \text{ and } y > 0, \text{ then } x + y > 0.$$

is equivalent to the statement

$$(x \leq 0 \text{ or } y \leq 0) \text{ or } x + y > 0.$$

Since we can express a conditional in terms of negation and disjunction, it follows that the statements "not (If A then B)" and "A and (not B)" are equivalent. For example, the statement

It is not the case that if Earth is a planet, then Earth is a star.

is equivalent to the statement

Earth is a planet and Earth is not a star.

Let's summarize the equivalences that we have discussed. Each row of the following table contains two equivalent statements S and T.

S	(is equivalent to) T
not (A and B)	(not A) or (not B)
not (A or B)	(not A) and (not B)
A and (B or C)	(A and B) or (A and C)
A or (B and C)	(A or B) and (A or C)
if A, then B	if not B, then not A
if A, then B	(not A) or B
not (if A then B)	A and (not B)

1.1.2 Something to Talk About

To discuss proof techniques, we need something to talk about when giving sample proofs. Since numbers are familiar to everyone, that's what we'll talk about. But to make sure that we all start on the same page, we'll review a little terminology.

The numbers that we'll be discussing are called *integers*, and we can list them as follows:

$$\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots .$$

The integers in the following list are called *even* integers:

$$..., -4, -2, 0, 2, 4, ...$$

The integers in the following list are called *odd* integers:

$$..., -3, -1, 1, 3, ...$$

So every integer is either even or odd but not both. In fact, every even integer has the form $2n$ for some integer n. Similarly, every odd integer has the form $2n + 1$ for some integer n.

Divisibility and Prime Numbers

An integer d *divides* an integer n if $d \neq 0$ and there is an integer k such that $n = dk$. For example, 3 divides 18 because we can write $18 = (3)(6)$. But 5 does not divide 18 because there is no integer k such that $18 = 5k$. The following list shows all the divisors of 18.

$$-18, -9, -6, -3, -2, -1, 1, 2, 3, 6, 9, 18.$$

Some alternative words for d *divides* n are d *is a divisor of* n or n *is divisible by* d. We often denote the fact that d divides n with the following shorthand notation:

$$d|n$$

For example, we have $-9|9$, $-3|9$, $-1|9$, $1|9$, $3|9$, and $9|9$. Here are two properties of divisibility that we'll record for future use.

Divisibility Properties (1.1)

a. If $d|a$ and $a|b$, then $d|b$.

b. If $d|a$ and $d|b$, then $d|(ax + by)$ for any integers x and y.

An integer $p > 1$ is called a *prime* number if 1 and p are its only positive divisors. For example, the first eight prime numbers are

$$2, 3, 5, 7, 11, 13, 17, 19.$$

Prime numbers have many important properties and they have many applications in computer science. But for now all we need to know is the definiton of a prime.

1.1.3 Proof Techniques

Now that we have something to talk about, we'll discuss some fundamental proof techniques and give some sample proofs for each technique.

Proof by Exhaustive Checking

When a statement asserts that each of a finite number of things has a certain property, then we might be able to prove the statement by checking that each thing has the stated property. For example, suppose someone says, "If n is an integer and $2 \leq n \leq 7$, then $n^2 + 2$ is not divisible by 4." We can prove the statement by *exhaustive checking.* For $2 \leq n \leq 7$, the corresponding values of $n^2 + 2$ are

$$6, 11, 18, 25, 38, 51.$$

We can check that these numbers are not divisible by 4. For another example, suppose someone says, "If n is an integer and $2 \leq n \leq 500$, then $n^2 + 2$ is not divisible by 4." Again, this statement can be proved by exhaustive checking, but perhaps by a computer rather than a person.

Exhaustive checking cannot be used to prove a statement that requires infinitely many things to check. For example, consider the statement, "If n is an integer, then $n^2 + 2$ is not divisible by 4." This statement is true, but there are infinitely many things to check. So another proof technique will be required. We'll get to it after a few more paragraphs.

An example that proves a statement false is often called a *counterexample.* Sometimes counterexamples can be found by exhaustive checking. For example, consider the statement, "Every odd number greater than 1 that is not prime has the form $2 + p$ for some prime p." We can observe that the statement is false because 27 is a counterexample.

Conditional Proof

Many statements that we wish to prove are in conditional form or can be phrased in conditional form (if A then B). The *direct approach* to proving such a statement starts with the assumption that the hypothesis A is true. The next step is to find a statement that is implied by the assumption or known facts. Each step proceeds in this fashion to find a statement that is implied by any of the previous statements or known facts. The *contitional proof* ends when the conclusion B is reached.

example **1.1 A Proof About Sums**

We'll prove the following general statement about integers:

The sum of any two odd integers is an even integer.

We can rephrase the statement in the conditional form

If x and y are odd integers, then $x + y$ is an even integer.

Proof: Assume the hypothesis that x and y are odd integers. It follows that x and y can be written in the the form $x = 2k + 1$ and $y = 2m + 1$, where k and m are arbitrary integers. Now substitute for x and y in $x + y$ to obtain

$$x + y = (2k + 1) + (2m + 1) = 2k + 2m + 2 = 2(k + m + 1).$$

Since the expression on the right-hand side contains 2 as a factor, it represents an even integer. QED.

end example

example

1.2 A Divisibility Proof

We'll prove the following statement (1.1a) about divisibility:

$$\text{If } d|a \text{ and } a|b, \text{ then } d|b.$$

Proof: Assume the hypothesis that $d|a$ and $a|b$. It follows from the definition of divisibility that there are integers m and n such that $a = dm$ and $b = an$. Now substitute for a in the first equation.

$$b = an = (dm)n = d(mn).$$

This equation says that $d|b$. QED.

end example

Proving the Contrapositive

Recall that a conditional statement "if A then B" and its contrapositive "if not B then not A" have the same truth table. So a proof of one is also a proof of the other. The *indirect approach* to proving "if A then B" is to *prove the contrapositve*. Start with the assumption that B is false. The next step is to find a statement that is implied by the assumption or known facts. Each step proceeds in this fashion to find a statement that is implied by any of the previous statements or known facts. The proof ends when an implied statement says that A is false.

example

1.3 An Odd Proof

We'll prove the following statement about the integers:

$$\text{If } x^2 \text{ is odd, then } x \text{ is odd.}$$

To prove the statement, we'll prove its contrapositive:

$$\text{If } x \text{ is even, then } x^2 \text{ is even.}$$

Proof: Assume the hypothesis (of the contrapositive) that x is even. It follows that $x = 2k$ for some integer k. Now square x and substitute for x to obtain

$$x^2 = (2k)^2 = 4k^2 = 2(2k^2).$$

The expression on the right side of the equation represents an even number. Therefore, x^2 is even. QED.

end example

Proof by Contradiction

A *contradiction* is a false statement. Another kind of indirect proof is *proof by contradiction*, where we start out by assuming that the statement to be proved is false. Then we argue until we reach a contradiction. Such an argument is often called a *refutation*.

Proof by contradiction is often the method of choice because we can wander wherever the proof takes us to find a contradiction. We'll give two examples to show the wandering that can take place.

example

1.4 A Not-Divisible Proof

We'll prove the following statement about divisibility:

If n is an integer, then $n^2 + 2$ is not divisible by 4.

Proof: Assume the statement is false. Then $4 \mid (n^2 + 2)$ for some intger n. This means that $n^2 + 2 = 4k$ for some integer k. We'll consider the two cases where n even and where n odd. If n is even, then $n = 2m$ for some integer m. Substituting for n we obtain

$$4k = n^2 + 2 = (2m)^2 + 2 = 4m^2 + 2.$$

We can divide both sides of the equations by 2 to obtain

$$2k = 2m^2 + 1.$$

This says that an even number $(2k)$ is equal to an odd number $(2m^2 + 1)$, which is a contradiction. Therefore, n cannot be even. Now assume n is odd. Then $n = 2m + 1$ for some integer m. Substituting for n we obtain

$$4k = n^2 + 2 = (2m + 1)^2 + 2 = 4m^2 + 4m + 3.$$

Isolate 3 on the right side of the equation to obtain

$$4k - 4m^2 - 4m = 3.$$

This is a contradiction because the left side is even and the right side is odd. Therefore, n cannot be odd. QED.

end example

example

1.5 Prime Numbers

We'll prove the following statement about integers:

Every integer greater than 1 is divisible by a prime.

Proof: Assume the statement is false. Then some integer $n > 1$ is not divisible by a prime. Since a prime divides itself, n cannot be a prime. So there is at least one integer d such that $d|n$ and $1 < d < n$. Assume that d is the smallest divisor of n between 1 and n. Now d is not prime, else it would be a prime divisor of n. So there is an integer a such that $a|d$ and $1 < a < d$. Since $a \mid d$ and $d \mid n$, have $a \mid d$ and $d \mid n$, it follows from Example 2 that $a \mid n$. But now we have $a \mid n$ and $1 < a < d$, which contradicts the assumption that d is the smallest such divisor of n. QED.

end example

If and Only If Proofs

The statement "A if and only if B" is shorthand for the two statements "If A, then B" and "If B, then A." The abbreviation "A *iff* B" is often used for "A if and only if B." Instead of "A iff B," some people write "A is a necessary and sufficient condition for B" or "B is a necessary and sufficient condition for A." Remember that two proofs are required for an iff statement, one for each conditional statement.

example

1.6 An Iff Proof

We'll prove the following iff statement about integers:

$$x \text{ is odd if and only if } 8 \mid (x^2 - 1).$$

To prove this iff statement, we must prove the following two statements:

a. If x is odd, then $8 \mid (x^2 - 1)$.

b. If $8 \mid (x^2 - 1)$, then x is odd.

Proof of (a): Assume x is odd. Then we can write x in the form $x = 2k + 1$ for some integer k. Substituting for x in $x^2 - 1$ gives

$$x^2 - 1 = 4k^2 + 4k = 4k(k + 1).$$

Since k and $k + 1$ are consecutive integers, one is odd and the other is even, so the product $k(k + 1)$ is even. So $k(k + 1) = 2m$ for some integer m. Substituting for $k(k + 1)$ gives

$$x^2 - 1 = 4k(k + 1) = 4(2m) = 8m.$$

Therefore, $8 \mid (x^2 - 1)$, so part (a) is proven.

Proof of (b): Assume $8 \mid (x^2 - 1)$. Then $x^2 - 1 = 8k$ for some integer k. Therefore, we have $x^2 = 8k + 1 = 2(4k) + 1$, which has the form of an odd integer. So x^2 is odd and it follows from Example 3 that x is odd, so part (b) is proven. Therefore, the iff statement is proven. QED.

end example

Sometimes we encounter iff statements that can be proven by using statements that are related to each other by iff. Then a proof can be constructed as a sequence of iff statements. For example, to prove A iff B we might be able to find a statement C such that A iff C and C iff B are both true. Then we can conclude that A iff B is true. The proof could then be put in the form A iff C iff B.

example

1.7 Two Proofs in One

We'll prove the following statement about integers:

$$x \text{ is odd if and only if } x^2 + 2x + 1 \text{ is even.}$$

Proof: The following sequence of iff statements connects the left side to the right side. (The reason for each step is given in parentheses.)

x is odd iff	$x = 2k + 1$ for some integer k	(definition)
iff	$x + 1 = 2k + 2$ for some integer k	(algebra)
iff	$x + 1 = 2m$ for some integer m	(algebra)
iff	$x + 1$ is given	(definition)
iff	$(x + 1)^2$ is even	(Exercise 8a)
iff	$x^2 + 2x + 1$ is even	(algebra) QED

end example

On Constructive Existence

If a statement asserts that some object exists, then we can try to prove the statement in either of two ways. One way is to use proof by contradiction, in which we assume that the object does not exist and then come up with some kind of contradiction. The second way is to construct an instance of the object. In either case we know that the object exists, but the second way also gives us an

instance of the object. Computer science leans toward the construction of objects by algorithms. So the *constructive approach* is usually preferred, although it's not always possible.

Important Note

Always try to write out your proofs. Use complete sentences that describe your reasoning. If your proof seems to consist only of a bunch of equations or expressions, you still need to describe how they contribute to the proof. Try to write your proofs the same way you would write a letter to a friend who wants to understand what you have written.

Exercises

1. See whether you can convince yourself, or a friend, that the conditional truth table is correct by making up English sentences of the form "If A, then B."

2. Verify that the truth tables for each of the following pairs of statements are identical.
   a. "not (A and B)" and "(not A) or (not B)."
   b. "not (A or B)" and "(not A) and (not B)."
   c. "if A, then B" and "if (not B), then (not A)."
   d. "if A, then B" and "(not A) or B."
   e. "not (if A then B)" and "A and (not B)."

3. Prove or disprove each of the following statements by exhaustive checking.
   a. There is a prime number between 45 and 54.
   b. The product of any two of the four numbers 2, 3, 4, and 5 is even.
   c. Every odd integer between 2 and 26 is either prime or the product of two primes.
   d. If $d \mid ab$, then $d \mid a$ or $d \mid b$.
   e. If m and n are integers, then $(3m + 2)(3n + 2)$ has the form $(3k + 2)$ for some integer k.

4. Prove each of the following statements about the integers.
   a. If x and y are even, then $x + y$ is even.
   b. If x is even and y is odd, then $x + y$ is odd.
   c. If x and y are odd, then $x - y$ is even.
   d. If $3n$ is even, then n is even.

5. Write down the converse of the following statement about integers:

   If x and y are odd, then $x - y$ is even.

   Is the statement that you wrote down true or false? Prove your answer.

6. Prove each of the following statements, where m and n are integers.

a. If $x = 3m + 4$ and $y = 3n + 4$, then $xy = 3k + 4$ for some integer k.

b. If $x = 5m + 6$ and $y = 5n + 6$, then $xy = 5k + 6$ for some integer k.

c. If $x = 7m + 8$ and $y = 7n + 8$, then $xy = 7k + 8$ for some integer k.

7. Prove each of the following statements about divisibility of integers.

a. If $d|(da + b)$, then $d|b$.

b. If $d \mid (a + b)$ and $d|a$, then $d|b$.

c. (1.1b) If $d|a$ and $d|b$, then $d|(ax + by)$ for any integers x and y.

8. Prove each of the following iff statements about integers.

a. x is even if and only if x^2 is even.

b. xy is odd if and only if x is odd and y is odd.

c. x is odd if and only if $x^2 + 6x + 9$ is even.

d. $m \mid n$ and $n \mid m$ if and only if $n = m$ or $n = -m$.

1.2 Sets

In our everyday discourse we sometimes run into the problem of trying to define a word in terms of other words whose definitions may include the word we are trying to define. That's the problem we have in trying to define the word *set*. To illustrate the point, we often think of some (perhaps all) of the words

set, collection, bunch, group, class

as synonyms for each other. We pick up the meaning for such a word intuitively by seeing how it is used.

1.2.1 Definition of a Set

We'll simply say that a *set* is a collection of things called its *elements*, *members*, or *objects*. Sometimes the word *collection* is used in place of *set* to clarify a sentence. For example, "a collection of sets" seems clearer than "a set of sets." We say that a set contains its elements, or that the elements belong to the set, or that the elements are in the set. If S is a set and x is an element in S, then we write

$$x \in S.$$

If x is not an element of S, then we write $x \notin S$. If $x \in S$ and $y \in S$, we often denote this fact by the shorthand notation

$$x, y \in S.$$

Describing Sets

To describe a set we need to describe its elements in some way. One way to define a set is to explicitly name its elements. A set defined in this way is denoted by listing its elements, separated by commas, and surrounding the listing with braces. For example, the set S consisting of the letters x, y, and z is denoted by

$$S = \{x, y, z\}.$$

Sets can have other sets as elements. For example, the set $A = \{x, \{x, y\}\}$ has two elements. One element is x, and the other element is $\{x, y\}$. So we can write $x \in A$ and $\{x, y\} \in$ A.

An important characteristic of sets is that there are no repeated occurrences of elements. For example, $\{H, E, L, L, O\}$ is not a set since there are two occurrences of the letter L.

We often use the three-dot ellipsis, ..., to informally denote a sequence of elements that we do not wish to write down. For example, the set

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

can be denoted in several different ways with ellipses, two of which are

$$\{1, 2, \ldots, 12\} \text{ and } \{1, 2, 3, \ldots, 11, 12\}.$$

The set with no elements is called the *empty set*—some people refer to it as the *null set*. The empty set is denoted by { } or more often by the symbol

$$\varnothing .$$

A set with one element is called a *singleton*. For example, $\{a\}$ and $\{b\}$ are singletons.

Equality of Sets

Two sets are *equal* if they have the same elements. We denote the fact that two sets A and B are equal by writing

$$A = B.$$

An important characteristic of sets is that here is no particular order or arrangement of the elements. For example, the set whose elements are g, h, and u can be represented in many different ways, two of which are

$$\{u, g, h\} = \{h, u, g\}.$$

If the sets A and B are not equal, we write

$$A \neq B.$$

For example, $\{a, b, c\} \neq \{a, b\}$ because c is an element of only one of the sets. We also have $\{a\} \neq \varnothing$ because the empty set doesn't have any elements.

Before we go any further let's record the two important characteristics of sets that we have discussed.

Two Characteristics of Sets

1. There are no repeated occurrences of elements.
2. There is no particular order or arrangement of the elements.

Finite and Infinite Sets

Suppose we start counting the elements of a set S, one element per second of time with a stop watch. If $S = \varnothing$, then we don't need to start, because there are no elements to count. But if $S \neq \varnothing$, we agree to start the counting after we have started the timer. If a point in time is reached when all the elements of S have been counted, then we stop the timer, or in some cases we might need to have one of our descendants stop the timer. In this case we say that S is a *finite* set. If the counting never stops, then S is an *infinite* set. All the examples that we have discussed to this point are finite sets. We will discuss counting finite and infinite sets in other parts of the book as the need arises.

Natural Numbers and Integers

Familiar infinite sets are sometimes denoted by listing a few of the elements followed by an ellipsis. We reserve some letters to denote specific sets that we'll refer to throughout the book. For example, the set of *natural numbers* will be denoted by $\mathbb{N}$[1] and the set of *integers* by $\mathbb{Z}$. So we can write

$$\mathbb{N} = \{0, 1, 2, 3, \dots\} \quad \text{and} \quad \mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}.$$

Describing Sets by Properties

Many sets are hard to describe by listing elements. Examples that come to mind are the *rational numbers*, which we denote by $\mathbb{Q}$, and the *real numbers*, which we denote by $\mathbb{R}$. Instead of listing the elements we can often describe a property that the elements of the set satisfy. For example, the set of odd integers consists of integers having the form $2k + 1$ for some integer k.

[1]Some people consider the natural numbers to be the set $\{1, 2, 3, \dots\}$. If you are one of these people, then think of $\mathbb{N}$ as the nonnegative integers.

If P is a property, then the set S whose elements have property P is denoted by writing

$$S = \{x \mid x \text{ has property } P\}.$$

We read this as "S is the set of all x such that x has property P." For example, if we let Odd be the set of odd integers, then we can describe Odd in several ways.

$$\begin{aligned} Odd &= \{\ldots, -5, -3, -1, 1, 3, 5, \ldots\} \\ &= \{x | x \text{ is an odd integer}\} \\ &= \{x | x = 2k + 1 \text{ for some integer } k\} \\ &= \{x | x = 2k + 1 \text{ for some } k \in \mathbb{Z}\}. \end{aligned}$$

Of course, we can also describe finite sets by finding properties that they possess. For example,

$$\{1, 2, \ldots, 12\} = \{x \mid x \in \mathbb{N} \text{ and } 1 = x = 12\}.$$

We can also describe a set by writing expressions for the elements. For example, the set Odd has the following additional descriptions.

$$\begin{aligned} Odd &= \{2k + 1 | k \text{ is an integer}\}. \\ &= \{2k + 1 | k \in \mathbb{Z}\}. \end{aligned}$$

Subsets

If A and B are sets and every element of A is also an element of B, then we say that A is a *subset* of B and write

$$A \subset B.$$

For example, we have $\{a, b\} \subset \{a, b, c\}$, $\{0, 1, 2\} \subset \mathbb{N}$, and $\mathbb{N} \subset \mathbb{Z}$. It follows from the definition that every set A is a subset of itself. Thus we have $A \subset A$. It also follows from the definition that the empty set is a subset of any set A. So we have $\varnothing \subset A$. Can you see why? We'll leave this as an exercise.

If $A \subset B$ and there is some element in B that does not occur in A, then A is called a *proper* subset of B. For example, $\{a, b\}$ is a proper subset of $\{a, b, c\}$. We also conclude that $\mathbb{N}$ is a proper subset of $\mathbb{Z}$, $\mathbb{Z}$ is a proper subset of $\mathbb{Q}$, and $\mathbb{Q}$ is a proper subset of $\mathbb{R}$.

If A is not a subset of B, we sometimes write

$$A \not\subset B.$$

For example, $\{a, b\} \not\subset \{a, c\}$ and $\{0, -1, -2\} \not\subset \mathbb{N}$. Remember that the idea of subset is different from the idea of membership. For example, if $A = \{a, b, c\}$, then $\{a\} \subset A$ and $a \in A$. But $\{a\} \notin A$ and $a \not\subset A$. For another example, let $A = \{a, \{b\}\}$. Then $a \in A$, $\{b\} \in A$, $\{a\} \subset A$, and $\{\{b\}\} \subset A$. But $b \notin A$ and $\{b\} \not\subset A$.

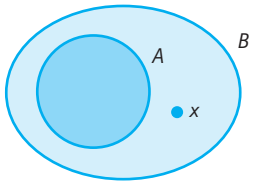


Figure 1.3 Venn diagram of proper subset $A \subset B$.

The Power Set

The collection of all subsets of a set S is called the *power set* of S, which we denote by power(S). For example, if $S = \{a, b, c\}$, then the power set of S can be written as follows:

$$\text{power}(S) = \{\varnothing\ , \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, S\}.$$

An interesting programming problem is to construct the power set of a finite set. We'll discuss this problem later, once we've developed some tools to help build an easy solution.

Venn Diagrams

In dealing with sets, it's often useful to draw a picture in order to visualize the situation. A *Venn diagram*—named after the logician John Venn (1834–1923)—consists of one or more closed curves in which the interior of each curve represents a set. For example, the Venn diagram in Figure 1.3 represents the fact that A is a proper subset of B and x is an element of B that does not occur in A.

Proof Strategies with Subsets and Equality

Subsets allow us to give a precise definition of set equality: Two sets are equal if they are subsets of each other. In more concise form we can write

Equality of Sets (1.2)

$$A = B \text{ means } A \subset B \text{ and } B \subset A$$

Let's record three useful proof strategies for comparing two sets.

Statement to Prove	*Proof Strategy*
$A \subset B$	For arbitrary $x \in A$, show that $x \in B$.
$A \not\subset B$	Find an element $x \in A$ such that $x \notin B$.
$A = B$	Show that $A \subset B$ and show that $B \subset A$.

example **1.8 Subset Proof**

We'll show that $A \subset B$, where A and B are defined as follows:

$$A = \{x \mid x \text{ is a prime number and } 42 \leq x \leq 51\},$$
$$B = \{x \mid x = 4k + 3 \text{ and } k \in \mathbb{N}\}.$$

We start the proof by letting $x \in A$. Then either $x = 43$ or $x = 47$. We can write $43 = 4(10) + 3$ and $47 = 4(11) + 3$. So in either case, x has the form of an element of B. Thus $x \in B$. Therefore, $A \subset B$.

end example

example **1.9 Not-Subset Proof**

We'll show that $A \not\subset B$ and $B \not\subset A$, where A and B are defined by

$$A = \{3k + 1 \mid k \in \mathbb{N}\} \text{ and } B = \{4k + 1 \mid k \in \mathbb{N}\}.$$

By listing a few elements from each set we can write A and B as follows:

$$A = \{1, 4, 7, \ldots\} \text{ and } B = \{1, 5, 9, \ldots\}.$$

Now it's easy to prove that $A \not\subset B$ because $4 \in A$ and $4 \notin B$. We can also prove that $B \not\subset A$ by observing that $5 \in B$ and $5 \notin A$.

end example

example **1.10 Equal Sets Proof**

We'll show that $A = B$, where A and B are defined as follows:

$$A = \{x \mid x \text{ is prime and } 12 \leq x \leq 18\},$$
$$B = \{x \mid x = 4k + 1 \text{ and } k \in \{3, 4\}\}.$$

First we'll show that $A \subset B$. Let $x \in A$. Then either $x = 13$ or $x = 17$. We can write $13 = 4(3) + 1$ and $17 = 4(4) + 1$. It follows that $x \in B$. Therefore, $A \subset B$. Next we'll show that $B \subset A$. Let $x \in B$. It follows that either $x = 4(3) + 1$ or $x = 4(4) + 1$. In either case, x is a prime number between 12 and 18. Therefore, $B \subset A$. So $A = B$.

end example

1.2.2 Operations on Sets

We'll discuss the operations of union, intersection, and complement, all of which combine sets to form new sets.

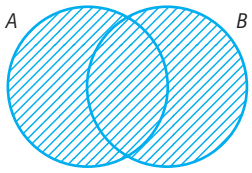


Figure 1.4 Venn diagram of $A \cup B$.

Union of Sets

The *union* of two sets A and B is the set of all elements that are either in A or in B or in both A and B. The union is denoted by $A \cup B$ and we can give the following formal definition.

Union of Sets (1.3)

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}.$$

The use of the word "or" in the definition is taken to mean "either or both." For example, if $A = \{a, b, c\}$ and $B = \{c, d\}$, then $A \cup B = \{a, b, c, d\}$. The union of two sets A and B is represented by the shaded regions of the Venn diagram in Figure 1.4.

The following properties give some basic facts about the union operation.

Properties of Union (1.4)

a. $A \cup \varnothing = A$.
b. $A \cup B = B \cup A$. ($\cup$ is commutative.)
c. $A \cup (B \cup C) = (A \cup B) \cup C$. ($\cup$ is associative.)
d. $A \cup A = A$.
e. $A \subset B$ if and only if $A \cup B = B$.

example 1.11 A Subset Condition

We'll prove the following statement (1.4e) and leave the other parts as exercises.

$$A \subset B \text{ if and only if } A \cup B = B.$$

Proof: Since this is an if and only if statement we have two statements to prove. First we'll prove that $A \subset B$ implies $A \cup B = B$. Assume that $A \subset B$. With this assumption we must show that $A \cup B = B$. Let $x \in A \cup B$. It follows that $x \in A$ or $x \in B$. Since we have assumed that $A \subset B$, it follows that $x \in B$. Thus $A \cup B \subset B$. But since we always have $B \subset A \cup B$, it follows from (1.2) that

$A \cup B = B$. So the first part is proven. Next we'll prove that $A \cup B = B$ implies $A \subset B$. Assume that $A \cup B = B$. If $x \in A$, then $x \in A \cup B$. Since we are assuming that $A \cup B = B$ it follows that $x \in B$. Therefore, $A \subset B$. So the second part is proven. QED.

end example

The union operation can be defined for an arbitrary collection of sets in a natural way. For example, the union of the n sets $A_1 \ldots, A_n$ can be denoted in the following way:

$$\bigcup_{i=1}^{n} A_i = A_1 \cup \cdots \cup A_n.$$

The union of the infinite collection of sets A_1, A_2, $\ldots$, A_n, $\ldots$ can be denoted in the following way.

$$\bigcup_{i=1}^{\infty} A_i = A_1 \cup \cdots \cup A_n \cup \cdots .$$

If I is a set of indices and A_i is a set for each $i \in I$, then the union of the sets in the collection can be denoted in the following way:

$$\bigcup_{i \in I} A_i.$$

example

1.12 English Words as a Union

Let W be the set of all words in the English language. Then we can represent W as an infinite union of sets. For each $i > 0$, let A_i denote the set of all words with i letters. Then W has the following representation:

$$W = \bigcup_{i=1}^{\infty} A_i.$$

If no English word has more than 25 letters, then $A_i = \varnothing$ for $i > 25$. In this case we could write W as the finite union

$$W = \bigcup_{i=1}^{25} A_i.$$

end example

example **1.13 An Infinite Union of Finite Sets**

Suppose we want to calculate the union of the sets $A_i = \{-2i, 2i\}$ where i is an odd natural number. For example, $A_3 = \{-6, 6\}$ and $A_5 = \{10, -10\}$. If we let Odd be the set of odd natural numbers, then we can write

$$\bigcup_{i \in \text{Odd}} A_i = \{\ldots, -10, -6, -2, 2, 6, 10, \ \ldots\}.$$

end example

Intersection of Sets

The *intersection* of two sets A and B is the set of all elements that are in both A and B. The intersection is denoted by $A \cap B$ and we can give the following formal definition.

Intersection of Sets (1.5)

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}.$$

For example, if $A = \{a, b, c\}$ and $B = \{c, d\}$, then $A \cap B = \{c\}$. If $A \cap B = \varnothing$, then A and B are said to be *disjoint*. The nonempty intersection of two sets A and B is represented by the shaded region of the Venn diagram in Figure 1.5.

The following properties give some basic facts about the intersection operation.

Properties of Intersection (1.6)

a. $A \cap \varnothing = \varnothing$.

b. $A \cap B = B \cap A$. ($\cap$ is commutative.)

c. $A \cap (B \cap C) = (A \cap B) \cap C$. ($\cap$ is associative.)

d. $A \cap A = A$.

e. $A \subset B$ if and only if $A \cap B = A$.

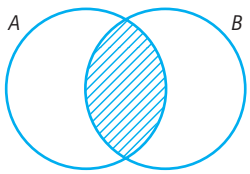


Figure 1.5 Venn diagram of $A \cap B$.

The intersection operation can be defined for an arbitrary collection of sets in a natural way. For example, the intersection of the n sets $A_1, \ldots, A_n$ can be denoted in the following way:

$$\bigcup_{i=1}^{n} A_i = A_1 \cap \cdots \cap A_n.$$

The intersection of the infinite collection of sets $A_1, A_2, \ldots, A_n, \ldots$ can be denoted in the following way:

$$\bigcap_{i=1}^{\infty} A_i = A_1 \cap \cdots \cap A_n \cap \cdots .$$

If I is a set of indices and A_i is a set for each $i \in I$, then the intersection of the sets in the collection can be denoted in the following way:

$$\bigcap_{i \in I} A_i.$$

example **1.14 An Infinite Intersection**

For each odd natural number i let $A_i = \{x \mid x \in \mathbb{N} \text{ and } -i \leq x \leq i\}$. For example, $A_3 = \{-3, -2, -1, 0, 1, 2, 3\}$. Letting Odd be the set of odd natural numbers, we can represent the intersection of the sets A_i as

$$\bigcap_{i \in Odd} A_i = \{-1, 0, 1\}.$$

end example

Difference of Sets

If A and B are sets, then the *difference* $A - B$ (also called the *relative complement* of B in A) is the set of elements in A that are not in B, which we can describe as a difference of sets.

Difference of Sets (1.7)

$$A - B = \{x \mid x \in A \text{ and } x \notin B\}.$$

For example, if $A = \{a, b, c\}$ and $B = \{c, d\}$, then $A - B = \{a, b\}$. We can picture the difference $A - B$ of two general sets A and B by the shaded region of the Venn diagram in Figure 1.6.

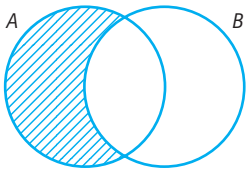


Figure 1.6 Venn diagram of $A - B$.

A natural extension of the difference $A - B$ is the *symmetric difference* of sets A and B, which is the union of $A - B$ with $B - A$ and is denoted by $A \oplus B$. The set $A \oplus B$ is represented by the shaded regions of the Venn diagram in Figure 1.7.

We can define the symmetric difference by using the "exclusive" form of "or" as follows:

Symmetric Difference of Sets (1.8)

$$A \oplus B = \{x \mid x \in A \text{ or } x \in B \text{ but not both}\}.$$

As is usually the case, there are many relationships to discover. For example, it's easy to see that

$$A \oplus B = (A \cup B) - (A \cap B).$$

Can you verify that $(A \oplus B) \oplus C = A \oplus (B \oplus C)$? For example, try to draw two Venn diagrams, one for each side of the equation.

Complement of a Set

If the discussion always refers to sets that are subsets of a particular set U, then U is called the *universe of discourse*, and the difference $U - A$ is called the *complement* of A, which we denote by A'. The Venn diagram in Figure 1.8 pictures the universe U as a rectangle, with two subsets A and B, where the shaded region represents the complement $(A \cup B)'$.

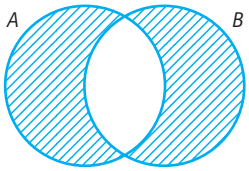


Figure 1.7 Venn diagram of $A \oplus B$.

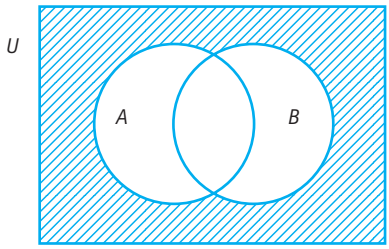


Figure 1.8 Venn diagram of $(A \cup B)'$.

Combining Set Operations

There are many useful properties that combine different set operations. Venn diagrams are often quite useful in trying to visualize sets that are constructed with different operations. For example, the set $A \cap (B \cup C)$ is represented by the shaded regions of the Venn diagram in Figure 1.9.

Here are two distributive properties and an absorption property that combine the operations of union and intersection.

Combining Properties of Union and Intersection (1.9)

a.	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.	($\cap$ distributes over $\cup$.)
b.	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.	($\cup$ distributes over $\cap$.)
c.	$A \cap (A \cup B) = A$.	(absorption law)
d.	$A \cup (A \cap B) = A$.	(absorption law)

example 1.15 A Distributive Proof

We'll prove the following statement (1.9a) about distribution:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

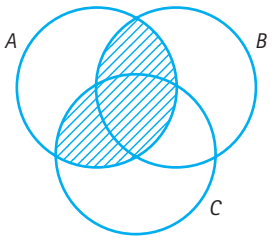


Figure 1.9 Venn diagram of $A \cap (B \cup C)$.

Proof: We'll show that $x \in A \cap (B \cup C)$ if and only if $x \in (A \cap B) \cup (A \cap C)$.

$$\begin{aligned} x \in A \cap (B \cup C) \text{ iff } & x \in A \text{ and } x \in B \cup C \\ \text{iff } & x \in A, \text{ and either } x \in B \text{ or } x \in C \\ \text{iff } & \text{either } (x \in A \text{ and } x \in B) \text{ or } (x \in A \text{ and } x \in C) \\ \text{iff } & x \in (A \cap B) \cup (A \cap C). \quad \text{QED.} \end{aligned}$$

end example

The complement operation combines with the other operations in many interesting ways. Here's a list of some of the useful properties.

Properties of Complement (1.10)

a. $(A')' = A$.
b. $\varnothing' = U$ and $U' = \varnothing$.
c. $A \cap A' = \varnothing$ and $A \cup A' = U$.
d. $A \subset B$ if and only if $B' \subset A'$.
e. $(A \cup B)' = A' \cap B'$ (De Morgan's law).
f. $(A \cap B)' = A' \cup B'$ (De Morgan's law).
g. $A \cap (A' \cup B) = A \cap B$ (absorption law).
h. $A \cup (A' \cap B) = A \cup B$ (absorption law).

example **1.16 Subset Conditions**

We'll prove the following statement (1.10d):

$$A \subset B \text{ if and only if } B' \subset A'.$$

Proof: In this case we're able to connect the two sides of the iff statement with a sequence of iff statements. Be sure that you know the reason for each step.

$$\begin{aligned} A \subset B \text{ iff } & x \in A \text{ implies } x \in B \\ \text{iff } & x \notin B \text{ implies } x \notin A \\ \text{iff } & x \in B' \text{ implies } x \in A' \\ \text{iff } & B' \subset A'. \quad \text{QED.} \end{aligned}$$

end example

1.2.3 Counting Finite Sets

Let's apply some of our knowledge about sets to counting finite sets. The size of a set S is called its *cardinality*, which we'll denote by

$$|S|.$$

For example, if $S = \{a, b, c\}$, then $|S| = |\{a, b, c\}| = 3$. We can say "the cardinality of S is 3," or "3 is the cardinal number of S," or simply "S has three elements."

Counting by Inclusion and Exclusion

Suppose we want to count the union of two sets. For example, suppose we have $A = \{1, 2, 3, 4, 5\}$ and $B = \{2, 4, 6, 8\}$. Since $A \cup B = \{1, 2, 3, 4, 5, 6, 8\}$, it follows that $|A \cup B| = 7$. Similarly, since $A \cap B = \{2, 4\}$, it follows that $|A \cap B| = 2$. If we know any three of the four numbers $|A|$, $|B|$, $|A \cap B|$, and $|A \cup B|$, then we can find the fourth by using the following counting rule for finite sets:

Union Rule (1.11)

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

It's easy to discover this rule by drawing a Venn diagram.

The union rule extends to three or more sets. For example, the following calculation gives the union rule for three finite sets:

$$\begin{aligned}
|A \cup B \cup C| &= |A \cup (B \cup C)| \\
&= |A| + |B \cup C| - |A \cap (B \cup C)| \\
&= |A| + |B| + |C| - |B \cap C| - |A \cap (B \cup C)| \\
&= |A| + |B| + |C| - |B \cap C| - |(A \cap B) \cup (A \cap C)| \\
&= |A| + |B| + |C| - |B \cap C| - |A \cap B| - |A \cap C| + |A \cap B \cap C|.
\end{aligned}$$

The popular name for the union rule and its extensions to three or more sets is the *principle of inclusion and exclusion*. The name is appropriate because the rule says to add (include) the count of each individual set. Then subtract (exclude) the count of all intersecting pairs of sets. Next, include the count of all intersections of three sets. Then exclude the count of all intersections of four sets, and so on.

example **1.17 A Building Project**

Suppose A, B, and C are sets of tools needed by three workers on a job. For convenience let's call the workers A, B, and C. Suppose further that the workers

share some of the tools (for example, on a housing project, all three workers might share a single table saw). Suppose that A uses 8 tools, B uses 10 tools, and C uses 5 tools. Suppose further that A and B share 3 tools, A and C share 2 tools, and B and C share 2 tools. Finally, suppose that A, B, and C share the use of 2 tools. How many distinct tools are necessary to do the job? Thus we want to find the value

$$|A \cup B \cup C|.$$

We can apply the result of (1.12) to obtain:

$$\begin{aligned} |A \cup B \cup C| &= |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C| \\ &= 8 + 10 + 5 - 3 - 2 - 2 + 2 \\ &= 18 \text{ tools.} \end{aligned}$$

end example

example **1.18 Surveys**

Suppose we survey 200 students to see whether they are taking courses in computer science, mathematics, or physics. The results show that 90 students take computer science, 110 take mathematics, and 60 take physics. Further, 20 students take computer science and mathematics, 20 take computer science and physics, and 30 take mathematics and physics. We are interested in those students that take courses in all three areas. Do we have enough information? Let C, M, and P stand for the sets of students that take computer science, mathematics, and physics. So we are interested in the number

$$|C \cap M \cap P|.$$

From the information given we know that $200 \geq |C \cup M \cup P|$. The reason for the inequality is that some students may not be taking any courses in the three areas. The statistics also give us the following information.

$$\begin{aligned} |C| &= 90 \\ |M| &= 110 \\ |P| &= 60 \\ |C \cap M| &= 20 \\ |C \cap P| &= 20 \\ |M \cap P| &= 30. \end{aligned}$$

Applying the result of (1.12), we have

$$\begin{aligned} 200 &\geq |C| + |M| + |P| - |C \cap M| - |C \cap P| - |M \cap P| + |C \cap M \cap P| \\ &= 90 + 110 + 60 - 20 - 20 - 30 + |C \cap M \cap P| \\ &= 190 + |C \cap M \cap P|. \end{aligned}$$

Therefore, $|C \cap M \cap P| \leq 10$. So we can say that at most 10 of the students polled take courses in all three areas. We would get an exact answer if we knew that each student in the survey took at least one course from one of the three areas.

end example

Counting the Difference of Two Sets

Suppose we need to count the difference of two sets. For example, if $A = \{1, 3, 5, 7, 9\}$ and $B = \{2, 3, 4, 8, 10\}$, then $A - B = \{1, 5, 7, 9\}$. So $|A - B| = 4$. If we know any two of the numbers $|A|$, $|A - B|$, and $|A \cap B|$, then we can find the third by using the following counting rule for finite sets.

Difference Rule (1.13)

$$|A - B| = |A| - |A \cap B|.$$

It's easy to discover this rule by drawing a Venn diagram. Two special cases of (1.13) are also intuitive and can be stated as follows:

$$\text{If } B \subset A, \text{ then } |A - B| = |A| - |B|. \tag{1.14}$$

$$\text{If } A \cap B = \varnothing, \text{ then } |A - B| = |A|. \tag{1.15}$$

example **1.19 Tool Boxes**

Continuing with the data from Example 10, suppose we want to know how many personal tools each worker needs (tools not shared with other workers). For example, Worker A needs a tool box of size

$$|A - (B \cup C)|.$$

We can compute this value using both the difference rule (1.13) and the union rule (1.11) as follows.

$$\begin{aligned} |A - (B \cup C)| &= |A| - |A \cap (B \cup C)| \\ &= |A| - |(A \cap B) \cup (A \cap C)| \\ &= |A| - (|(A \cap B)| + |(A \cap C)| - |A \cap B \cap C|) \\ &= 8 - (3 + 2 - 2) \\ &= 5 \text{ personal tools for A.} \end{aligned}$$

end example

1.2.4 Bags (Multisets)

A *bag* (or *multiset*) is a collection of objects that may contain repeated occurrences of elements. Here are the important characteristics.

Two Characteristics of Bags

1. There may be repeated occurrences of elements.

2. There is no particular order or arrangement of the elements.

To differentiate bags from sets, we'll use brackets to enclose the elements. For example, $[h, u, g, h]$ is a bag with four elements. Two bags A and B are *equal* if the number of occurrences of each element in A or B is the same in either bag. If A and B are equal bags, we write $A = B$. For example, $[h, u, g, h] = [h, h, g, u]$, but $[h, u, g, h] \neq [h, u, g]$.

We can also define the subbag notion. Define A to be a *subbag* of B, and write $A \subset B$, if the number of occurrences of each element x in A is less than or equal to the number of occurrences of x in B. For example, $[a, b] \subset [a, b, a]$, but $[a, b, a] \not\subset [a, b]$. It follows from the definition of subbag that two bags A and B are equal if and only if A is a subbag of B and B is a subbag of A.

If A and B are bags, we define the *sum* of A and B, denoted by $A + B$, as follows: If x occurs m times in A and n times in B, then x occurs $m + n$ times in $A + B$. For example,

$$[2, 2, 3] + [2, 3, 3, 4] = [2, 2, 2, 3, 3, 3, 4].$$

We can define union and intersection for bags also (we will use the same symbols as for sets). Let A and B be bags, and let m and n be the number of times x occurs in A and B, respectively. Put the larger of m and n occurrences of x in $A \cup B$. Put the smaller of m and n occurrences of x in $A \cap B$. For example, we have

$$[2, 2, 3] \cup [2, 3, 3, 4] = [2, 2, 3, 3, 4]$$

and

$$[2, 2, 3] \cap [2, 3, 3, 4] = [2, 3].$$

example 1.20 Least and Greatest

Let $p(x)$ denote the bag of prime numbers that occur in the prime factorization of the natural number x. For example, we have

$$p(54) = [2, 3, 3, 3] \text{ and } p(12) = [2, 2, 3].$$

Let's compute the union and intersection of these two bags. The union gives $p(54) \cup p(12) = [2, 2, 3, 3, 3] = p(108)$, and 108 is the least common multiple of 54 and 12 (i.e., the smallest positive integer that they both divide). Similarly, we get $p(54) \cap p(12) = [2, 3] = p(6)$, and 6 is the greatest common divisor of 54 and 12 (i.e., the largest positive integer that divides them both). Can we discover anything here? It appears that $p(x) \cup p(y)$ and $p(x) \cap p(y)$ compute the least common multiple and the greatest common divisor of x and y. Can you convince yourself?

end example

1.2.5 Sets Should Not Be Too Complicated

Set theory was created by the mathematician Georg Cantor (1845–1918) during the period 1874 to 1895. Later some contradictions were found in the theory. Everything works fine as long as we don't allow sets to be too complicated. Basically, we never allow a set to be defined by a test that checks whether a set is a member of itself. If we allowed such a thing, then we could not decide some questions of set membership. For example, suppose we define the set T as follows:

$$T = \{A \mid A \text{ is a set and } A \notin A\}.$$

In other words, T is the set of all sets that are not members of themselves. Now ask the question "Is $T \in T$?" If so, then the condition for membership in T must hold. But this says that $T \notin T$. On the other hand, if we assume that $T \notin T$, then we must conclude that $T \in T$. In either case we get a contradiction. This example is known as *Russell's paradox*—after the philosopher and mathematician Bertrand Russell (1872–1970).

This kind of paradox led to a more careful study of the foundations of set theory. For example, Whitehead and Russell [1910] developed a theory of sets based on a hierarchy of levels that they called *types*. The lowest type contains individual elements. Any other type contains only sets whose elements are from the next lower type in the hierarchy. We can list the hierarchy of types as T_0, $T_1, \ldots, T_k, \ldots$, where T_0 is the lowest type containing individual elements and in general T_{k+1} is the type consisting of sets whose elements are from T_k. So any set in this theory belongs to exactly one type T_k for some $k \geq 1$.

As a consequence of the definition, we can say that $A \notin A$ for all sets A in the theory. To see this, suppose A is a set of type T_{k+1}. This means that the elements of A are of type T_k. If we assume that $A \in A$, we would have to conclude that A is also a set of type T_k. This says that A belongs to the two types T_k and T_{k+1}, contrary to the fact that A must belong to exactly one type.

Let's examine why Russell's paradox can't happen in this new theory of sets. Since $A \notin A$ for all sets A in the theory, the original definition of T can be simplified to $T = \{A \mid A \text{ is a set}\}$. This says that T contains all sets. But

T itself isn't even a set in the theory because it contains sets of different types. In order for T to be a set in the theory, each A in T must belong to the same type. For example, we could pick some type T_k and define $T = \{A \mid A \text{ has type } T_k\}$. This says that T is a set of type T_{k+1}. Now since T is a set in the theory, we know that $T \notin T$. But this fact doesn't lead us to any kind of contradictory statement.

Exercises

Describing Sets

1. The set $\{x \mid x \text{ is a vowel}\}$ can also be described by $\{a, e, i, o, u\}$. Describe each of the following sets by listing its elements.

a. $\{x \mid x \in \mathbb{N} \text{ and } 0 < x < 8\}$.

b. $\{2k + 1 \mid k \text{ is an even integer between 1 and 10}\}$.

c. $\{x \mid x \text{ is an odd prime less than 20}\}$.

d. $\{x \mid x \text{ is a month ending with the letter "}y\text{"}\}$.

e. {x | x is a letter in the words MISSISSIPPI RIVER}.

f. $\{x \mid x \in \mathbb{N} \text{ and } x \text{ divides } 24\}$.

2. The set $\{a, e, i, o, u\}$ can also be described by $\{x \mid x \text{ is a vowel}\}$. Describe each of the following sets in terms of a property of its elements.

a. The set of dates in the month of January.

b. $\{1, 3, 5, 7, 9, 11, 13, 15\}$.

c. $\{1, 4, 16, 25, 36, 47, 64\}$.

d. The set of even integers $\{..., -4, -2, 0, 2, 4,...\}$.

Subsets

3. Let $A = \{a, \varnothing\}$. Answer true or false for each of the following statements.

a. $a \in A$. b. $\{a\} \in A$. **c.** $a \subset A$. d. $\{a\} \subset A$.

e. $\varnothing \subset A$. f. $\varnothing \in A$. **g.** $\{\varnothing\} \subset A$. h. $\{\varnothing\} \in A$.

4. Show that $\varnothing \subset A$ for every set A.

5. Find two finite sets A and B such that $A \in B$ and $A \subset B$.

6. Write down the power set for each of the following sets.

a. $\{x, y, z, w\}$. b. $\{a, \{a, b\}\}$. **c.** $\varnothing$.

d. $\{\varnothing\}$. e. $\{\{a\}, \varnothing\}$.

7. For each collection of sets, find the smallest set A such that the collection is a subset of power(A).

a. $\{\{a\}, \{b, c\}\}$. b. $\{\{a\}, \{\varnothing\}\}$. **c.** $\{\{a\}, \{\{a\}\}\}$.

d. $\{\{a\}, \{\{b\}\}, \{a, b\}\}$.

Set Operations

8. Suppose A and B are sets defined as follows:

$$A = \{x | x = 4k + 1 \text{ and } k \in \mathbb{N}\},$$
$$B = \{x | x = 3k + 5 \text{ and } k \in \mathbb{N}\}.$$

a. List ten elements of $A \cup B$.
b. List four elements of $A \cap B$.

9. Is $\text{power}(A \cup B) = \text{power}(A) \cup \text{power}(B)$?

10. For each integer i, define A_i as follows:

If i is even then $A_i = \{x \mid x \in \mathbb{Z} \text{ and } x < -i \text{ or } i < x\}$.
If i is odd then $A_i = \{x \mid x \in \mathbb{Z} \text{ and } -i < x < i\}$.

a. Describe each of the sets A_0, A_1, A_2, A_3, A_{-2}, and A_{-3}.
b. Find the union of the collection $\{A_i \mid i \in \{1, 3, 5, 7, 9\}\}$.
c. Find the union of the collection $\{A_i \mid i \text{ is even}\}$.
d. Find the union of the collection $\{A_i \mid i \text{ is odd}\}$.
e. Find the union of the collection $\{A_i \mid i \in \mathbb{N}\}$.
f. Find the intersection of the collection $\{A_i \mid i \in \{1, 3, 5, 7, 9\}\}$.
g. Find the intersection of the collection $\{A_i \mid i \text{ is even}\}$.
h. Find the intersection of the collection $\{A_i \mid i \text{ is odd}\}$.
i. Find the intersection of the collection $\{A_i \mid i \in \mathbb{N}\}$.

11. For each natural number n, let A_n be defined by

$$A_n = \{x \mid x \in \mathbb{N} \text{ and } x \text{ divides } n \text{ with no remainder}\}.$$

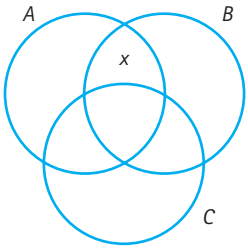
a. Describe each of the sets A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, and A_{100}.
b. Find the union of the collection $\{A_n \mid n \in \{1, 2, 3, 4, 5, 6, 7\}\}$.
c. Find the intersection of the collection $\{A_n \mid n \in \{1, 2, 3, 4, 5, 6, 7\}\}$.
d. Find the union of the collection $\{A_n \mid n \in \mathbb{N}\}$.
e. Find the intersection of the collection $\{A_n \mid n \in \mathbb{N}\}$.

12. For each of the following expressions, use a Venn diagram representing a universe U and two subsets A and B. Shade the part of the diagram that corresponds to the given set.

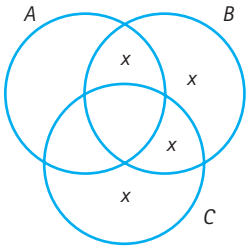
a. A'. b. B'. c. $(A \cup B)'$.
d. $A' \cap B'$. e. $A' \cup B'$. f. $(A \cap B)'$.

13. Each Venn diagram in the following figure represents a set whose regions are indicated by the letter x. Find an expression for each of the three sets in terms of set operations. Try to simplify your answers.

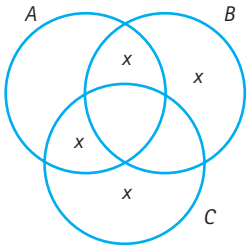
a.



b.



c.



Counting Finite Sets

14. Discover an inclusion exclusion formula for the number of elements in the union of four sets A, B, C, and D.

15. Given three sets A, B, and C. Suppose the union of the three sets has cardinality 280. Suppose also that $|A| = 100$, $|B| = 200$, and $|C| = 150$. And suppose we also know $|A \cap B| = 50$, $|A \cap C| = 80$, and $|B \cap C| = 90$. Find the cardinality of the intersection of the three given sets.

16. Suppose A, B, and C represent three bus routes through a suburb of your favorite city. Let A, B, and C also be sets whose elements are the bus stops for the corresponding bus route. Suppose A has 25 stops, B has 30 stops, and C has 40 stops. Suppose further that A and B share (have in common) 6 stops, A and C share 5 stops, and B and C share 4 stops. Lastly, suppose that A, B, and C share 2 stops. Answer each of the following questions.

    a. How many distinct stops are on the three bus routes?

    b. How many stops for A are not stops for B?

    c. How many stops for A are not stops for both B and C?

    d. How many stops for A are not stops for any other bus?

17. Suppose a highway survey crew noticed the following information about 500 vehicles: In 100 vehicles the driver was smoking, in 200 vehicles the driver was talking to a passenger, and in 300 vehicles the driver was tuning the radio. Further, in 50 vehicles the driver was smoking and talking, in 40 vehicles the driver was smoking and tuning the radio, and in 30 vehicles the driver was talking and tuning the radio. What can you say about the number of drivers who were smoking, talking, and tuning the radio?

18. Suppose the following people went to a summer camp: 27 boys, 15 city children, 27 men, 21 noncity boys, 42 people from the city, 18 city males, and 21 noncity females. How many people went to summer camp?

Bags

19. Find the union and intersection of each of the following pairs of bags.

a. $[x, y]$ & $[x, y, z]$.
b. $[x, y, x]$ & $[y, x, y, x]$.
c. $[a, a, a, b]$ & $[a, a, b, b, c]$.
d. $[1, 2, 2, 3, 3, 4, 4]$ & $[2, 3, 3, 4, 5]$.
e. $[x, x, [a, a], [a, a]]$ & $[a, a, x, x]$.
f. $[a, a, [b, b], [a, [b]]]$ & $[a, a, [b], [b]]$.

20. Find a bag B that solves the following two simultaneous bag equations:

$$B \cup [2, 2, 3, 4] = [2, 2, 3, 3, 4, 4, 5]\,.$$
$$B \cap [2, 2, 3, 4, 5] = [2, 3, 4, 5]\,.$$

21. How would you define the difference operation for bags? Try to make your definition agree with the difference operation for sets whenever the bags are like sets (without repeated occurrences of elements).

Proofs and Challenges

22. Prove each of the following facts about the union operation (1.3). Use subset arguments that are written in complete sentences.

a. $A \cup \varnothing = A$.
b. $A \cup B = B \cup A$.
c. $A \cup A = A$.
d. $A \cup (B \cup C) = (A \cup B) \cup C$.

23. Prove each of the following facts about the intersection operation (1.6). Use subset arguments that are written in complete sentences.

a. $A \cap \varnothing = \varnothing$.
b. $A \cap B = B \cap A$.
c. $A \cap (B \cap C) = (A \cap B) \cap C$.
d. $A \cap A = A$.
e. $A \subset B$ if and only if $A \cap B = A$.

24. Prove that $\text{power}(A \cap B) = \text{power}(A) \cap \text{power}(B)$.

25. Prove that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

26. Prove each of the following absorption laws (1.9) twice. The first proof should use subset arguments. The second proof should use an already known result.

a. $A \cap (B \cup A) = A$.
b. $A \cup (B \cap A) = A$.

27. Show that $(A \cap B) \cup C = A \cap (B \cup C)$ if and only if $C \subset A$.

28. Give a proof or a counterexample for each of the following statements.

a. $A \cap (B \cup \mathrm{A}) = A \cap B$.
b. $A - (B \cap \mathrm{A}) = A - B$.
c. $A \cap (B \cup C) = (A \cup B) \cap (A \cup C)$.
d. $A \oplus A = A$.

29. Prove each of the following properties of the complement (1.10).

a. $(A')' = A$.
b. $\varnothing' = U$ and $U' = \varnothing$.
c. $A \cap A' = \varnothing$ and $A \cup A' = U$.
d. $(A \cup B)' = A' \cap B'$. (De Morgan's law)
e. $(A \cap B)' = A' \cup B'$. (De Morgan's law)
f. $A \cap (A' \cup B) = A \cap B$. (absorption law)
g. $A \cup (A' \cap B) = A \cup B$. (absorption law)

30. Can you find a description of a set A satisfying the equation $A = \{a, A, b\}$. Notice in this case that $A \in A$.

1.3 Ordered Structures

In the previous section we saw that sets and bags are used to represent unordered information. In this section we'll introduce some notions and notations for structures that have some kind of ordering to them.

1.3.1 Tuples

When we write down a sentence, it always has a sequential nature. For example, in the previous sentence the word "When" is the first word, the word "we" is the second word, and so on. Informally, a *tuple* is a collection of things, called its *elements,* where there is a first element, a second element, and so on. The elements of a tuple are also called *members*, *objects*, or *components.* We'll denote a tuple by writing down its elements, separated by commas, and surrounding everything with the two symbols "(" and ")". For example, the tuple (12, R, 9) has three elements. The first element is 12, the second element is the letter R, and third element is 9. The beginning sentence of this paragraph can be represented by the following tuple:

(When, we, write, down, . . . , sequential, nature).

If a tuple has n elements, we say that its *length* is n, and we call it an *n-tuple*. So the tuple (8, k, hello) is a 3-tuple, and $(x_1, \ldots, x_8)$ is an 8-tuple. The 0-tuple is denoted by ( ), and we call it the *empty* tuple. A 2-tuple is often called an *ordered pair*, and a 3-tuple might be called an *ordered triple*. Other words used in place of the word *tuple* are *vector* and *sequence*, possibly modified by the word *ordered*.

Two n-tuples $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are said to be *equal* if $x_i = y_i$ for $1 \leq i \leq n$, and we denote this by $(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$. Thus the ordered pairs (3, 7) and (7, 3) are not equal, and we write $(3, 7) \neq (7, 3)$. Since tuples convey the idea of order, they are different from sets and bags. Here are some examples:

Sets: $\{b, a, t\} = \{t, a, b\}$.
Bags: $[t, o, o, t] = [o, t, t, o]$.
Tuples: $(t, o, o, t) \neq (o, t, t, o)$ and $(b, a, t) \neq (t, a, b)$.

Here are the two important characteristics of tuples.

Two Characteristics of Tuples

1. There may be repeated occurrences of elements.
2. There is an order or arrangement of the elements.

The rest of this section introduces structures that are represented as tuples. We'll also see in the next section that graphs and trees are often represented using tuples.

Cartesian Product of Sets

We often need to represent information in the form of tuples, where the elements in each tuple come from known sets. Such a set is called a Cartesian product, in honor of René Descartes (1596–1650), who introduced the idea of graphing ordered pairs. The Cartesian product is also referred to as the *cross product*. Here's the formal definition.

Definition of Cartesian Product

If A and B are sets, then the *Cartesian product* of A and B, which is denoted by $A \times B$, is the set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$ In other words, we have

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}.$$

For example, if $A = \{x, y\}$ and $B = \{0, 1\}$, then

$$A \times B = \{(x, 0), (x, 1), (y, 0), (y, 1)\}.$$

Suppose we let $A = \varnothing$ and $B = \{0, 1\}$ and then ask the question "What is $A \times B$?". If we apply the definition of Cartesian product, we must conclude that there are no ordered pairs with first elements from the empty set. Therefore, $A \times B = \varnothing$. So it's easy to generalize and say that $A \times B$ is nonempty if and only if both A and B are nonempty sets. The Cartesian product of two sets is easily extended to any number of sets $A_1, \ldots, A_n$ by writing

$$A_1 \times \cdots \times A_n = \{(x_1, \ldots, x_n) \mid x_i \in A_i\}.$$

If all the sets A_i in a Cartesian product are the same set A, then we use the abbreviated notation $A^n = A \times \cdots \times A$. With this notation we have the following definitions for the sets A^1 and A^0:

$$A^1 = \{(a) \mid a \in A\} \text{ and } A^0 = \{(\,)\}.$$

So we must conclude that $A^1 \neq A$ and $A^0 \neq \varnothing$.

example

1.21 Some Products

Let $A = \{a, b, c\}$. Then we have the following Cartesian products:

$A^0 = \{()\}$,
$A^1 = \{(a), (b), (c)\}$,
$A^2 = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)\}$,
A^3 is bigger yet, with twenty-seven 3-tuples.

end example

When working with tuples, we need the ability to randomly access any component. The components of an n-tuple can be indexed in several different ways depending on the problem at hand. For example, if $t \in A \times B \times C$, then we might represent t in any of the following ways:

(t_1, t_2, t_3),
$(t(1), t(2), t(3))$,
$(t[1], t[2], t[3])$,
$(t(A), t(B), t(C))$,
$(A(t), B(t), C(t))$.

Let's look at an example that shows how Cartesian products and tuples are related to some familiar objects of programming.

example **1.22 Arrays, Matrices, and Records**

In computer science, a 1-dimensional array of size n with elements in the set A is an n-tuple in the Cartesian product A^n. So we can think of the Cartesian product A^n as the set of all 1-dimensional arrays of size n over A. If $x = (x_1, \ldots, x_n)$, then the component x_i is usually denoted—in programming languages—by $x[i]$.

A 2-dimensional array—also called a *matrix*—can be thought of as a table of objects that are indexed by rows and columns. If x is a matrix with m rows and n columns, we say that x is an m by n matrix. For example, if x is a 3 by 4 matrix, then x can be represented by the following diagram:

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{12} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix}.$$

We can also represent x as a 3-tuple whose components are 4-tuples as follows:

$$x = ((x_{11}, x_{12}, x_{13}, x_{14}), (x_{21}, x_{22}, x_{23}, x_{24}), (x_{31}, x_{32}, x_{33}, x_{34})).$$

In programming, the component x_{ij} is usually denoted by $x[i, j]$. We can think of the Cartesian product $(A^4)^3$ as the set of all 2-dimensional arrays over A with 3 rows and 4 columns. Of course, this idea extends to higher dimensions. For example, $((A^5)^7)^4$ represents the set of all 3-dimensional arrays over A consisting of 4-tuples whose components are 7-tuples whose components are 5-tuples of elements of A.

For another example we can think of $A \times B$ as the set of all records, or structures, with two fields A and B. For a record $r = (a, b) \in A \times B$ the components a and b are normally denoted by $r.A$ and $r.B$.

end example

There are at least three nice things about tuples: They are easy to understand; they are basic building blocks for the representation of information; and they are easily implemented by a computer, which we'll discuss in the next example.

example **1.23 Computer Repesentation of Tuples**

Computers represent tuples in contiguous cells of memory so that each component can be accessed quickly. For example, suppose that each component of the tuple $x = (x_1, \ldots, x_n)$ needs M memory cells to store it. If B is the beginning address of memory allocated for the tuple x, then x_1 is at location B, x_2 is at location $B + M$, and in general, x_k is at location

$$B + M(k - 1).$$

So each component x_k of x can be accessed in the amount of time that it takes to evaluate $B + M(k - 1)$.

For multidimensional arrays, the access time is also fast. For example, suppose x is a 3 by 4 matrix represented in the following "row-major" form as a three tuple of rows, where each row is a 4-tuple.

$$x = ((x_{11}, x_{12}, x_{13}, x_{14}), (x_{21}, x_{22}, x_{23}, x_{24}), (x_{31}, x_{32}, x_{33}, x_{34})).$$

Suppose that each component of x needs M memory cells. If B is the beginning address of memory allocated for x, then x_{11} is at location B, x_{21} is at location $B + 4M$, and x_{31} is at location $B + 8M$. The location of an arbitrary element x_{jk} is given by the expression

$$B + 4M(j - 1) + M(k - 1).$$

Expressions such as this are called *address polynomials*. Each component x_{jk} can be accessed in the amount of time that it takes to evaluate the address polynomial, which is close to a constant for any j and k.

end example

1.3.2 Lists

A *list* is a finite ordered sequence of zero or more elements that can be repeated. At this point a list seems just like a tuple. So what's the difference between tuples and lists? The difference—a big one in computer science—is in what parts can be randomly accessed. In the case of tuples we can randomly access any component in a constant amount of time. In the case of lists we can randomly access only two things in a constant amount of time: the first component of a list, which is called its *head*, and the list made up of everything except the first component, which is called its *tail*.

So we'll use a different notation for lists. We'll denote a list by writing down its elements, separated by commas, and surrounding everything with the two symbols "$\langle$ " and "$\rangle$". The *empty list* is denoted by

$$\langle\ \rangle.$$

The number of elements in a list is called its *length*. For example, the list

$$\langle\ w, x, y, z\rangle,$$

has length 4, its head is w, and its tail is the list $\langle\ x, y, z\rangle$. If L is a list, we'll use the notation

$$\text{head}(L) \text{ and tail } (L)$$

to denote the head of L and the tail of L. For example,

$$\text{head}(\langle\ w, x, y, z\rangle) = w,$$
$$\text{tail}(\langle\ w, x, y, z\rangle\ ) = \langle\ x, y, z\rangle.$$

Notice that the empty list $\langle\ \rangle$ does not have a head or tail.

An important computational property of lists is the ability to easily construct a new list by adding a new element at the head of an existing list. The name *cons* will be used to denote this construction operation. If h is an element of some kind and L is a list, then

$$\text{cons}(h,\ L)$$

dentes the list whose head is h and whose tail is L. Here are a few examples:

$$\text{cons}(w,\ \langle\ x,\ y,\ z\rangle) = \langle\ w,\ x,\ y,\ z\rangle,$$
$$\text{cons}(a,\ \langle\ \rangle) = \langle a\rangle.$$
$$\text{cons}(\text{this},\ \langle\text{is},\ \text{helpful}\rangle) = \langle\text{this},\ \text{is},\ \text{helpful}\rangle.$$

The operations head, tail, and cons can be done efficiently and dynamically during the execution of a program. The three operations are related by the following equation for any nonempty list L.

$$\text{cons}(\text{head}(L),\ \text{tail}(L)) = L.$$

There is no restriction on the kind object that a list can contain. In fact, it is often quite useful to represent information in the form of lists and the elements of those lists may be lists, whose elements may be lists, and so on. Here are a few examples of such lists, together with their heads and tails.

L	head(L)	tail(L)
$\langle a, \langle b\rangle\rangle$	a	$\langle\langle b\rangle\rangle$
$\langle\langle a\rangle, \langle b, a\rangle\rangle$	$\langle a\rangle$	$\langle\langle b, a\rangle\rangle$
$\langle\langle\langle\ \rangle, a, \langle\ \rangle\rangle, b, \langle\ \rangle\rangle$	$\langle\langle\ \rangle, a, \langle\ \rangle\rangle$	$\langle b, \langle\ \rangle\rangle$

If all the elements of a list L are from a particular set A, then L is said to be a *list over* A. For example, each of the following lists is a list over $\{a,\ b,\ c\}$.

$$\langle\ \rangle,\ \langle a\rangle,\ \langle a,\ b\rangle,\ \langle b,\ a\rangle,\ \langle b,\ c,\ a,\ b,\ c\rangle.$$

We'll denote the collection of all lists over A by

$$\text{lists}(A).$$

There are at least four nice things about lists: They are easy to understand; they are basic building blocks for the representation of information; they are easily implemented by a computer; and they are easily manipulated by a computer program. We'll discuss this in the next example.

example 1.24 Computer Repesentation of Lists

A simple way to represent a list in a computer is to allocate a block of memory for each element of the list. The block of memory contains the element together

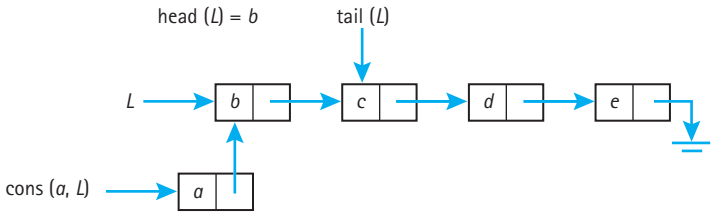


Figure 1.10 Memory representation of a list.

with an address (called a pointer or link) to the next block of memory for the next element of the list. In this way there is no need to have list elements next to each other in memory, so that the creation and deletion of list elements can occur dynamically during the execution of a program.

For example, let's consider the list $L = \langle b, c, d, e \rangle$. Figure 1.10 shows the memory representation of L in which each arrow represents an address (i.e., a pointer or link) and each box represents a block of memory containing an element of the list and an arrow pointing to the address of the next box. The last arrow in the box for e points to the "ground" symbol to signify the end of the list. Empty lists point to the ground symbol, too. The figure also shows $\text{head}(L) = b$ and $\text{tail}(L) = \langle c, d, e \rangle$. So head and tail are easily calculated from L. The figure also shows how the cons operation constructs a new list $\text{cons}(a, L) = \langle a, b, c, d, e \rangle$ by allocating a new block of memory to contain a and a pointer to L.

end example

1.3.3 Strings and Languages

A *string* is a finite ordered sequence of zero or more elements that are placed next to each other in juxtaposition. The individual elements that make up a string are taken from a finite set called an *alphabet*. If A is an alphabet, then a string of elements from A is said to be a string over A. For example, here are a few strings over $\{a, b, c\}$:

$$a,\ ba,\ bba,\ aacabb.$$

The string with no elements is called the *empty string*, and we denote it by the Greek capital letter lambda:

$$\Lambda.$$

The number of elements that occur in a string s is called the *length* of s, which we sometimes denote by

$$|s|.$$

For example, $|\Lambda| = 0$ and $|aacabb| = 6$ over the alphabet $\{a, b, c\}$.

Concatenation of Strings

The operation of placing two strings s and t next to each other to form a new string st is called *concatenation*. For example, if aab and ba are two strings over the alphabet $\{a, b\}$, then the concatenation of aab and ba is the string

$$aabba.$$

We should note that if the empty string occurs as part of another string, then it does not contribute anything new to the string. In other words, if s is a string, then

$$s\Lambda = \Lambda s = s.$$

Strings are used in the world of written communication to represent information: computer programs; written text in all the languages of the world; and formal notation for logic, mathematics, and the sciences.

There is a strong association between strings and lists because both are defined as finite sequences of elements. This association is important in computer science because computer programs must be able to recognize certain kinds of strings. This means that a string must be decomposed into its individual elements, which can then be represented by a list. For example, the string $aacabb$ can be represented by the list $\langle a, a, c, a, b, b\rangle$. Similarly, the empty string Λ can be represented by the empty list $\langle\ \rangle$.

Languages (Sets of Strings)

A *language* is a set of strings. If A is an alphabet, then a *language* over A is a set of strings over A. The set of all strings over A is denoted by A^*. So any language over A is a subset of A^*. Four simple examples of languages over an alphabet A are the sets $\varnothing$, $\{\Lambda\}$, A, and A^*.

For example, if $A = \{a\}$, then these four simple languages over A become

$$\varnothing\ , \{\Lambda\}, \{a\}, \text{and } \{\Lambda\ , a, aa, aaa, \ldots\}.$$

For any natural number n the concatenation of a string s with itself n times is denoted by s^n. For example,

$$s^0 = \Lambda\ , s^1 = s, s^2 = ss, \text{and } s^3 = sss.$$

example 1.25 Language Representations

The exponent notation allows us to represent some languages in a nice concise manner. Here are a few examples of languages.

$\{a^n \mid n \in \mathbb{N}\} = \{\Lambda\ , a, aa, aaa, \ldots\}.$
$\{ab^n \mid n \in \mathbb{N}\} = \{a, ab, abb, abbb, \ldots\}$

$$\{a^n b^n \mid n \in \mathbb{N}\} = \{\Lambda\ ,\ ab,\ aabb,\ aaabbb,\ \dots\}$$
$$\{(ab)^n \mid n \in \mathbb{N}\} = \{\Lambda\ ,\ ab,\ abab,\ ababab,\ \dots\}.$$

end example

example

1.26 Numerals

A *numeral* is a written number. In terms of strings, we can say that a numeral is a nonempty string of symbols that represents a number. Most of us are familiar with the following three numeral systems. The *Roman numerals* represent the set of nonnegative integers by using the alphabet

$$\{\text{I, V, X, L, C, D, M}\}.$$

The *decimal numerals* represent the set of natural numbers by using the alphabet

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}.$$

The *binary numerals* represent the natural numbers by using the alphabet

$$\{0, 1\}.$$

For example, the Roman numeral MDCLXVI, the decimal numeral 1666, and the binary numeral 11010000010 all represent the same number.

end example

Products of Languages

Since languages are sets of strings, they can be combined by the usual set operations of union, intersection, difference, and complement. But there is another important way to combine languages.

We can combine two languages L and M to obtain the set of all concatenations of strings in L with strings in M. This new language is called the *product* of L and M and is denoted by LM. Here's a formal definition.

Product of Languages
The product of languages L and M is the language

$$LM = \{st \mid s \in L \text{ and } t \in M\}.$$

For example, if $L = \{ab,\ ac\}$ and $M = \{a,\ bc,\ abc\}$, then the product LM is the language

$$LM = \{aba,\ abbc,\ ababc,\ aca,\ acbc,\ acabc\}.$$

It's easy to see, from the definition of product, that the following simple properties hold for any language L.

$$L\{\Lambda\} = \{\Lambda\}L = L.$$
$$L\varnothing = \varnothing L = \varnothing.$$

It's also easy to see that the product is associative. In other words, if L, M, and N are languages, then $L(MN) = (LM)N$. Thus we can write down products without using parentheses. On the other hand, it's easy to see that the product is not commutative. In other words, we can find two languages L and M such that $LM \neq ML$.

For any natural number n the product of a language L with itself n times is denoted by L^n. In other words, we have

$$L^n = \{s_1 s_2 \dots s_n \mid s_k \in L \text{ for each } k\}.$$

The special case when $n = 0$ has the following definition.

$$L^0 = \{\Lambda\ \}.$$

example

1.27 Some Language Products

We'll calculate some products for the langauge $L = \{a,\ bb\}$.

$L^0 = \{\Lambda\ \}$,
$L^1 = L = \{a,\ bb\}$,
$L^2 = LL = \{aa,\ abb,\ bba,\ bbbb\}$,
$L^3 = LL^2 = \{aaa,\ aabb,\ abba,\ abbbb,\ bbaa,\ bbabb,\ bbbba,\ bbbbbb\}$.

end example

Closure of a Language

If L is a language, then the *closure* of L, denoted by L^*, is the set of all possible concatenations of strings from L. In other words, we have

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots \cup L^n \cup \cdots.$$

So $x \in L^*$if and only if $x \in L^n$ for some n. Therefore, we have

$$x \in L^* \text{ if and only if either } x = \Lambda \text{ or } x = l_1 l_2 \dots l_n$$

for some $n \geq 1$, where $l_k \in L$ for $1 \leq k \leq n$.

If L is a language, then the *positive closure* of L, which is denoted by L^+, is defined by

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \cdots.$$

It follows from the definition that $L^* = L^+ \cup \{\Lambda\}$. But it's not necessarily true that $L^+ = L^* - \{\Lambda\}$. For example, if $L = \{\Lambda, a\}$, then $L^+ = L^*$.

We should observe that any alphabet A is itself a language and its closure coincides with our original definition of A^* as the set of all strings over A. The following properties give some basic facts about the closure of languages.

Properties of Closure (1.16)

a. $\{\Lambda\}^* = \varnothing^* = \{\Lambda\}$.

b. $\Lambda \in L$ if and only if $L^+ = L^*$.

c. $L^* = L^*L^* = (L^*)^*$.

d. $(L^*M^*)^* = (L^* \cup M^*)^* = (L \cup M)^*$.

e. $L\,(M\ L)^* = (LM)^*L$.

Proof: We'll prove part (e) and leave the others as exercises. We'll start by examining the structure of an arbitrary string $x \in L(ML)^*$. Since $L(ML)^*$ is the product of L and $(ML)^*$, we can write $x = ly$, where $l \in L$ and $y \in (ML)^*$. Since $y \in (ML)^*$, it follows that $y \in (ML)^n$ for some n. If $n = 0$, then $y = \Lambda$ and we have $x = ly = l\Lambda = l \in L$.

If $n > 0$, then $y = w_1 \ldots w_n$, where $w_k \in ML$ for $1 = k = n$. So we can write each w_k in the form $w_k = m_k l_k$, where $m_k \in M$ and $l_k \in L$. Now we can collect our facts and write x as a concatenation of strings from L and M.

$$\begin{aligned} x &= ly && \text{where } l \in L \text{ and } y \in (ML)* \\ &= l(w_1 \ldots w_n) && \text{where } l \in L \text{ and each } w_k \in (ML) \\ &= l(m_1 l_1 \ldots m_n l_n) && \text{where } l \in L \text{ and each } l_k \in L, \text{ and } m_k \in M. \end{aligned}$$

Since we can group strings with parentheses any way we want, we can put things back together in the following order.

$$\begin{aligned} &= (lm_1 l_1 \ldots m_n) l_n && \text{where } l \in L \text{ and each } l_k \in L, \text{ and } m_k \in M \\ &= (z_1 \ldots z_n) l_n && \text{where each } z_k \in LM \text{ and } l_n \in L \\ &= u l_n && \text{where } u \in (LM)* \text{ and } l_n \in L. \end{aligned}$$

So $x \in (LM)^*L$. Therefore, $L(ML)^* \subset (LM)^*L$. The argument is reversible. So we have $L(ML)^* = (L\ M)^*L$. QED.

example **1.28 Decimal Numerals**

The product is a useful tool for describing languages in terms of simpler languages. For example, suppose we need to describe the language L of all strings

of the form a . b, where a and b are decimal numerals. For example, the strings 0.45, 1.569, 000.34000 are elements of L. If we let

$$D = \{.\} \text{ and } N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\},$$

Then we can describe L as the following product in terms of D and N.

$$L = N(N)^* \, DN(N)^*.$$

end example

1.3.4 Relations

Definition of Relation

Ideas such as kinship, connection, and association of objects are keys to the concept of a relation. Informally, a *relation* is a set of n-tuples, where the elements in each tuple are related in some way.

For example, the parent-child relation can be described as the following set of ordered pairs:

$$\text{isParentOf} = \{(x, y) \mid x \text{ is a parent of } y\}.$$

For another example, recall from geometry that if the sides of a right triangle have lengths x, y, and z, where z is the hypotenuse, then $x^2 + y^2 = z^2$. Any 3-tuple of positive real numbers (x, y, z) with this propoerty is called a *Pythagorean triple*. For example, $(1, \sqrt{3}, 2)$ and $(3, 4, 5)$ are Pythagorean triples. The Pythagorean triple relation can be described as the following set of ordered triples:

$$PT = \{(x, y, z) \mid x^2 + y^2 = z^2\}.$$

When we discuss relations in terms of where the tuples come from, there is some terminology that can be helpful. Here it is.

Definition of Relation

If R is a subset of $A_1 \times \cdots \times A_n$, then R is said to be *an n-ary relation on* (or *over*) $A_1 \times \cdots \times A_n$. If R is a subset of A^n, then we say R is *an n-ary relation on A*. Instead of 1-ary, 2-ary, and 3-ary, we say *unary*, *binary*, and *ternary*.

For example, the isParentOf relation is a binary relation on the set of people and the Pythagorean triple relation is a ternary operation on the set of positive real numbers. In formal terms, if P is the set of all people who are living or who have ever lived, then

$$\text{isParentOf} \subset P \times P.$$

Similarly, if we let $\mathbb{R}^+$ denote the set of positive real numbers, then

$$PT \subset \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+.$$

Since there are many subsets of a set, there can be many relations. The smallest relation is the empty set $\varnothing$, which is called the *empty relation*. The largest relation is $A_1 \times \cdots \times A_n$ itself, which is called the *universal relation*.

If R is a relation and $(x_1, \ldots, x_n) \in R$, this fact is often denoted by the *prefix expression*

$$R(x_1, \ldots, x_n).$$

For example, $PT(1, \sqrt{3}, 2)$ means that $(1, \sqrt{3}, 2) \in PT$.

If R is a binary relation, then the statement $(x, y) \in R$ can be denoted by $R(x, y)$, but it is often denoted by the *infix expression*

$$x \; R \; y.$$

For example, "John isParentOf Mary" means that (John, Mary) $\in$ isParentOf.

We use many binary relations without even thinking about it. For example, we use the less-than relation on numbers m and n by writing $m < n$ instead of $(m, n) \in <$ or $< (m, n)$. We also use equality without thinking about it as a binary relation. The *equality relation* on a set A is the set

$$\{(x, x) \mid x \in A\}.$$

For example, if $A = \{a, b, c\}$, then the equality relation on A is the set $\{(a, a), (b, b), (c, c)\}$. We normally denote equality by the symbol $=$ and we write $a = a$ instead of $(a, a) \in =$ or $= (a, a)$.

Since unary relations are sets of 1-tuples, we usually dispense with the tuple notation and simply write a unary relation as a set of elements. For example, instead of writing $R = \{(2), (3), (5), (7)\}$ we write $R = \{2, 3, 5, 7\}$. So $R(2)$ and $2 \in R$ mean the same thing.

Relational Databases

A *relational database* is a collection of facts that are represented by tuples in such a way that the tuples can be accessed in various ways to answer queries about the facts. To accomplish these tasks each component of a tuple must have an associated name, called an *attribute*.

For example, suppose we have a database called Borders that describes the foreign countries and large bodies of water that border each state of the United States. The following table represents a sample of the information in the database with attribute names State, Foreign, and Water.

Borders

State	Foreign	Water
Washington	Canada	Pacific Ocean
Minnesota	Canada	Lake Superior
Wisconsin	None	Lake Michigan
Oregon	None	Pacific Ocean
Maine	Canada	Atlantic Ocean
Michigan	Canada	Lake Superior
Michigan	Canada	Lake Huron
Michigan	Canada	Lake Michigan
California	Mexico	Pacific Ocean
Arizona	Mexico	None

There is no special order to the rows of a relational database. So the table can be represented as a set of tuples.

$$\begin{aligned}\text{Borders} \quad = \quad \{&(\text{Washington, Canada, Pacific Ocean}),\\ &(\text{Minnesota, Canada, Lake Superier}),\\ &(\text{Wisconsin, None, Lake Michigan}),\ldots\ \}.\end{aligned}$$

example

1.29 Questions about Borders

Let's look at a few questions or queries that can be asked about the Borders database. Each question can be answered by describing a set or a relation. For example, suppose we ask the question

What states border Mexico?

The answer is the set

$$\{x \mid (x, \text{Mexico}, z) \in \text{Borders, for some } z\}.$$

Here are a few more questions that we'll leave to the exercises.

What bodies of water border Michigan?
What states border the Pacific Ocean?
What states are landlocked?
What relation represents the state-water pairs?
What state-water pairs contain a state bordering Canada?

end example

Queries can be answered not only by describing a set or a relation as in the example, but also by describing an expression in terms of basic operations that construct new relations by selecting certain tuples, by eliminating certain attributes, or by combining attributes of two relations. We'll discuss these basic operations on relational databases in Section 10.4.

1.3.5 Counting Tuples

How can we count a set of tuples, lists, or strings? Since tuples, lists, and strings represent finite ordered sequences of objects, the only difference is how we represent them, not whether there are more of one kind than another. For example, over the set $\{a, b\}$ there are eight 3-tuples, eight lists of length 3, and eight strings of length 3. So without any loss of generality we'll discuss counting sets of tuples. The main tools that we'll use are the rules for counting Cartesian products of finite sets.

The Product Rule

Suppose we need to know the cardinality of $A \times B$ for finite sets A and B. In other words, we want to know how many 2-tuples are in $A \times B$. For example, suppose that $A = \{a, b, c\}$ and $B = \{0, 1, 2, 3\}$. The sets A and B are small enough so that we can write down all 12 of the tuples. The exercise might also help us notice that each element of A can be paired with any one of the four elements in B. Since there are three elements in A, it follows that

$$|A \times B| = (3)(4) = 12.$$

This is an example of a general counting technique called the product rule, which we'll state as follows for any two finite sets A and B.

Product Rule (1.17)

$$|A \times B| = |A||B|.$$

It's easy to see that (1.17) generalizes to a Cartesian product of three or more finite sets. For example, $A \times B \times C$ and $A \times (B \times C)$ are not actually equal because an arbitrary element in $A \times \mathrm{B} \times C$ is a 3-tuple (a, b, c), while an arbitrary element in $A \times (B \times C)$ is a 2-tuple $(a, (b, c))$. Still the two sets have the same cardinality. Can you convince yourself of this fact? Now proceed as follows:

$$\begin{aligned} |A \times B \times C| &= |A \times (B \times C)| \\ &= |A|\,|B \times C| \\ &= |A|\,|B|\,|C|\,. \end{aligned}$$

The extension of (1.17) to any number of sets allows us to obtain other useful formulas for counting tuples of things. For example, for any finite set A and any natural number n we have the following product rule.

$$|A^n| = |A|^n\,. \tag{1.18}$$

Counting Strings as Tuples

We can use product rules to count strings as well as tuples because a string can be represented as a tuple. In each of the following examples the problem to be solved is expressed in terms of strings.

example **1.30 Counting All Strings**

Suppose we need to count the number of strings of length 5 over the alphabet $A = \{a, b, c\}$. Any string of length 5 can be considered as a 5-tuple. For example, the string *abcbc* can be represented by the tuple (a, b, c, b, c). So the number of strings of length 5 over A equals the number of 5-tuples over A, which by product rule (1.18) is

$$|A^5| = |A|^5 = 3^5 = 243.$$

end example

example **1.31 Strings with Restrictions**

Suppose we need to count the number of strings of length 6 over the alphabet $A = \{a, b, c, d\}$ that begin with either a or c and contain at least one occurrence of b.

Since strings can be represented by tuples, we'll count the number of 6-tuples over A that begin with either a or c and contain at least one occurrence of b. We'll break up the problem into two simpler problems. First, let U be the set of 6-tuples over A that begin with a or c. In other words, $U = \{a, c\} \times A^5$. Next, let S be the subset of U consisting of those 6-tuples that do not contain any occurrences of b. In other words, $S = \{a, c\} \times \{a, c, d\}^5$. Then the set $U - S$ is the desired set of 6-tuples over A that begin with either a or c and contain at least one occurrence of b. So we have

$$\begin{aligned} |U - S| &= |U| - |S| && (1.14) \\ &= |\{a, c\} \times A^5| - |\{a, c\} \times \{a, c, d\}^5| \\ &= |\{a, c\}||A|^5 - |\{a, c\}||\{a, c, d\}^5 && (1.17) \text{ and } (1.18) \\ &= 2(4^5) - 2(3^5) \\ &= 1,562. \end{aligned}$$

end example

example **1.32 Strings with More Restrictions**

We'll count the number of strings of length 6 over $A = \{a, b, c, d\}$ that start with a or c and contain at least one occurrence of either b or d.

As in the previous example, let U be the set of 6-tuples over A that start with a or c. Then $U = \{a, c\} \times A^5$ and $|U| = 2(4^5)$. Now let S be the subset

of U whose 6-tuples do not contain any occurrences of b and do not contain any occurrences of d. So $S = \{a, c\}^6$ and $|S| = 2^6$. Then the set $U - S$ is the desired set of 6-tuples over A that begin with either a or c and contain at least one occurrence of either b or d. So by (1.14) we have

$$\begin{aligned} |U - S| &= |U| - |S| \\ &= 2(4^5) - 2^6 \\ &= 1,984. \end{aligned}$$

end example

example **1.33 Strings with More Restrictions**

Suppose we need to count the number of strings of length 6 over $A = \{a, b, c, d\}$ that start with a or c and contain at least one occurrence of b and at least one occurrence of d.

In this case we'll break up the problem into three simpler problems. First, let U be the set of 6-tuples that start with a or c. So $U = \{a, c\} \times A^5$ and $|U| = 2(4^5)$. Next, let S be the subset of U whose 6-tuples don't contain b. So $S = \{a, c\} \times \{a, c, d\}^5$ and $|S| = 2(3^5)$. Similarly, let T be the subset of U whose 6-tuples don't contain d. So $T = \{a, c\} \times \{a, b, c\}^5$ and $|T| = 2(3^5)$. Then the set $U - (S \cup T)$ is the desired set of 6-tuples that start with a or c and contain at least one occurrence of b and at least one occurrence of d. The cardinality of this set has the form

$$\begin{aligned} |U - (S \cup T)| &= |U| - |S \cup T| && \text{(by 1.14)} \\ &= |U| - (|S| + |T| - |S \cap T|)\,. && \text{(by 1.11)} \end{aligned}$$

We'll be done if we can calculate the cardinality of $S \cap T$. Notice that

$$\begin{aligned} S \cap T &= \{a, c\} \times \{a, c, d\}^5 \cap \{a, c\} \times \{a, b, c\}^5 \\ &= \{a, c\} \times \{a, c\}^5 \\ &= \{a, c\}^6\,. \end{aligned}$$

So $|S \cap T| = 2^6$. Now we can complete the calculation of $|U - (S \cup T)|$.

$$\begin{aligned} |U - (S \cup T)| &= |U| - (|S| + |T| - |S \cap T|) \\ &= 2\left(4^5\right) - \left[2\left(3^5\right) + 2\left(3^5\right) - 2^6\right] \\ &= 1,140. \end{aligned}$$

end example

Exercises

Tuples

1. Write down all possible 3-tuples over the set $\{x, y\}$.

2. Let $A = \{a, b, c\}$ and $B = \{a, b\}$. Compute each of the following sets.

a. $A \times B$. b. $B \times A$. **c.** A^0.
d. A^1. **e.** A^2. f. $A^2 \cap (A \times B)$.

Lists

3. Write down all possible lists of length 2 or less over the set $A = \{a, b\}$.

4. Find the head and tail of each list.

a. $\langle a \rangle$. b. $\langle a, b, c \rangle$. **c.** $\langle \langle a, b \rangle, c \rangle$.
d. $\langle \langle a, b \rangle, \langle a, c \rangle \rangle$.

5. For positive integers m and n let D be the list of integers greater than 1 that divide both m and n, where D is ordered from smallest to largest. For example, if $m = 12$ and $n = 18$, then $D = \langle 2, 3, 6 \rangle$. We'll combine this information into a list $\langle m, n, D \rangle = \langle 12, 18, \langle 2, 3, 6 \rangle \rangle$. Construct $\langle m, n, D \rangle$ for each of the following cases.

a. $m = 24$ and $n = 60$.
b. $m = 36$ and $n = 36$.
c. $m = 14$ and $n = 15$.

6. Write down all possible lists over $\{a, b\}$ that can be represented with five symbols, where the symbols that we count are a or b or $\langle$or $\rangle$. For example, $\langle a, \langle\rangle\rangle$ uses five of the symbols. Can you do the same for lists over A that have six symbols? There are quite a few of them.

Strings and Languages

7. Write down all possible strings of length 2 over the set $A = \{a, b, c\}$.

8. Let $L = \{\Lambda , abb, b\}$ and $M = \{bba, ab, a\}$. Evaluate each of the following language expressions.

a. LM. b. ML. **c.** L^0. d. L^1. **e.** L^2.

9. Use your wits to solve each of the following language equations for the unknown language.

a. $\{\Lambda, a, ab\}\, L = \{b, ab, ba, aba, abb, abba\}$.
b. $L\,\{a, b\} = \{a, baa, b, bab\}$.
c. $\{a, aa, ab\}\, L = \{ab, aab, abb, aa, aaa, aba\}$.
d. $L\,\{\Lambda, a\} = \{\Lambda, a, b, ab, ba, aba\}$.
e. $\{a, b\}\, L = \{a, b, aba, bba\}$.
f. $L\,\{b, \Lambda, ab\} = \{abb, ab, abab, bab, b, bb\}$.

10. Let L and M be two languages. For each of the following languages describe the general structure of a string x by writing it as a concatenation of strings that are in either L or M.

a. LML.
b. LM^*.
c. $(L \cup M)^*$.
d. $(L \cap M)^*$.
e. L^*M^*.
f. $(LM)^*$.

11. Try to describe each of the following languages in some way.

a. $\{a, b\}^* \cap \{b, c\}^*$
b. $\{a, b\}^* - \{b\}^*$.
c. $\{a, b, c\}^* - \{a, b\}^*$.

Relations

12. Represent each relation as a set by listing each individual tuple.

a. $\{(d, 12) \mid d > 0$ and d divides $12\}$.
b. $\{(d, n) \mid d, n \in \{2, 3, 4, 5, 6\}$ and d divides $n\}$.
c. $\{(x, y, z) \mid x = y + z$, where $x, y, z \in \{1, 2, 3\}\}$.
d. Let $(x, y) \in S$ if and only if $x = y$ and $x, y \in \{1, 2, 3\}$.
e. Let $(x, y) \in U$ if and only if $x \in \{a, b\}$ and $y \in \{1, 2\}$.

13. Each of the following database queries refers to the Borders relational database given prior to Example 9. Express each answer by defining a set or relation in the same manner as the answer given in Example 9.

a. What bodies of water border Michigan?
b. What states border the Pacific Ocean?
c. What states are landlocked?
d. What relation represents the state-water pairs?
e. What state-water pairs contain a state bordering Canada?

Counting Tuples

14. For each of the following cases, find the number of strings over the alphabet $\{a, b, c, d, e\}$ that satisfy the given conditions.

a. Length 4, begins with a or b, contains at least one c.
b. Length 5, begins with a, ends with b, contains at least one c or d.
c. Length 6, begins with d, ends with b or d, contains no c's.
d. Length 6, contains at least one a and at least one b.

15. Find a formula for the number of strings of length n over an alphabet A such that each string contains at least one occurrence of a letter from a subset B of A. Express the answer in terms of $|A|$ and $|B|$.

Proofs and Challenges

16. Prove each of the following statements about combining set operations with Cartesian product.

a. $(A \cup B) \times C = (A \times C) \cup (B \times C)$.
b. $(A - B) \times C = (A \times C) - (B \times C)$.
c. Find and prove a similar equality using the intersection operation.

17. Let L, M, and N be languages. Prove each of the following properties of the product operation on languages.

a. $L\{\Lambda\} = \{\Lambda\}L = L$.
b. $L\varnothing = \varnothing L = \varnothing$.
c. $L(M \cup N) = LM \cup LN$ and $(M \cup N)L = ML \cup NL$.
d. $L(M \cap N) \subset LM \cap LN$ and $(M \cap N)L \subset ML \cap NL$.

18. Let L and M be languages. Prove each of the following statements about the closure of languages (1.16).

a. $\{\Lambda\}^* = \varnothing^* = \{\Lambda\}$.
b. $L^* = L^*L^* = (L^*)^*$.
c. $\Lambda \in L$ if and only if $L^+ = L^*$.
d. $(L^*M^*)^* = (L^* \cup M^*) = (L \cup M)^*$.

19. (*Tuples Are Special Sets*). We can define the concept of tuples in terms of sets. For example, we'll define

$$(\,) = \varnothing\,,\ (x) = \{x\}, \quad \text{and} \quad (x, y) = \{\{x\}, \{x, y\}\}.$$

Use this definition to verify each of the following statements.

a. Show that $(3, 7) \neq (7, 3)$.
b. Show that $(x, y) = (u, v)$ if and only if $x = u$ and $y = v$.
c. Find an example to show that the definition $(x, y) = \{x, \{y\}\}$ will not distinguish between distinct 2-tuples.

20. (*Tuples Are Special Sets*). Continuing with Exercise 19, we can define a 3-tuple in terms of sets by letting S be the set representing the ordered pair (x, y) from Exercise 19. Then define

$$(x, y, z) = \{\{S\}, \{S, z\}\}.$$

a. Write down the complete set to represent (x, y, z).

b. Show that $(a, b, c) = (d, e, f)$ if and only if $a = d$, $b = e$, and $c = f$.

c. Find an example to show that the definition

$$(x, y, z) = \{\{x\}, \{x, y\}, \{x, y, z\}\}$$

will not distinguish between distinct 3-tuples.

Note: We could continue in this manner and define n-tuples as sets for any natural number n. Although defining a tuple as a set is not at all intuitive, it does illustrate how sets can be used as a foundation from which to build objects and ideas. It also shows why good notation is so important for communicating ideas.

21. Use Example 3 as a guide to find the address polynomial for an arbitrary element in each of the following cases. Assume that all indexes start with 1, that the beginning address is B, and each component needs M memory cells.

a. A matrix of size 3 by 4 stored in column-major form as a 4-tuple of columns, each of which is a 3-tuple.

b. A matrix of size m by n stored in row-major form.

c. A three-dimensional array of size l by m by n stored as an l-tuple where each component is an m by n matrix stored in row-major form.

1.4 Graphs and Trees

When we think about graphs we might think about pictures of some kind that are used to represent information. The graphs that we'll discuss can be thought about in the same way. But we need to describe them in a little more detail if they are to be much use to us. We'll also see that trees are special kinds of graphs.

1.4.1 Definition of a Graph

Informally, a *graph* is a set of objects in which some of the objects are connected to each other in some way. The objects are called *vertices* or *nodes*, and the

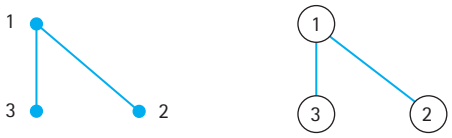


Figure 1.11 Graphs.

connections are called *edges*. For example, the United States can be represented by a graph where the vertices are states and the edges are the common borders between adjacent states. In this case, Hawaii and Alaska would be vertices without any edges connected to them. We say that two vertices are *adjacent* if there is an edge connecting them.

Picturing a Graph

We can picture a graph in several ways. For example, Figure 1.11 shows two ways to represent the graph with vertices 1, 2, and 3 and edges connecting 1 to 2 and 1 to 3.

example

1.34 States and Provinces

Figure 1.12 represents a graph of those states in the United States and those provinces in Canada that touch the Pacific Ocean or that touch states and provinces that touch the Pacific Ocean.

end example

Coloring a Graph

An interesting problem dealing with maps is to try to color a map with the fewest number of colors subject to the restriction that any two adjacent areas must have distinct colors. From a graph point of view, this means that any two distinct adjacent vertices must have different colors. Before reading any further, try to color the graph in Figure 1.12 with the fewest colors. It's usually easier to represent the colors by numbers like 1, 2, ...

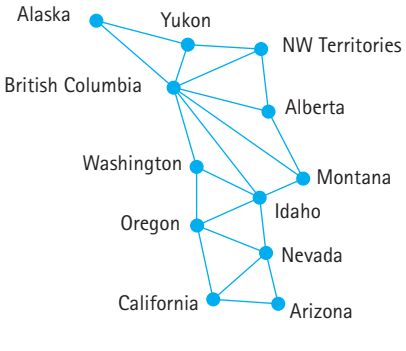


Figure 1.12 Graph of states and provinces.

A graph is *n-colorable* if there is an assignment of n colors to its vertices such that any two distinct adjacent vertices have distinct colors. The *chromatic number* of a graph is the smallest n for which it is n-colorable. For example, the chromatic number of the graph in Figure 1.12 is 3. A graph whose edges are the connections between all pairs of distinct vertices is called a *complete graph*. It's easy to see that the chromatic number of a complete graph with n vertices is n.

A graph is *planar* if it can be drawn on a plane such that no edges intersect. For example, the graph in Figure 1.12 is planar. A complete graph with four vertices is planar, but a complete graph with five vertices is not planar. See whether you can convince yourself of these facts. A fundamental result on graph coloring—that remained an unproven conjecture for over 100 years—states that *every planar graph is 4-colorable*. The result was proven in 1976 by Kenneth Appel and Wolfgang Haken. They used a computer to test over 1900 special cases. For example, see Appel and Haken [1976, 1977].

More Terminology

A *directed graph* (*digraph* for short) is a graph where each edge points in one direction. For example, the vertices could be cities and the edges could be the one-way air routes between them. For digraphs we use arrows to denote the edges. For example, Figure 1.13 shows two ways to represent the digraph with three vertices a, b, and c and edges from a to b, c to a, and c to b.

The *degree* of a vertex is the number of edges that it touches. However, we add two to the degree of a vertex if it has a *loop*, which is an edge that starts and ends at the same vertex. For directed graphs the *indegree* of a vertex is the number of edges pointing at the vertex, whereas the *outdegree* of a vertex is the number of edges pointing away from the vertex. In a digraph a vertex is called a *source* if its indegree is zero and a *sink* if its outdegree is zero. For example, in the digraph of Figure 1.13, c is a source and b is a sink.

If a graph has more than one edge between some pair of vertices, the graph is called a *multigraph*, or a *directed multigraph* in case the edges point in the same direction. For example, there are usually two or more road routes between most cities. So a graph representing road routes between a set of cites is most likely a multigraph.

Figure 1.13 Directed graphs.

Representaions of Graphs

From a computational point of view, we need to represent graphs as data. This is easy to do because we can define a graph in terms of tuples, sets, and bags. For example, we can define a graph G as an ordered pair (V, E), where V is a set of vertices and E is a set or bag of edges. If G is a digraph, then the edges in E can be represented by ordered pairs, where (a, b) represents the edge with an arrow from a to b. In this case the set E of edges is a subset of $V \times V$. In other words, E is a binary relation on V. For example, the digraph in Figure 1.13 has vertex set $\{a, b, c\}$ and edge set

$$\{(a, b), (c, b), (c, a)\}.$$

If G is a directed multigraph, then we can represent the edges as a bag (or multiset) of ordered pairs. For example, the bag $[(a, b), (a, b), (b, a)]$ represents three edges: two from a to b and one from b to a.

If a graph is not directed, we have more ways to represent the edges. We could still represent an edge as an ordered pair (a, b) and agree that it represents an undirected line between a and b. But we can also represent an edge between vertices a and b by a set $\{a, b\}$. For example, the graph in Figure 1.11 has vertex set $\{1, 2, 3\}$ and edge set $\{\{1, 2\}, \{1, 3\}\}$.

Weighted Graphs

We often encounter graphs that have information attached to each edge. For example, a good road map places distances along the roads between major intersections. A graph is called *weighted* if each edge is assigned a number, called a *weight*. We can represent an edge (a, b) that has weight w by the 3-tuple

$$(a, b, w).$$

In some cases we might want to represent an unweighted graph as a weighted graph. For example, if we have a multigraph in which we wish to distinguish between multiple edges that occur between two vertices, then we can assign a different weight to each edge, thereby creating a weighted multigraph.

Graphs and Binary Relations

We can observe from our discussion of graphs that any binary relation R on a set A can be thought of as a digraph $G = (A, R)$ with vertices A and edges R. For example, let $A = \{1, 2, 3\}$ and

$$R = \{(1, 2), (1, 3), (2, 3), (3, 3)\}.$$

Figure 1.14 shows the digraph corresponding to this binary relation. Representing a binary relation as a graph is often quite useful in trying to establish properties of the relation.

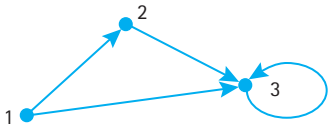


Figure 1.14 Digraph of binary relation.

Subgraphs

Sometimes we need to discuss graphs that are part of other graphs. A graph (V', E') is a *subgraph* of a graph (V, E) if $V' \subset V$ and $E' \subset E$. For example, the four graphs in Figure 1.15 are subgraphs of the graph in Figure 1.14.

1.4.2 Paths in Graphs

Problems that use graphs often involve moving from one vertex to another along a sequence of edges, where each edge shares a vertex with the next edge in the sequence. In formal terms, a *path* from x_0 to x_n is a sequence of edges that we denote by a sequence of vertices $x_0, x_1, \ldots, x_n$ such that there is an edge from x_{i-1} to x_i for $1 = i = n$. A path allows the possibility that some edge or some vertex occurs more than once. A *cycle* is a path whose beginning and ending vertices are equal and in which no edge occurs more than once. A graph with no cycles is called *acyclic*. The *length* of the path $x_0, \ldots, x_n$ is the number n of edges.

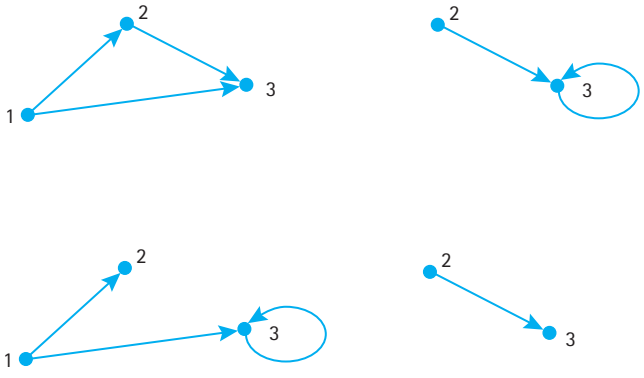


Figure 1.15 Subgraphs.

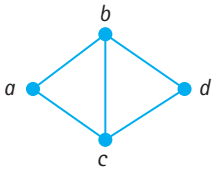


Figure 1.16 Sample graph.

example

1.35 Paths in a Graph

We'll examine a few paths in the graph pictured in Figure 1.16.

1. The path b, c, d, b, a visits b twice. The length of the path is 4.
2. The path a, b, c, b, d visits b twice uses the edge between b and c twice. The length of the path is 4.
3. The path a, b, c, a is a cycle of length 3.
4. The path a, b, a is not a cycle because the edge from a to b occurs twice. The path has length 2.

end example

Connected Graphs

Let's emphasize here that the definitions for path and cycle apply to both graphs and directed graphs. But now we come to an idea that needs a separate definition for each type of graph. A graph is *connected* if there is a path between every pair of vertices. A directed graph is *connected* if, when direction is ignored, the resulting undirected graph is connected.

Two Graph Problems

Let's look at two graph problems that you may have seen. For the first problem you are to trace the first diagram in Figure 1.17 without taking your pencil off the paper and without retracing any line.

After some fiddling, it's easy to see that it can be done by starting at one of the bottom two corners and finishing at the other bottom corner. The second diagram in Figure 1.17 emphasizes the graphical nature of the problem. From this point of view we can say that there is a path that travels each edge exactly once.

The second famous problem is named after the seven bridges of Königsberg that, in the early 1800s, connected two islands in the river Pregel to the rest of the town. The problem is to tour the town by walking across each of the seven

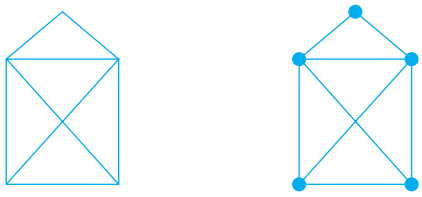


Figure 1.17 Tracing a graph.

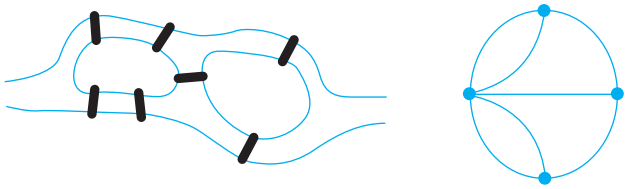


Figure 1.18 Seven bridges of Königsberg.

bridges exactly once. In Figure 1.18 we've pictured the two islands and seven bridges of Königsberg together with a multigraph representing the situation. The vertices of the multigraph represent the four land areas and the edges represent the seven bridges.

The mathematician Leonhard Euler (1707–1783) proved that there aren't any such paths by finding a general condition for such paths to exist. In his honor, any path that contains each edge of a graph exactly once is called an *Euler path*. For example, the graph for the tracing problem has an Euler path, but the seven bridges problem does not. We can go one step further and define an *Euler circuit* to be any path that begins and ends at the same vertex and contains each edge of a graph exactly once. There are no Euler circuits in the graphs of Figure 1.17 and Figure 1.18. We'll discuss conditions for the existence of Euler paths and Euler circuits in the exercises.

1.4.3 Graph Traversals

A *graph traversal* starts at some vertex v and visits all vertices x that can be reached from v by traveling along some path from v to x. If a vertex has already been visited, it is not visited again. Two popular traversal algorithms are called *breadth-first* and *depth-first*.

Breadth-First Traversal

To describe *breadth-first traversal*, we'll let visit(v, k) denote the procedure that visits every vertex x not yet visited for which there is a length k path from v to x. If the graph has n vertices, a breadth-first traversal starting at v can be

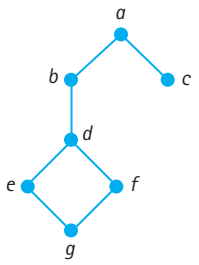


Figure 1.19 Sample graph.

described as follows:

$$\textbf{for } k := 0 \textbf{ to } n - 1 \textbf{ do } \text{visit}(v, k) \textbf{ od}.$$

Since we haven't specified how visit(v, k) does it's job, there are usually several different traversals from any given starting vertex.

example

1.36 Breadth-First Traversals

We'll do some breadth-first traversals of the graph in Figure 1.19.

If we start at vertex a, then there are four possible breadth-first traversals, which we've represented by the following strings.

$a\ b\ c\ d\ e\ f\ g$ $\qquad$ $a\ b\ c\ d\ f\ e\ g$ $\qquad$ $a\ c\ b\ d\ e\ f\ g$ $\qquad$ $a\ c\ b\ d\ f\ e\ g.$

Of course, we can start a breadth-first traversal at any vertex. For example, one of several breadth-first traversals that start with d is represented by the string

$$d\ b\ e\ f\ a\ g\ c.$$

end example

Depth-First Traversal

We can describe *depth-first traversal* with a recursive procedure–one that calls itself. Let DF(v) denote the depth-first procedure that traverses the graph starting at vertex v. Then DF(v) has the following definition.

DF(v): **if** v has not been visited **then**
visit v;
for each edge from v to x **do** DF(x) **od**
fi

Since we haven't specified how each edge from v to x is picked in the **for** loop, there are usually several different traversals from any given starting vertex.

example **1.37 Depth-First Traversals**

We'll do some depth-first traversals of the graph represented in Figure 1.19. For example, starting from vertex a in the graph, there are four possible depth-first traversals, which are represented by the following strings.

$$a\ b\ d\ e\ g\ f\ c \qquad a\ b\ d\ f\ g\ e\ c \qquad a\ c\ b\ d\ e\ g\ f \qquad a\ c\ b\ d\ f\ g\ e.$$

end example

1.4.4 Trees

From an informal point of view, a *tree* is a structure that looks like a real tree. For example, a family tree and an organizational chart for a business are both trees. In computer science, as well as many other disciplines, trees are usually drawn upside down, as in Figure 1.20. From a formal point of view we can say that a *tree* is a graph that is connected and has no cycles.

Trees have their own terminology. The elements of a tree are called *nodes*, and the lines between nodes are called *branches*. The node at the top is called the *root*. The nodes that hang immediately below a given node are its *children*, and the node immediately above a given node is its *parent*. If a node is childless, then it is a *leaf*. The *height* or *depth* of a tree is the length of the longest path from the root to the leaves.

example **1.38 Parts of a Tree**

We'll make some observations about the tree in Figure 1.20. The root is A. The children of A are B, C, and D. The parent of F is B. The leaves of the tree are are E, F, J, H, and I. The height or depth of the tree is 3.

end example

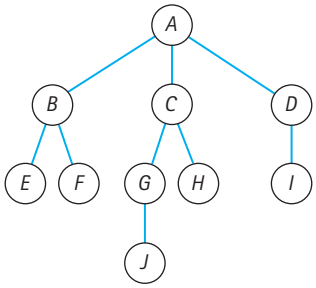


Figure 1.20 Sample tree.

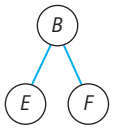


Figure 1.21 A subtree.

Subtrees

If x is a node in a tree T, then x together with all its descendants forms a tree S with x as its root. S is called a *subtree* of T. If y is the parent of x, then S is sometimes called a subtree of y.

example **1.39 A Subtree**

The tree pictured in Figure 1.21 is a subtree of the tree in Figure 1.20. Since A is the parent of B, we can also say that this tree is a subtree of node A.

end example

Ordered and Unordered Trees

If we don't care about the ordering of the children of a tree, then the tree is called an *unordered tree*. A tree is *ordered* if there is a unique ordering of the children of each node. For example, any algebraic expression can be represented as ordered tree.

example **1.40 Representing Algebraic Expressions**

The expression $x - y$ can be represented by a tree whose root is the minus sign and with two subtrees, one for x on the left and one for y on the right. Ordering is important here because the subtraction operation is not commutative. For example, Figure 1.22 shows the expression $3 - (4 + 8)$ and Figure 1.23 shows the expression $(4 + 8) - 3$.

end example

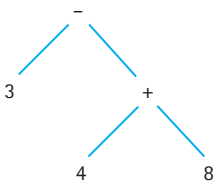


Figure 1.22 Tree for $3 - (4 + 8)$.

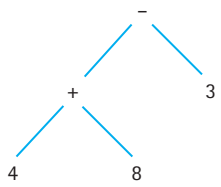


Figure 1.23 Tree for $(4 + 8) - 3$.

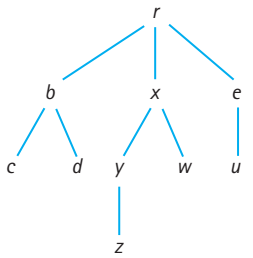


Figure 1.24 Sample tree.

Representations of Trees

How can we represent a tree as a data object? The key to any representation is that we should be able to recover the tree from its representation. One method is to let the tree be a list whose first element is the root and whose next elements are lists that represent the subtrees of the root.

For example, the tree with a single node r is represented by $\langle r \rangle$, and the list representation of the tree for the algebraic expression $a - b$ is

$$\langle -, \langle a \rangle, \langle b \rangle \rangle.$$

For another example, the list representation of the tree for the arithmetic expression $3 - (4 + 8)$ is

$$\langle -, \langle 3 \rangle, \langle +, \langle 4 \rangle, \langle 8 \rangle \rangle \rangle.$$

For a more complicated example, let's consider the tree represented by the following list.

$$T = \langle r, \langle b, \langle c \rangle, \langle d \rangle \rangle, \langle x, \langle y, \langle z \rangle \rangle, \langle w \rangle \rangle, \langle e, \langle u \rangle \rangle \rangle.$$

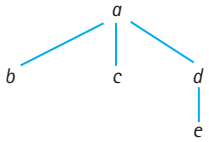
Notice that T has root r, which has the following three subtrees:

$$\langle b, \langle c \rangle, \langle d \rangle \rangle$$
$$\langle x, \langle y, \langle z \rangle \rangle, \langle w \rangle \rangle$$
$$\langle e, \langle u \rangle \rangle.$$

Similarly, the subtree $\langle b, \langle c \rangle, \langle d \rangle \rangle$ has root b, which has two children c and d. We can continue in this way to recover the picture of T in Figure 1.24.

example 1.41 Computer Repesentation of Trees

Let's represent a tree as a list and then see what it looks like in computer memory. For example, let T be the following tree.



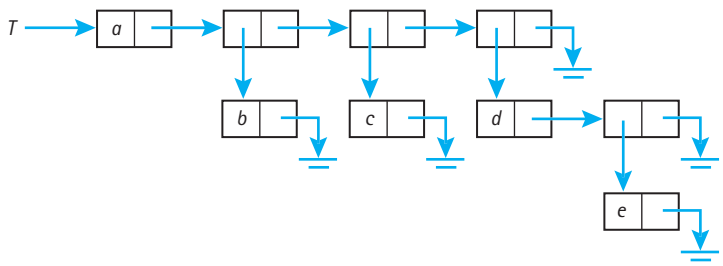


Figure 1.25 Computer representation of a tree.

We'll represent the tree as the list $T = \langle a, \langle b\rangle, \langle c\rangle, \langle d, \langle e\rangle\rangle\rangle\rangle$. Figure 1.25 shows the representation of T in computer memory, where we have used the same notation as Example 4 of Section 1.3.

end example

Binary Trees

A *binary tree* is an ordered tree that may be empty or else has the property that each node has two subtrees, called the *left* and *right* subtrees of the node, which are binary trees. We can represent the empty binary tree by the empty list $\langle\ \rangle$. Since each node has two subtrees, we represent nonempty binary trees as 3-element lists of the form

$$\langle L, x, R\rangle,$$

where x is the root, L is the left subtree, and R is the right subtree. For example, the tree with one node x is represented by the list $\langle\langle\ \rangle, x, \langle\ \rangle\rangle$.

When we draw a picture of a binary tree, it is common practice to omit the empty subtrees. For example, the binary tree represented by the list

$$\langle\langle\langle\ \rangle, a, \langle\ \rangle\rangle, b, \langle\ \rangle\rangle$$

is usually, but not always, pictured as the simpler tree in Figure 1.26.

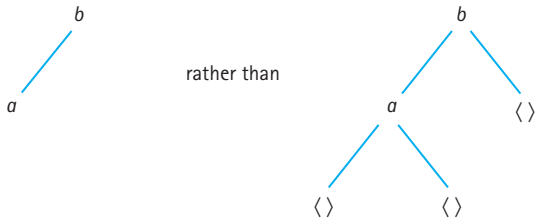


Figure 1.26 Simplified binary tree.

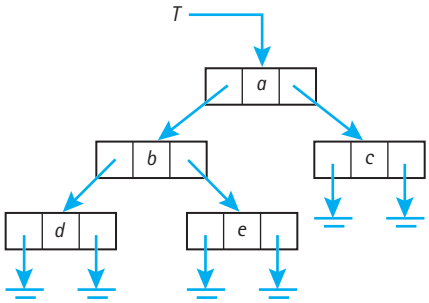
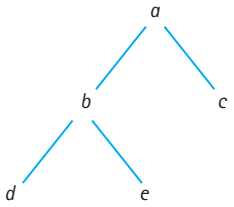


Figure 1.27 Computer representation of a binary tree.

example **1.42 Computer Repesentation of Binary Trees**

Let's see what the representation of binary tree as a list looks like in computer memory. For example, let T be the following tree.



We can represent T as the list $T = \langle a, \langle b\rangle, \langle c\rangle, \langle d, \langle e\rangle\rangle\rangle$. Figure 1.27 shows the representation of T in computer memory, where we have used the same notation used in Example 4 of Section 1.3.

end example

Binary Search Trees

Binary trees can be used to represent sets whose elements have some ordering. Such a tree is called a *binary search tree* and has the property that for each node of the tree, each element in its left subtree precedes the node element and each element in its right subtree succeeds the node element.

example **1.43 A Binary Search Tree**

The binary search tree in Figure 1.28 holds three-letter abbreviations for six of the months, where we are using the dictionary ordering of the words. So the correct order is FEB, JAN, JUL, NOV, OCT, SEP.

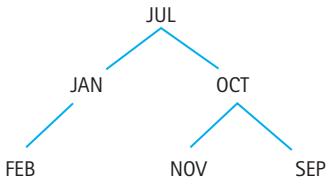


Figure 1.28 Binary search tree.

Figure 1.29 Two spanning trees.

This binary search tree has depth 2. There are many other binary search trees to hold these six months. Find another one that has depth 2. Then find one that has depth 3.

end example

1.4.5 Spanning Trees

A *spanning tree* for a connected graph is a subgraph that is a tree and contains all the vertices of the graph. For example, Figure 1.29 shows a graph followed by two of its spanning trees. This example shows that a graph can have many spanning trees. A *minimal spanning tree* for a connected weighted graph is a spanning tree such that the sum of the edge weights is minimum among all spanning trees.

Prim's Algorithm

A famous algorithm, due to Prim [1957], constructs a minimal spanning tree for any undirected connected weighted graph. Starting with any vertex, the algorithm searches for an edge of minimum weight connected to the vertex. It adds the edge to the tree and then continues by trying to find new edges of minimum weight such that one vertex is in the tree and the other vertex is not. Here's an informal description of the algorithm.

Prim's Algorithm (1.19)

Construct a minimal spanning tree for an undirected connected weighted graph. The variables: V is the vertex set of the graph; W is the vertex set and S is the edge set of the spanning tree.

Continued ➧

1. Initialize $S := \varnothing$.

2. Pick any vertex $v \in V$ and set $W := \{v\}$.

3. **while** $W \ ? \ V$ **do**

Find a minimum weight edge $\{x, y\}$ where $x \in W$ and $y \in V - W$;

$S := S \cup \{\{x, y\}\}$;

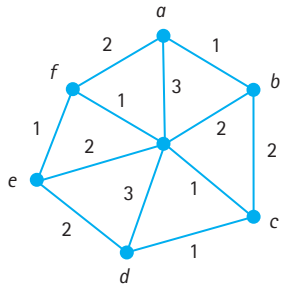
$W := W \cup \{y\}$

od

Of course, Prim's algorithm can also be used to find a spanning tree for an unweighted graph. Just assign a weight of 1 to each edge of the graph. Or modify the first statement in the **while** loop to read "Find an edge $\{x, y\}$ such that $x \in W$ and $y \in V - W$."

example **1.44 A Minimal Spanning Tree**

We'll construct a minimal spanning tree for the following weighted graph.



To see how the algorithm works we'll do a trace of each step showing the values of the variables S and W. The algorithm gives us several choices since it is not implemented as a computer program. So we'll start with the letter a since it's the first letter of the alphabet.

S	S
$\{\}$	$\{a\}$
$\{\{a, b\}\}$	$\{a, b\}$
$\{\{a, \}, \{b, c\}\}$	$\{a, b, c\}$
$\{\{a, b\}, \{b, c\}, \{c, d\}\}$	$\{a, b, c, d\}$
$\{\{a, b\}, \{b, c\}, \{c, d\}, \{c, g\}, \{g, f\}\}$	$\{a, b, c, d, g\}$
$\{\{a, b\}, \{b, c\}, \{c, d\}, \{c, g\}, \{g, f\}\}$	$\{a, b, c, d, g, f\}$
$\{\{a, b\}, \{b, c\}, \{c, d\}, \{c, g\}, \{g, f\}, \{f, e\}\}$	$\{a, b, c, d, g, f, e\}$

The algorithm stops because $W = V$. So S is a spanning tree.

end example

Exercises

Graphs

1. Draw a picture of a graph that represents those states of the United States and those provinces of Canada that touch the Atlantic Ocean or touch states or provinces that do.

2. Find planar graphs with the smallest possible number of vertices that have chromatic numbers of 1, 2, 3, and 4.

3. What is the chromatic number of the graph representing the map of the United States? Explain your answer.

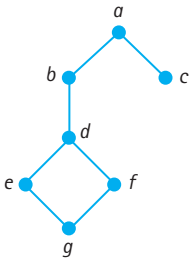
4. Draw a picture of the directed graph that corresponds to each of the following binary relations.

a. $\{(a, a), (b, b), (c, c)\}$.

b. $\{(a, b), (b, b), (b, c), (c, a)\}$.

c. The relation $=$ on the set $\{1, 2, 3\}$.

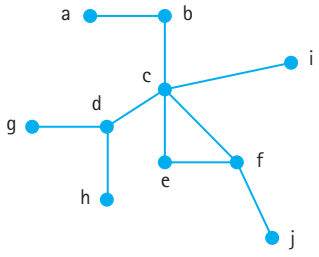
5. Given the following graph:



a. Write down all breadth-first traversals that start at vertex f.

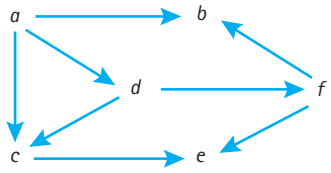
b. Write down all depth-first traversals that start at vertex f.

6. Given the following graph.



a. Write down one breadth-first traversals that start at vertex *f*.
b. Write down one depth-first traversals that start at vertex *f*.

7. Given the following graph:



a. Write down one breadth-first traversal that starts at vertex *a*.
b. Write down one depth-first traversal that starts at vertex *a*.

Trees

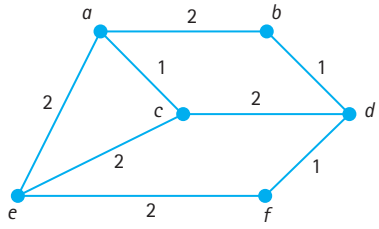
8. Given the algebraic expression $a \times (b + c) - (d \,/\, e)$. Draw a picture of the tree representation of this expression. Then convert the tree into a list representation of the expression.

9. Draw a picture of the ordered tree that is represented by the list

$$\langle a, \langle b, \langle c\rangle, \langle d, \langle e\rangle\rangle\ \rangle, \langle r, \langle s\rangle, \langle t\rangle\rangle, \langle x\rangle\rangle.$$

10. Draw a picture of a binary search tree containing the three-letter abbreviations for the 12 months of the year in dictionary order. Make sure that your tree has the least possible depth.

11. Find two distinct minimal spanning trees for the following weighted graph.



12. For the weighted graph in Example 11, find two distinct minimal spanning trees that are different from the spanning tree given.

Path Problems

13. Try to find a necessary and sufficient condition for an undirected graph to have an Euler path. *Hint:* Look at the degrees of the vertices.

14. Try to find a necessary and sufficient condition for an undirected graph to have an Euler circuit. *Hint:* Look at the degrees of the vertices.

1.5 Chapter Summary

We normally prove things informally, and we use a variety of proof techniques: proof by exhaustive checking, conditional proof, indirect proofs (i.e., proving the contrapositive and proof by contradiction), and iff proofs.

Sets are characterized by lack of order and no repeated occurrences of elements. There are easy techniques for comparing sets by subset and by equality. Sets can be combined by the operations of union, intersection, difference, and complement. Venn diagrams are useful for representing these operations. Two useful rules for counting sets are the union rule—also called the inclusion-exclusion principle—and the difference rule.

Bags—also called multisets—are characterized by lack of order, and they may contain repeated occurrences of elements.

Tuples are characterized by order, and they may contain repeated occurrences of elements. Many useful structures are related to tuples. Cartesian products of sets are collections of tuples. Lists are similar to tuples except that lists can be accessed only by head and tail. Strings are like lists except that elements from an alphabet are placed next to each other in juxtaposition. Languages are sets of strings and they can be combined by concatenating strings as well as by set operations. Relations are sets of tuples that are related in some way. A useful rule for counting tuples is the product rule.

Graphs are characterized by a set of vertices and a set of edges, where the edges may be undirected or directed. Graphs can be colored, and they can be traversed. Trees are special graphs that look like real trees. Prim's algorithm constructs a minimal spanning tree for an undirected, connected, weighted graph.