61. Let $a_1, a_2, \ldots, a_n$ be positive real numbers. The **arithmetic mean of these numbers is defined by

$$A = (a_1 + a_2 + \cdots + a_n)/n,$$

and the **geometric mean** of these numbers is defined by

$$G = (a_1 a_2 \cdots a_n)^{1/n}.$$

Use mathematical induction to prove that $A \geq G$.

*62. Use mathematical induction to show that 21 divides $4^{n+1} + 5^{2n-1}$ whenever n is a positive integer.

63. Use mathematical induction to prove Lemma 2 of Section 2.6, which states that if p is a prime and $p \mid a_1 a_2 \cdots a_n$, where a_i is an integer for $i = 1, 2, 3, \ldots, n$, then $p \mid a_i$ for some integer i.

64. Use infinite descent to show that the equation $8x^4 + 4y^4 + 2z^4 = w^4$ has no solutions in positive integers x, y, z, and w.

65. Use infinite descent to show that there are no solutions in positive integers w, x, y, and z to $w^2 + x^2 + y^2 + z^2 = 2wxyz$. (*Hint:* First show that if this equation holds, then all of w, x, y, and z must be even. Then show that all four of these integers must be divisible by 4, by 8, and so on.)

*66. The well-ordering property can be used to show that there is a unique greatest common divisor of two positive integers. Let a and b be positive integers, and let S be the set of positive integers of the form $as + bt$, where s and t are integers.

a) Show that S is nonempty.

b) Use the well-ordering property to show that S has a smallest element c.

c) Show that if d is a common divisor of a and b, then d is a divisor of c.

d) Show that $c \mid a$ and $c \mid b$. (*Hint:* First, assume that $c \nmid a$. Then $a = qc + r$, where $0 < r < c$. Show that $r \in S$, contradicting the choice of c.)

e) Conclude from (c) and (d) that the greatest common divisor of a and b exists. Finish the proof by showing that this greatest common divisor of two positive integers is unique.

*67. Show that if $a_1, a_2, \ldots, a_n$ are n distinct real numbers, exactly $n - 1$ multiplications are used to compute the product of these n numbers no matter how parentheses are inserted into their product. (*Hint:* Use strong induction and consider the last multiplication.)

68. Construct a tiling using L-shaped pieces of the 4×4 chessboard with the square in the upper left corner removed.

69. Construct a tiling using L-shaped pieces of the 8×8 chessboard with the square in the upper left corner removed.

70. Prove or disprove that all chessboards of these shapes can be completely covered using L-shaped pieces whenever n is a positive integer.

a) 3×2^n b) 6×2^n
c) $3^n \times 3^n$ d) $6^n \times 6^n$

*71. Show that a three-dimensional $2^n \times 2^n \times 2^n$ chessboard with one $1 \times 1 \times 1$ cube missing can be completely covered by $2 \times 2 \times 2$ cubes with one $1 \times 1 \times 1$ cube removed.

*72. Show that an $n \times n$ chessboard with one square removed can be completely covered using L-shaped pieces if $n > 5$, n is odd, and $3 \nmid n$.

73. Show that a 5×5 chessboard with a corner square removed can be tiled using L-shaped pieces.

*74. Find a 5×5 chessboard with a square removed that cannot be tiled using L-shaped pieces. Prove that such a tiling does not exist for this board.

75. Let a be an integer and d be a positive integer. Show that the integers q and r with $a = dq + r$ and $0 \leq r < d$, which were shown to exist in Example 16, are unique.

☞ 76. Use the principle of mathematical induction to show that $P(n)$ is true for $n = b, b + 1, b + 2, \ldots$, where b is an integer, if $P(b)$ is true and the implication $P(k) \rightarrow P(k + 1)$ is true for all positive integers k with $k \geq b$.

**77. Can you use the well-ordering property to prove this statement? "Every positive integer can be described using no more than 15 English words"?

78. Use the well-ordering principle to show that if x and y are real numbers with $x < y$, then there is a rational number r with $x < r < y$. [*Hint:* Show that there exists a positive integer A with $A > 1/(y - x)$. Then show that there is a rational number r with denominator A between x and y by looking at the numbers $\lfloor x \rfloor + j/A$, where j is a positive integer.]

3.4 Recursive Definitions and Structural Induction

INTRODUCTION

Sometimes it is difficult to define an object explicitly. However, it may be easy to define this object in terms of itself. This process is called **recursion.** For instance, the picture shown in Figure 1 is produced recursively. First, an original picture is given. Then a process of

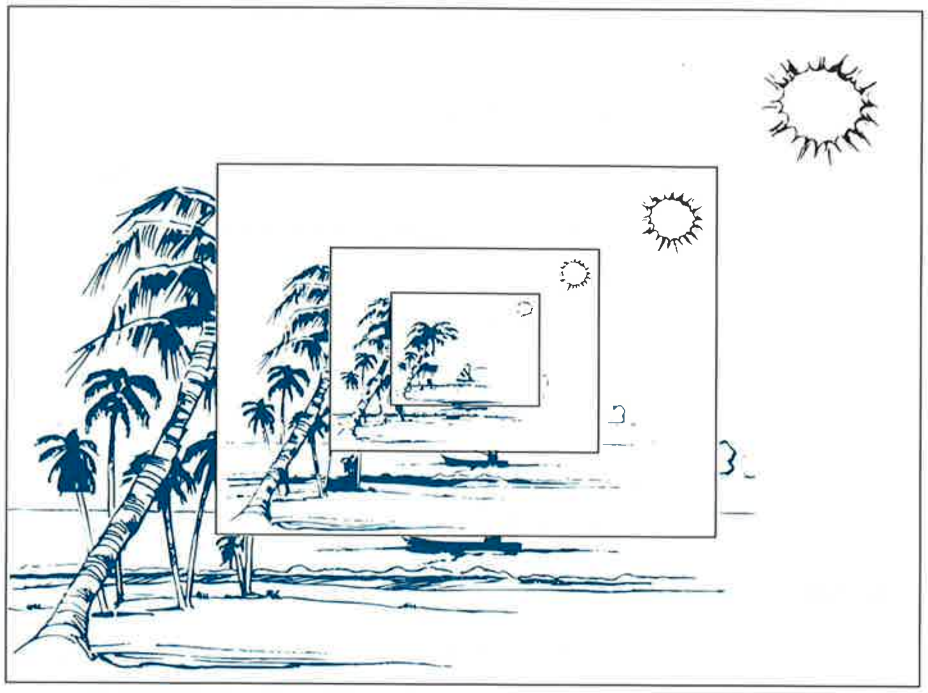

FIGURE 1 **A Recursively Defined Picture.**

successively superimposing centered smaller pictures on top of the previous pictures is carried out.

We can use recursion to define sequences, functions, and sets. In previous discussions, we specified the terms of a sequence using an explicit formula. For instance, the sequence of powers of 2 is given by $a_n = 2^n$ for $n = 0, 1, 2, \ldots$. However, this sequence can also be defined by giving the first term of the sequence, namely, $a_0 = 1$, and a rule for finding a term of the sequence from the previous one, namely, $a_{n+1} = 2a_n$ for $n = 0, 1, 2, \ldots$. When we define a sequence *recursively* by specifying how terms of the sequence are found from previous terms, we can use induction to prove results about the sequence.

When we define sets recursively, we specify some initial elements in a basis step and provide a rule for constructing new elements from those we already have in the recursive step. To prove results about recursively defined sets we use a method called *structural induction.*

RECURSIVELY DEFINED FUNCTIONS

We use two steps to define a function with the set of nonnegative integers as its domain:

BASIS STEP: Specify the value of the function at zero.

RECURSIVE STEP: Give a rule for finding its value at an integer from its values at smaller integers.

Such a definition is called a **recursive** or **inductive definition.**

EXAMPLE 1 Suppose that f is defined recursively by

Extra Examples

$$f(0) = 3,$$
$$f(n+1) = 2f(n) + 3.$$

Find $f(1), f(2), f(3)$, and $f(4)$.

Solution: From the recursive definition it follows that

$$f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9,$$
$$f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21,$$
$$f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45,$$
$$f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93.$$

◀

Many functions can be studied using their recursive definitions. The factorial function is one such example.

EXAMPLE 2 Give an inductive definition of the factorial function $F(n) = n!$.

Solution: We can define the factorial function by specifying the initial value of this function, namely, $F(0) = 1$, and giving a rule for finding $F(n+1)$ from $F(n)$. This is obtained by noting that $(n+1)!$ is computed from $n!$ by multiplying by $n+1$. Hence, the desired rule is

$$F(n+1) = (n+1)F(n).$$

◀

To determine a value of the factorial function, such as $F(5) = 5!$, from the recursive definition found in Example 2, it is necessary to use the rule that shows how to express $F(n+1)$ in terms of $F(n)$ several times:

$$\begin{aligned} F(5) &= 5F(4) = 5 \cdot 4F(3) = 5 \cdot 4 \cdot 3F(2) = 5 \cdot 4 \cdot 3 \cdot 2F(1) \\ &= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot F(0) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 120. \end{aligned}$$

Once $F(0)$ is the only value of the function that occurs, no more reductions are necessary. The only thing left to do is to insert the value of $F(0)$ into the formula.

Recursively defined functions are well defined. This is a consequence of the principle of mathematical induction. (See Exercise 56 at the end of this section.) Additional examples of recursive definitions are given in the following examples.

EXAMPLE 3 Give a recursive definition of a^n, where a is a nonzero real number and n is a nonnegative integer.

Solution: The recursive definition contains two parts. First a^0 is specified, namely, $a^0 = 1$. Then the rule for finding a^{n+1} from a^n, namely, $a^{n+1} = a \cdot a^n$, for $n = 0, 1, 2, 3, \ldots$, is given. These two equations uniquely define a^n for all nonnegative integers n. ◀

EXAMPLE 4 Give a recursive definition of

$$\sum_{k=0}^{n} a_k .$$

Solution: The first part of the recursive definition is

$$\sum_{k=0}^{0} a_k = a_0.$$

The second part is

$$\sum_{k=0}^{n+1} a_k = \left(\sum_{k=0}^{n} a_k\right) + a_{n+1}. \quad ◀$$

In some recursive definitions of functions, the values of the function at the first k positive integers are specified, and a rule is given for determining the value of the function at larger integers from its values at some or all of the preceding k integers. That such definitions produce well-defined functions follows from strong induction (see Exercise 57 at the end of this section).

DEFINITION 1

Links

The *Fibonacci numbers,* $f_0, f_1, f_2, \ldots$, are defined by the equations $f_0 = 0$, $f_1 = 1$, and

$$f_n = f_{n-1} + f_{n-2}$$

for $n = 2, 3, 4, \ldots$.

EXAMPLE 5 Find the Fibonacci numbers f_2, f_3, f_4, f_5, and f_6.

Solution: Since the first part of the definition states that $f_0 = 0$ and $f_1 = 1$, it follows from the second part of the definition that

$$f_2 = f_1 + f_0 = 1 + 0 = 1,$$
$$f_3 = f_2 + f_1 = 1 + 1 = 2,$$
$$f_4 = f_3 + f_2 = 2 + 1 = 3,$$
$$f_5 = f_4 + f_3 = 3 + 2 = 5,$$
$$f_6 = f_5 + f_4 = 5 + 3 = 8. \quad ◀$$

We can use the recursive definition of the Fibonacci numbers to prove many properties of these numbers. We give one such property in Example 6.

Links

FIBONACCI (1170–1250) Fibonacci (short for *filius Bonacci,* or "son of Bonacci") was also known as Leonardo of Pisa. He was born in the Italian commercial center of Pisa. Fibonacci was a merchant who traveled extensively throughout the Mideast, where he came into contact with Arabian mathematics. In his book *Liber Abaci,* Fibonacci introduced the European world to Arabic notation for numerals and algorithms for arithmetic. It was in this book that his famous rabbit problem (described in Section 6.1) appeared. Fibonacci also wrote books on geometry and trigonometry and on Diophantine equations, which involve finding integer solutions to equations.

EXAMPLE 6 Show that whenever $n \geq 3$, $f_n > \alpha^{n-2}$, where $\alpha = (1 + \sqrt{5})/2$.

Extra Examples

Solution: We can use strong induction to prove this inequality. Let $P(n)$ be the statement $f_n > \alpha^{n-2}$. We want to show that $P(n)$ is true whenever n is an integer greater than or equal to 3.

BASIS STEP: First, note that

$$\alpha < 2 = f_3, \qquad \alpha^2 = (3 + \sqrt{5})/2 < 3 = f_4,$$

so that $P(3)$ and $P(4)$ are true.

INDUCTIVE STEP: Assume that $P(j)$ is true, namely, that $f_j > \alpha^{j-2}$, for all integers j with $3 \leq j \leq k$, where $k \geq 4$. We must show that $P(k+1)$ is true, that is, that $f_{k+1} > \alpha^{k-1}$. Since α is a solution of $x^2 - x - 1 = 0$ (as the quadratic formula shows), it follows that $\alpha^2 = \alpha + 1$. Therefore,

$$\alpha^{k-1} = \alpha^2 \cdot \alpha^{k-3} = (\alpha + 1)\alpha^{k-3} = \alpha \cdot \alpha^{k-3} + 1 \cdot \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}.$$

By the inductive hypothesis, if $k \geq 4$, it follows that

$$f_{k-1} > \alpha^{k-3}, \qquad f_k > \alpha^{k-2}.$$

Therefore, we have

$$f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}.$$

It follows that $P(k+1)$ is true. This completes the proof. ◀

Remark: The inductive step shows that whenever $k \geq 4$, $P(k+1)$ follows from the assumption that $P(j)$ is true for $3 \leq j \leq k$. Hence, the inductive step does *not* show that $P(3) \rightarrow P(4)$. Therefore, we had to show that $P(4)$ is true separately.

We can now show that the Euclidean algorithm uses $O(\log b)$ divisions to find the greatest common divisor of the positive integers a and b, where $a \geq b$.

THEOREM 1 **LAMÉ'S THEOREM** Let a and b be positive integers with $a \geq b$. Then the number of divisions used by the Euclidean algorithm to find $\gcd(a, b)$ is less than or equal to five times the number of decimal digits in b.

GABRIEL LAMÉ (1795–1870) Gabriel Lamé entered the École Polytechnique in 1813, graduating in 1817. He continued his education at the École des Mines, graduating in 1820.

In 1820 Lamé went to Russia, where he was appointed director of the Schools of Highways and Transportation in St. Petersburg. Not only did he teach, but he also planned roads and bridges while in Russia. He returned to Paris in 1832, where he helped found an engineering firm. However, he soon left the firm, accepting the chair of physics at the École Polytechnique, which he held until 1844. While holding this position, he was active outside academia as an engineering consultant, serving as chief engineer of mines and participating in the building of railways.

Lamé contributed original work to number theory, applied mathematics, and thermodynamics. His best-known work involves the introduction of curvilinear coordinates. His work on number theory includes proving Fermat's Last Theorem for $n = 7$, as well as providing the upper bound for the number of divisions used by the Euclidean algorithm given in this text.

Links

In the opinion of Gauss, one of the most important mathematicians of all time, Lamé was the foremost French mathematician of his time. However, French mathematicians considered him too practical, whereas French scientists considered him too theoretical.

Proof: Recall that when the Euclidean algorithm is applied to find $\gcd(a, b)$ with $a \geq b$, this sequence of equations (where $a = r_0$ and $b = r_1$) is obtained.

$$\begin{aligned}
r_0 &= r_1 q_1 + r_2 & 0 &\leq r_2 < r_1 \\
r_1 &= r_2 q_2 + r_3 & 0 &\leq r_3 < r_2 \\
&\vdots \\
r_{n-2} &= r_{n-1} q_{n-1} + r_n & 0 &\leq r_n < r_{n-1} \\
r_{n-1} &= r_n q_n.
\end{aligned}$$

Here n divisions have been used to find $r_n = \gcd(a, b)$. Note that the quotients $q_1, q_2, \ldots, q_{n-1}$ are all at least 1. Moreover, $q_n \geq 2$, since $r_n < r_{n-1}$. This implies that

$$\begin{aligned}
r_n &\geq 1 = f_2, \\
r_{n-1} &\geq 2r_n \geq 2f_2 = f_3, \\
r_{n-2} &\geq r_{n-1} + r_n \geq f_3 + f_2 = f_4, \\
&\vdots \\
r_2 &\geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n, \\
b = r_1 &\geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}.
\end{aligned}$$

It follows that if n divisions are used by the Euclidean algorithm to find $\gcd(a, b)$ with $a \geq b$, then $b \geq f_{n+1}$. From Example 6 we know that $f_{n+1} > \alpha^{n-1}$ for $n > 2$, where $\alpha = (1 + \sqrt{5})/2$. Therefore, it follows that $b > \alpha^{n-1}$. Furthermore, since $\log_{10} \alpha \sim 0.208 > 1/5$, we see that

$$\log_{10} b > (n - 1) \log_{10} \alpha > (n - 1)/5.$$

Hence, $n - 1 < 5 \cdot \log_{10} b$. Now suppose that b has k decimal digits. Then $b < 10^k$ and $\log_{10} b < k$. It follows that $n - 1 < 5k$, and since k is an integer, it follows that $n \leq 5k$. This finishes the proof. ◁

Since the number of decimal digits in b, which equals $\lfloor \log_{10} b \rfloor + 1$, is less than or equal to $\log_{10} b + 1$, Theorem 1 tells us that the number of divisions required to find $\gcd(a, b)$ with $a > b$ is less than or equal to $5(\log_{10} b + 1)$. Since $5(\log_{10} b + 1)$ is $O(\log b)$, we see that $O(\log b)$ divisions are used by the Euclidean algorithm to find $\gcd(a, b)$ whenever $a > b$.

RECURSIVELY DEFINED SETS AND STRUCTURES

We have explored how functions can be defined recursively. We now turn our attention to how sets can be defined recursively. Just as in the recursive definition of functions, recursive definitions of sets have two parts, a **basis step** and a **recursive step.** In the basis step, an initial collection of elements is specified. In the recursive step, rules for forming new elements in the set from those already known to be in the set are provided. Recursive definitions may also include an **exclusion rule,** which specifies that a recursively defined set contains nothing other than those elements specified in the basis step or generated by applications of the recursive step. In our discussions, we will always tacitly assume that the exclusion rule holds and no element belongs to a recursively defined set unless it is in the initial collection specified in the basis step or can be generated using the recursive step one or more times. Later we will see how we can use a technique known as structural induction to prove results about recursively defined sets.

Examples 7, 8, 10, and 11 illustrate the recursive definition of sets. In each example, we show those elements generated by the first few applications of the recursive step.

EXAMPLE 7 Consider the subset S of the set of integers defined by

Extra Examples

BASIS STEP: $3 \in S$.

RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y \in S$.

The new elements found to be in S are 3 by the basis step, $3 + 3 = 6$ at the first application of the recursive step, $3 + 6 = 6 + 3 = 9$ and $6 + 6 = 12$ at the second application of the recursive step, and so on. ◀

Recursive definitions play an important role in the study of strings. (See Chapter 11 for an introduction to the theory of formal languages, for example.) Recall from Section 3.2 that a string over an alphabet Σ is a finite sequence of symbols from Σ. We can define Σ^*, the set of strings over Σ, recursively, as Definition 2 shows.

DEFINITION 2 The set Σ^* of *strings* over the alphabet Σ can be defined recursively by

BASIS STEP: $\lambda \in \Sigma^*$ (where λ is the empty string containing no symbols).

RECURSIVE STEP: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.

The basis step of the recursive definition of strings says that the empty string belongs to Σ^*. The recursive step states that new strings are produced by adding a symbol from Σ to the end of strings in Σ^*. At each application of the recursive step, strings containing one additional symbol are generated.

EXAMPLE 8 If $\Sigma = \{0, 1\}$, the strings found to be in Σ^*, the set of all bit strings, are λ, specified to be in Σ^* in the basis step, 0 and 1 formed during the first application of the recursive step, 00, 01, 10, and 11 formed during the second application of the recursive step, and so on. ◀

Recursive definitions can be used to define operations or functions on the elements of recursively defined sets. This is illustrated in Definition 3 of the concatenation of two strings and Example 9 concerning the length of a string.

DEFINITION 3 Two strings can be combined via the operation of *concatenation*. Let Σ be a set of symbols and Σ^* the set of strings formed from symbols in Σ. We can define the concatenation of two strings, denoted by $\cdot$, recursively as follows.

BASIS STEP: If $w \in \Sigma^*$, then $w \cdot \lambda = w$, where λ is the empty string.

RECURSIVE STEP: If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

The concatenation of the strings w_1 and w_2 is often written as $w_1 w_2$ rather than $w_1 \cdot w_2$. By repeated application of the recursive definition, it follows that the concatenation of two strings w_1 and w_2 consists of the symbols in w_1 followed by the symbols in w_2. For instance, the concatenation of $w_1 = abra$ and $w_2 = cadabra$ is $w_1 w_2 = abracadabra$.

EXAMPLE 9 **Length of a String** Give a recursive definition of $l(w)$, the length of the string w.

Solution: The length of a string can be defined by

$$l(\lambda) = 0;$$
$$l(wx) = l(w) + 1 \text{ if } w \in \Sigma^* \text{ and } x \in \Sigma.$$ ◀

Another important use of recursive definitions is to define **well-formed formulae** of various types. This is illustrated in Examples 10 and 11.

EXAMPLE 10 **Well-Formed Formulae for Compound Propositions** We can define the set of well-formed formulae for compound propositions involving **T**, **F**, propositional variables, and operators from the set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

BASIS STEP: **T**, **F**, and p, where p is a propositional variable, are well-formed formulae.

RECURSIVE STEP: If E and F are well-formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, and $(E \leftrightarrow F)$ are well-formed formulae.

For example, by the basis step we know that **T**, **F**, p, and q are well-formed formulae, where p and q are propositional variables. From an initial application of the recursive step, we know that $(p \vee q)$, $(p \rightarrow \mathbf{F})$, $(\mathbf{F} \rightarrow q)$, and $(q \wedge \mathbf{F})$ are well-formed formulae. A second application of the recursive step shows that $((p \vee q) \rightarrow (q \wedge \mathbf{F}))$, $(q \vee (p \vee q))$, and $((p \rightarrow \mathbf{F}) \rightarrow \mathbf{T})$ are well-formed formulae. ◀

EXAMPLE 11 **Well-Formed Formulae of Operators and Operands** We can define the set of well-formed formulae consisting of variables, numerals, and operators from the set $\{+, -, *, /, \uparrow\}$ (where $*$ denotes multiplication and $\uparrow$ denotes exponentiation) recursively.

BASIS STEP: x is a well-formed formula if x is a numeral or variable.

RECURSIVE STEP: If F and G are well-formed formulae, then $(F + G)$, $(F - G)$, $(F * G)$, (F/G), and $(F \uparrow G)$ are well-formed formulae.

For example, by the basis step we see that $x, y, 0$, and 3 are well-formed formulae (as is any variable or numeral). Well-formed formulae generated by applying the recursive step once include $(x + 3)$, $(3 + y)$, $(x - y)$, $(3 - 0)$, $(x * 3)$, $(3 * y)$, $(3/0)$, (x/y), $(3 \uparrow x)$, and $(0 \uparrow 3)$. Applying the recursive step twice shows that formulae such as $((x + 3) + 3)$ and $(x - (3 * y))$ are well-formed formulae. [Note that $(3/0)$ is a well-formed formula since we are concerned only with syntax matters here.] ◀

We will study trees extensively in Chapter 9. A tree is a special type of a graph; a graph is made up of vertices and edges connecting some pairs of vertices. We will study graphs in Chapter 8. We will briefly introduce them here to illustrate how they can be defined recursively.

DEFINITION 4 The set of *rooted trees,* where a rooted tree consists of a set of vertices containing a distinguished vertex called the *root,* and edges connecting these vertices, can be defined recursively by these steps:

BASIS STEP: A single vertex r is a rooted tree.

RECURSIVE STEP: Suppose that $T_1, T_2, \ldots, T_n$ are rooted trees with roots $r_1, r_2, \ldots, r_n$, respectively. Then the graph formed by starting with a root r, which is not in any of the rooted trees $T_1, T_2, \ldots, T_n$, and adding an edge from r to each of the vertices $r_1, r_2, \ldots, r_n$, is also a rooted tree.

In Figure 2 we illustrate some of the rooted trees formed starting with the basis step and applying the recursive step one time and two times. Note that infinitely many rooted trees are formed at each application of the recursive definition.

Rooted trees are a special type of binary trees. We will provide recursive definitions of two types of binary trees, full binary trees and extended binary trees. In the recursive step

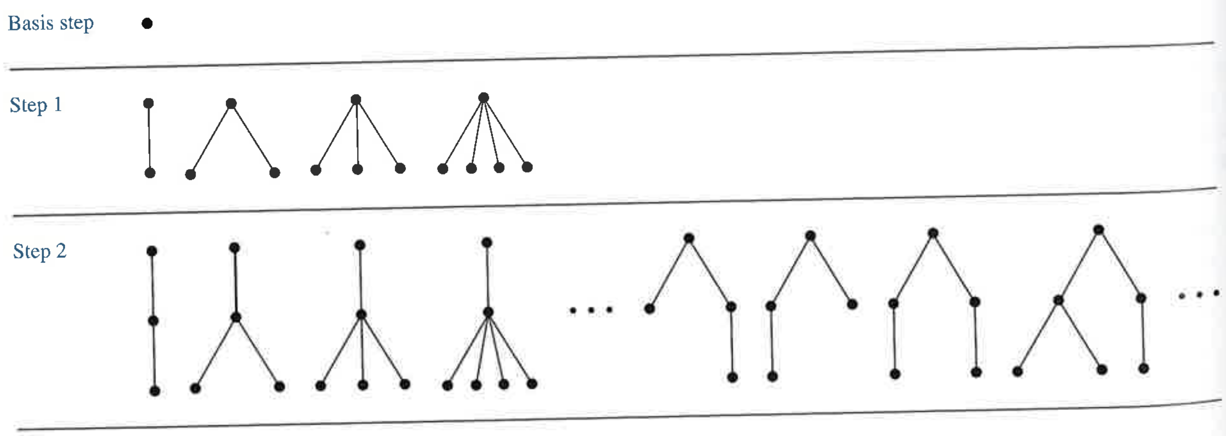

FIGURE 2 **Building Up Rooted Trees.**

of the definition of each type of binary tree, two binary trees are combined to form a new tree with one of these trees designated the left subtree and the other the right subtree. In extended binary trees, the left subtree or the right subtree can be empty, but in full binary trees this is not possible. Binary trees are one of the most important types of structures in computer science. In Chapter 9 we will see how they can be used in searching and sorting algorithms, in algorithms for compressing data, and in many other applications. We first define extended binary trees.

DEFINITION 5

The set of *extended binary trees* can be defined recursively by these steps:

BASIS STEP: The empty set is an extended binary tree.

RECURSIVE STEP: If T_1 and T_2 are extended binary trees, there is an extended binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2 when these trees are nonempty.

Figure 3 shows how extended binary trees are built up by applying the recursive step from one to three times.

We now show how to define the set of full binary trees. Note that the difference between this recursive definition and that of extended binary trees lies entirely in the basis step.

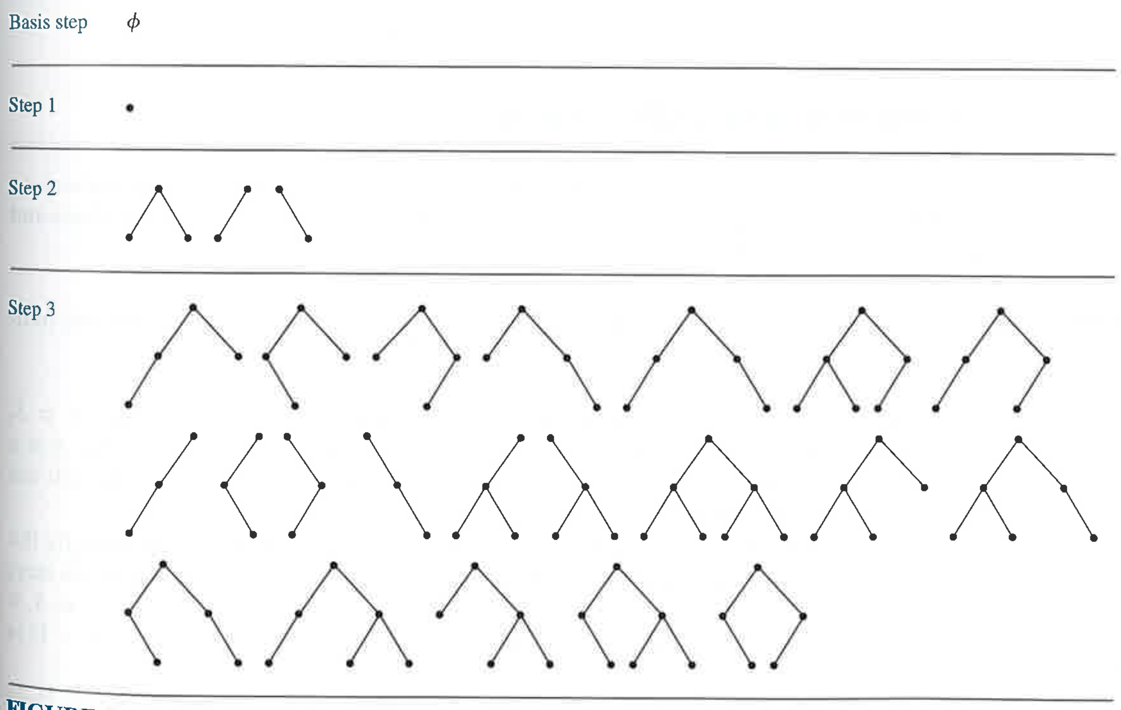

FIGURE 3 **Building Up Extended Binary Trees.**

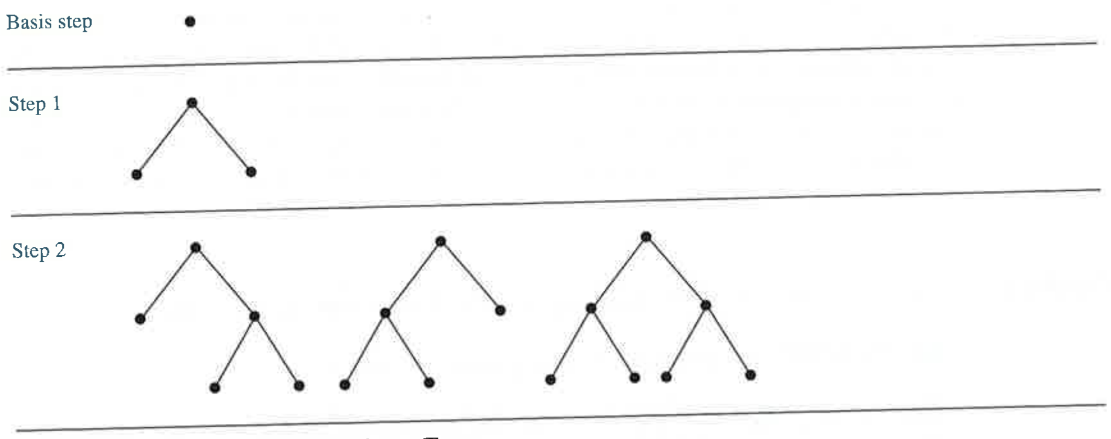

FIGURE 4 **Building Up Full Binary Trees.**

DEFINITION 6

The set of full binary trees can be defined recursively by these steps:

BASIS STEP: There is a full binary tree consisting only of a single vertex r.

RECURSIVE STEP: If T_1 and T_2 are full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2.

Figure 4 shows how full binary trees are built up by applying the recursive step one and two times.

STRUCTURAL INDUCTION

To prove results about recursively defined sets we generally use some form of mathematical induction. Example 12 illustrates the connection between recursively defined sets and mathematical induction.

EXAMPLE 12 Show that the set S defined in Example 7 is the set of all positive integers that are multiples of 3.

Solution: Let A be the set of all positive integers divisible by 3. To prove that $A = S$, we must show that A is a subset of S and that S is a subset of A. To prove that A is a subset of S, we must show that every positive integer divisible by 3 is in S. We will use mathematical induction to prove this.

Let $P(n)$ be the statement that $3n$ belongs to S. The basis step holds since by the first part of the recursive definition of S, $3 \cdot 1 = 3$ is in S. To establish the inductive step, assume that $P(k)$ is true, namely, that $3k$ is in S. Since $3k$ is in S and since 3 is in S, it follows from the second part of the recursive definition of S that $3k + 3 = 3(k + 1)$ is also in S.

To prove that S is a subset of A, we use the recursive definition of S. First, the basis step of the definition specifies that 3 is in S. Since $3 = 3 \cdot 1$, all elements specified to be in S in this step are divisible by 3. To finish the proof, we must show that all integers in S generated using the second part of the recursive definition are in A. This consists of showing that $x + y$ is in A whenever x and y are elements of S also assumed to be in A. Now if x and y are both in A, it follows that $3 \mid x$ and $3 \mid y$. By Theorem 1 of Section 2.4, it follows that $3 \mid x + y$, completing the proof. ◀

In Example 12 we used mathematical induction over the set of positive integers and a recursive definition to prove a result about a recursively defined set. However, instead of using mathematical induction directly to prove results about recursively defined sets, we can use a more convenient form of induction known as **structural induction.** A proof by structural induction consists of two parts. These parts are

BASIS STEP: Show that the result holds for all elements specified in the basis step of the recursive definition to be in the set.

RECURSIVE STEP: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

The validity of structural induction follows from the principle of mathematical induction for the nonnegative integers. To see this, let $P(n)$ state that the claim is true for all elements of the set that are generated by n or fewer applications of the rules in the recursive step of a recursive definition. We will have established that the principle of mathematical induction implies the principle of structural induction if we can show that $P(n)$ is true whenever n is a positive integer. In the basis step of a proof by structural induction we show that $P(0)$ is true. That is, we show that the result is true of all elements specified to be in the set in the basis step of the definition. A consequence of the inductive step is that if we assume $P(k)$ is true, it follows that $P(k + 1)$ is true. When we have completed a proof using structural induction, we have shown that $P(0)$ is true and that $P(k)$ implies $P(k + 1)$. By mathematical induction it follows that $P(n)$ is true for all nonnegative integers n. This also shows that the result is true for all elements generated by the recursive definition, and shows that structural induction is a valid proof technique.

EXAMPLES OF PROOFS USING STRUCTURAL INDUCTION To use structural induction to prove a result about the set of well-formed expressions defined in Example 10, we need to complete this basis step and this recursive step.

BASIS STEP: Show that the result is true for $\mathbf{T}$, $\mathbf{F}$, and p whenever p is a propositional variable.

RECURSIVE STEP: Show that if the result is true for the compound propositions p and q, it is also true for $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$, and $(p \leftrightarrow q)$.

Example 13 illustrates how we can prove results about well-formed formulae using structural induction.

EXAMPLE 13 Show that every well-formed formula for compound propositions, as defined in Example 10, contains an equal number of left and right parentheses. ◀

Proof:

BASIS STEP: Each of the formulae $\mathbf{T}$, $\mathbf{F}$, and p contains no parentheses, so clearly they contain an equal number of left and right parentheses.

RECURSIVE STEP: Assume p and q are well-formed formulae each containing an equal number of left and right parentheses. That is, if l_p and l_q are the number of left parentheses in p and q, respectively, and r_p and r_q are the number of right parentheses in p and q, respectively, then $l_p = r_p$ and $l_q = r_q$. To complete the inductive step, we need to show that each of $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$, and $(p \leftrightarrow q)$ also contains an equal number of left and right parentheses. The number of left parentheses in the first of these compound propositions equals $l_p + 1$ and in each of the other compound propositions equals $l_p + l_q + 1$. Similarly, the number of right parentheses in the first of these compound propositions equals $r_p + 1$ and in each of the other compound propositions equals $r_p + r_q + 1$. Since $l_p = r_p$ and $l_q = r_q$, it follows that each of these compound expressions contains the same number of left and right parentheses. This completes the inductive proof. ◁

Suppose that $P(w)$ is a propositional function over the set of strings $w \in \Sigma^*$. To use structural induction to prove that $P(w)$ holds for all strings $w \in \Sigma^*$, we need to complete both a basis step and a recursive step. These steps are:

BASIS STEP: Show that $P(\lambda)$ is true.

RECURSIVE STEP: Assume that $P(w)$ is true, where $w \in \Sigma^*$. Show that if $x \in \Sigma$, then $P(wx)$ must also be true.

Example 14 illustrates how structural induction can be used in proofs about strings.

EXAMPLE 14 Use structural induction to prove that $l(xy) = l(x) + l(y)$, where x and y belong to Σ^*, the set of strings over the alphabet Σ.

Solution: We will base our proof on the recursive definition of the set Σ^* given in Definition 2 and the definition of the length of a string in Example 9. Let $P(y)$ be the statement that $l(xy) = l(x) + l(y)$ whenever x belongs to Σ^*.

BASIS STEP: To complete the basis step, we must show that $P(\lambda)$ is true. That is, we must show that $l(x\lambda) = l(x) + l(\lambda)$ for all $x \in \Sigma^*$. Since $l(x\lambda) = l(x) = l(x) + 0 = l(x) + l(\lambda)$ for every string x, it follows that $P(\lambda)$ is true.

RECURSIVE STEP: To complete the inductive step, we assume that $P(y)$ is true and show that this implies that $P(ya)$ is true whenever $a \in \Sigma$. What we need to show is that $l(xya) = l(x) + l(ya)$ for every $a \in \Sigma$. To show this, note that by the recursive definition of $l(w)$ (given in Example 9), we have $l(xya) = l(xy) + 1$ and $l(ya) = l(y) + 1$. And, by the inductive hypothesis, $l(xy) = l(x) + l(y)$. We conclude that $l(xya) = l(x) + l(y) + 1 = l(x) + l(ya)$. ◀

We can prove results about trees or special classes of trees using structural induction. For example, to prove a result about full binary trees using structural induction we need to complete this basis step and this recursive step.

BASIS STEP: Show that the result is true for the tree consisting of a single vertex.

RECURSIVE STEP: Show that if the result is true for the trees T_1 and T_2, then it is true for tree $T_1 \cdot T_2$ consisting of a root r, which has T_1 as its left subtree and T_2 as its right subtree.

Before we provide an example showing how structural induction can be used to prove a result about full binary trees, we need some definitions. We will recursively define the height $h(T)$ and the number of vertices $n(T)$ of a full binary tree T. We begin by defining the height of a full binary tree.

DEFINITION 7 We define the height $h(T)$ of a full binary tree T recursively.

BASIS STEP: The height of the full binary tree T consisting of only a root r is $h(T) = 0$.

RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + \max(h(T_1), h(T_2))$.

If we let $n(T)$ denote the number of vertices in a full binary tree, we observe that $n(T)$ satisfies the following recursive formula:

BASIS STEP: The number of vertices $n(T)$ of the full binary tree T consisting of only a root r is $n(T) = 1$.

RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the number of vertices of the full binary tree $T = T_1 \cdot T_2$ is $n(T) = 1 + n(T_1) + n(T_2)$.

We now show how structural induction can be used to prove a result about full binary trees.

THEOREM 2 If T is a full binary tree T, then $n(T) \leq 2^{h(T)+1} - 1$.

Proof: We prove this inequality using structural induction.

BASIS STEP: For the full binary tree consisting of just the root r the result is true since $n(T) = 1$ and $h(T) = 0$, so that $n(T) = 1 \leq 2^{0+1} - 1 = 1$.

INDUCTIVE STEP: For the inductive hypothesis we assume that $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$ whenever T_1 and T_2 are full binary trees. By the recursive formulae for $n(T)$ and $h(T)$ we have $n(T) = 1 + n(T_1) + n(T_2)$ and $h(T) = 1 + \max(h(T_1), h(T_2))$.

We find that

$$\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{by the recursive formula for } n(T)\\
&\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) && \text{by the inductive hypothesis}\\
&\leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 && \text{since the sum of two terms is at most 2 times the larger}\\
&= 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1\\
&= 2 \cdot 2^{h(T)} - 1 && \text{by the recursive definition of } h(T)\\
&= 2^{h(T)+1} - 1.
\end{aligned}$$

This completes the inductive step. ◁

GENERALIZED INDUCTION

We can extend mathematical induction to prove results about other sets that have the well-ordering property besides the set of integers. Although we will discuss this concept in detail in Section 7.6, we provide an example here to illustrate the usefulness of such an approach.

As an example, note that we can define an ordering on $\mathbf{N} \times \mathbf{N}$, the ordered pairs of nonnegative integers, by specifying that (x_1, y_1) is less than or equal to (x_2, y_2) if either $x_1 < x_2$, or $x_1 = x_2$ and $y_1 < y_2$; this is called the **lexicographic ordering.** The set $\mathbf{N} \times \mathbf{N}$ with this ordering has the property that every subset of $\mathbf{N} \times \mathbf{N}$ has a least element (see Supplementary Exercise 47 in Chapter 7). This implies that we can recursively define the terms $a_{m,n}$, with $m \in \mathbf{N}$ and $n \in \mathbf{N}$, and prove results about them using a variant of mathematical induction, as illustrated in Example 15.

EXAMPLE 15 Suppose that $a_{m,n}$ is defined recursively for $(m, n) \in \mathbf{N} \times \mathbf{N}$ by $a_{0,0} = 0$ and

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\ a_{m,n-1} + n & \text{if } n > 0. \end{cases}$$

Show that $a_{m,n} = m + n(n+1)/2$ for all $(m, n) \in \mathbf{N} \times \mathbf{N}$, that is, for all pairs of nonnegative integers.

Solution: We can prove that $a_{m,n} = m + n(n+1)/2$ using a generalized version of mathematical induction. The basis step requires that we show that this formula is valid when $(m, n) = (0, 0)$. The induction step requires that we show that if the formula holds for all pairs smaller than (m, n) in the lexicographic ordering of $\mathbf{N} \times \mathbf{N}$, then it also holds for (m, n).

BASIS STEP: Let $(m, n) = (0, 0)$. Then by the basis case of the recursive definition of $a_{m,n}$ we have $a_{0,0} = 0$. Furthermore, when $m = n = 0$, $m + n(n+1)/2 = 0 + (0 \cdot 1)/2 = 0$. This completes the basis step.

INDUCTIVE STEP: Suppose that $a_{m',n'} = m' + n'(n'+1)/2$ whenever (m', n') is less than (m, n) in the lexicographic ordering of $\mathbf{N} \times \mathbf{N}$. By the recursive definition, if $n = 0$, then $a_{m,n} = a_{m-1,n} + 1$. Because $(m-1, n)$ is smaller than (m, n), the induction hypothesis tells us that $a_{m-1,n} = m - 1 + n(n+1)/2$, so that $a_{m,n} = m - 1 + n(n+1)/2 + 1 = m + n(n+1)/2$, giving us the desired equality. Now suppose that $n > 0$, so $a_{m,n} = a_{m,n-1} + n$. Since $(m, n-1)$ is smaller than (m, n), the induction hypothesis tells us that $a_{m,n-1} = m + (n-1)n/2$, so $a_{m,n} = m + (n-1)n/2 + n = m + (n^2 - n + 2n)/2 = m + n(n+1)/2$. This finishes the inductive step. ◀

As mentioned, we will justify this proof technique in Section 7.6.

Exercises

1. Find $f(1)$, $f(2)$, $f(3)$, and $f(4)$ if $f(n)$ is defined recursively by $f(0) = 1$ and for $n = 0, 1, 2, \ldots$

a) $f(n+1) = f(n) + 2$.
b) $f(n+1) = 3f(n)$.
c) $f(n+1) = 2^{f(n)}$.
d) $f(n+1) = f(n)^2 + f(n) + 1$.

2. Find $f(1)$, $f(2)$, $f(3)$, $f(4)$, and $f(5)$ if $f(n)$ is defined recursively by $f(0) = 3$ and for $n = 0, 1, 2, \ldots$

a) $f(n+1) = -2f(n)$.
b) $f(n+1) = 3f(n) + 7$.
c) $f(n+1) = f(n)^2 - 2f(n) - 2$.
d) $f(n+1) = 3^{f(n)/3}$.

3. Find $f(2)$, $f(3)$, $f(4)$, and $f(5)$ if f is defined recursively by $f(0) = -1$, $f(1) = 2$ and for $n = 1, 2, \ldots$

a) $f(n+1) = f(n) + 3f(n-1)$.
b) $f(n+1) = f(n)^2 f(n-1)$.

c) $f(n+1) = 3f(n)^2 - 4f(n-1)^2$.
d) $f(n+1) = f(n-1)/f(n)$.

4. Find $f(2), f(3), f(4)$, and $f(5)$ if f is defined recursively by $f(0) = f(1) = 1$ and for $n = 1, 2, \ldots$

a) $f(n+1) = f(n) - f(n-1)$.
b) $f(n+1) = f(n)f(n-1)$.
c) $f(n+1) = f(n)^2 + f(n-1)^3$.
d) $f(n+1) = f(n)/f(n-1)$.

5. Determine whether each of these proposed definitions is a valid recursive definition of a function f from the set of nonnegative integers to the set of integers. If f is well defined, find a formula for $f(n)$ when n is a nonnegative integer and prove that your formula is valid.

a) $f(0) = 0, f(n) = 2f(n-2)$ for $n \geq 1$
b) $f(0) = 1, f(n) = f(n-1) - 1$ for $n \geq 1$
c) $f(0) = 2, f(1) = 3, f(n) = f(n-1) - 1$ for $n \geq 2$
d) $f(0) = 1, f(1) = 2, f(n) = 2f(n-2)$ for $n \geq 2$
e) $f(0) = 1, f(n) = 3f(n-1)$ if n is odd and $n \geq 1$ and $f(n) = 9f(n-2)$ if n is even and $n \geq 2$

6. Determine whether each of these proposed definitions is a valid recursive definition of a function f from the set of nonnegative integers to the set of integers. If f is well defined, find a formula for $f(n)$ when n is a nonnegative integer and prove that your formula is valid.

a) $f(0) = 1, f(n) = -f(n-1)$ for $n \geq 1$
b) $f(0) = 1, f(1) = 0, f(2) = 2, f(n) = 2f(n-3)$ for $n \geq 3$
c) $f(0) = 0, f(1) = 1, f(n) = 2f(n+1)$ for $n \geq 2$
d) $f(0) = 0, f(1) = 1, f(n) = 2f(n-1)$ for $n \geq 1$
e) $f(0) = 2, f(n) = f(n-1)$ if n is odd and $n \geq 1$ and $f(n) = 2f(n-2)$ if $n \geq 2$

7. Give a recursive definition of the sequence $\{a_n\}$, $n = 1, 2, 3, \ldots$ if

a) $a_n = 6n$. **b)** $a_n = 2n + 1$.
c) $a_n = 10^n$. **d)** $a_n = 5$.

8. Give a recursive definition of the sequence $\{a_n\}$, $n = 1, 2, 3, \ldots$ if

a) $a_n = 4n - 2$. **b)** $a_n = 1 + (-1)^n$.
c) $a_n = n(n+1)$. **d)** $a_n = n^2$.

9. Let F be the function such that $F(n)$ is the sum of the first n positive integers. Give a recursive definition of $F(n)$.

10. Give a recursive definition of $S_m(n)$, the sum of the integer m and the nonnegative integer n.

11. Give a recursive definition of $P_m(n)$, the product of the integer m and the nonnegative integer n.

In Exercises 12–19 f_n is the nth Fibonacci number.

12. Prove that $f_1^2 + f_2^2 + \cdots + f_n^2 = f_n f_{n+1}$ whenever n is a positive integer.

13. Prove that $f_1 + f_3 + \cdots + f_{2n-1} = f_{2n}$ whenever n is a positive integer.

***14.** Show that $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$ whenever n is a positive integer.

***15.** Show that $f_0 f_1 + f_1 f_2 + \cdots + f_{2n-1} f_{2n} = f_{2n}^2$ whenever n is a positive integer.

***16.** Show that $f_0 - f_1 + f_2 - \cdots - f_{2n-1} + f_{2n} = f_{2n-1} - 1$ whenever n is a positive integer.

17. Determine the number of divisions used by the Euclidean algorithm to find the greatest common divisor of the Fibonacci numbers f_n and f_{n+1} where n is a nonnegative integer. Verify your answer using mathematical induction.

18. Let

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Show that

$$\mathbf{A}^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$$

whenever n is a positive integer.

19. By taking determinants of both sides of the equation in Exercise 18, prove the identity given in Exercise 14. (This exercise depends on the notion of the determinant of a 2×2 matrix.)

***20.** Give a recursive definition of the functions max and min so that $\max(a_1, a_2, \ldots, a_n)$ and $\min(a_1, a_2, \ldots, a_n)$ are the maximum and minimum of the n numbers $a_1, a_2, \ldots, a_n$, respectively.

***21.** Let $a_1, a_2 \ldots, a_n$, and $b_1, b_2, \ldots, b_n$ be real numbers. Use the recursive definitions that you gave in Exercise 20 to prove these.

a) $\max(-a_1, -a_2, \ldots, -a_n) = -\min(a_1, a_2, \ldots, a_n)$
b) $\max(a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$
$\leq \max(a_1, a_2, \ldots, a_n) + \max(b_1, b_2, \ldots, b_n)$
c) $\min(a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$
$\geq \min(a_1, a_2, \ldots, a_n) + \min(b_1, b_2, \ldots, b_n)$

22. Show that the set S defined by $1 \in S$ and $s + t \in S$ whenever $s \in S$ and $t \in S$ is the set of positive integers.

23. Give a recursive definition of the set of positive integers that are multiples of 5.

24. Give a recursive definition of

a) the set of odd positive integers.
b) the set of positive integer powers of 3.
c) the set of polynomials with integer coefficients.

25. Give a recursive definition of

a) the set of even integers.
b) the set of positive integers congruent to 2 modulo 3.
c) the set of positive integers not divisible by 5.

26. Let S be the subset of the set of ordered pairs of integers defined recursively by

Basis step: $(0, 0) \in S$.

Recursive step: If $(a, b) \in S$, then $(a+2, b+3) \in S$ and $(a+3, b+2) \in S$.

a) List the elements of S produced by the first five applications of the recursive definition.
b) Use strong induction on the number of applications of the recursive step of the definition to show that $5 \mid a+b$ when $(a, b) \in S$.
c) Use structural induction to show that $5 \mid a+b$ when $(a, b) \in S$.

27. Let S be the subset of the set of ordered pairs of integers defined recursively by

Basis step: $(0, 0) \in S$.

Recursive step: If $(a, b) \in S$, then $(a, b+1) \in S$, $(a+1, b+1) \in S$, and $(a+2, b+1) \in S$.

a) List the elements of S produced by the first four applications of the recursive definition.
b) Use strong induction on the number of applications of the recursive step of the definition to show that $a \leq 2b$ whenever $(a, b) \in S$.
c) Use structural induction to show that $a \leq 2b$ whenever $(a, b) \in S$.

28. Give a recursive definition of each of these sets of ordered pairs of positive integers. (*Hint:* Plot the points in the set in the plane and look for lines containing points in the set.)

a) $S = \{(a, b) \mid a \in \mathbf{Z}^+, b \in \mathbf{Z}^+, \text{ and } a+b \text{ is odd}\}$
b) $S = \{(a, b) \mid a \in \mathbf{Z}^+, b \in \mathbf{Z}^+, \text{ and } a \mid b\}$
c) $S = \{(a, b) \mid a \in \mathbf{Z}^+, b \in \mathbf{Z}^+, \text{ and } 3 \mid a+b\}$

29. Give a recursive definition of each of these sets of ordered pairs of positive integers. Use structural induction to prove that the recursive definition you found is correct. (*Hint:* To find a recursive definition plot the points in the set in the plane and look for patterns.)

a) $S = \{(a, b) \mid a \in \mathbf{Z}^+, b \in \mathbf{Z}^+, \text{ and } a+b \text{ is even}\}$
b) $S = \{(a, b) \mid a \in \mathbf{Z}^+, b \in \mathbf{Z}^+, \text{ and } a \text{ or } b \text{ is odd}\}$
c) $S = \{(a, b) \mid a \in \mathbf{Z}^+, b \in \mathbf{Z}^+, \text{ and } a+b \text{ is odd and } 3 \mid b\}$

30. Prove that in a bit string, the string 01 occurs at most one more time than the string 10.

31. Define well-formed formulae of sets, variables representing sets, and operators from $\{\,\overline{\ }\,, \cup, \cap, -\}$.

32. a) Give a recursive definition of the function *ones*(s), which counts the number of ones in a bit string s.
b) Use structural induction to prove that *ones*(st) = *ones*(s) + *ones*(t).

33. a) Give a recursive definition of the function $m(s)$, which equals the smallest digit in a nonempty string of decimal digits.
b) Use structural induction to prove that $m(st) = \min(m(s), m(t))$.

The **reversal** of a string is the string consisting of the symbols of the string in reverse order. The reversal of the string w is denoted by w^R.

34. Find the reversal of the following bit strings.
a) 0101 **b)** 1 1011 **c)** 1000 1001 0111

35. Give a recursive definition of the reversal of a string. (*Hint:* First define the reversal of the empty string. Then write a string w of length $n+1$ as xy, where x is a string of length n, and express the reversal of w in terms of x^R and y.)

***36.** Use structural induction to prove that $(w_1 w_2)^R = w_2^R w_1^R$.

37. Give a recursive definition of w^i where w is a string and i is a nonnegative integer. (Here w^i represents the concatenation of i copies of the string w.)

***38.** Give a recursive definition of the set of bit strings that are palindromes.

39. When does a string belong to the set A of bit strings defined recursively by

$$\lambda \in A$$
$$0x1 \in A \text{ if } x \in A,$$

where λ is the empty string?

***40.** Recursively define the set of bit strings that have more zeros than ones.

41. Use Exercise 37 and mathematical induction to show that $l(w^i) = i \cdot l(w)$, where w is a string and i is a nonnegative integer.

***42.** Show that $(w^R)^i = (w^i)^R$ whenever w is a string and i is a nonnegative integer; that is, show that the ith power of the reversal of a string is the reversal of the ith power of the string.

43. Use structural induction to show that $n(T) \geq 2h(T) + 1$, where T is a full binary tree, $n(T)$ equals the number of vertices of T, and $h(T)$ is the height of T.

The set of leaves and the set of internal vertices of a full binary tree can be defined recursively.

Basis step: The root r is a leaf of the full binary tree with exactly one vertex r. This tree has no internal vertices.

Recursive step: The set of leaves of the tree $T = T_1 \cdot T_2$ is the union of the set of leaves of T_1 and the set of leaves of T_2. The internal vertices of T are the root r of T and the union of the set of internal vertices of T_1 and the set of internal vertices of T_2.

44. Use structural induction to show that $l(T)$, the number of leaves of a full binary tree T, is 1 more than $i(T)$, the number of internal vertices of T.

45. Use generalized induction as was done in Example 15 to show that if $a_{m,n}$ is defined recursively by $a_{0,0} = 0$ and

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\ a_{m,n-1} + 1 & \text{if } n > 0, \end{cases}$$

then $a_{m,n} = m + n$ for all $(m, n) \in \mathbf{N} \times \mathbf{N}$.

46. Use generalized induction as was done in Example 15 to show that if $a_{m,n}$ is defined recursively by $a_{1,1} = 5$ and

$$a_{m,n} = \begin{cases} a_{m-1,n} + 2 & \text{if } n = 1 \text{ and } m > 1 \\ a_{m,n-1} + 2 & \text{if } n > 1, \end{cases}$$

then $a_{m,n} = 2(m+n) + 1$ for all $(m, n) \in \mathbf{Z}^+ \times \mathbf{Z}^+$.

***47.** A **partition** of a positive integer n is a way to write n as a sum of positive integers. For instance, $7 = 3 + 2 + 1 + 1$ is a partition of 7. Let P_m equal the number of different partitions of m, where the order of terms in the sum does not matter, and let $P_{m,n}$ be the number of different ways to express m as the sum of positive integers not exceeding n.

a) Show that $P_{m,m} = P_m$.

b) Show that the following recursive definition for $P_{m,n}$ is correct:

$$P_{m,n} = \begin{cases} 1 & \text{if } m = 1 \\ 1 & \text{if } n = 1 \\ P_{m,m} & \text{if } m < n \\ 1 + P_{m,m-1} & \text{if } m = n > 1 \\ P_{m,n-1} + P_{m-n,n} & \text{if } m > n > 1. \end{cases}$$

c) Find the number of partitions of 5 and of 6 using this recursive definition.

Consider an inductive definition of a version of **Ackermann's function.** This function was named after Wilhelm Ackermann, a German mathematician who was a student of the great mathematician David Hilbert. Ackermann's function plays an important role in the theory of recursive functions and in the study of the complexity of certain algorithms involving set unions. (There are several different variants of this function. All are called Ackermann's function and have similar properties even though their values do not always agree.)

$$A(m, n) = \begin{cases} 2n & \text{if } m = 0 \\ 0 & \text{if } m \ge 1 \text{ and } n = 0 \\ 2 & \text{if } m \ge 1 \text{ and } n = 1 \\ A(m-1, A(m, n-1)) & \\ & \text{if } m \ge 1 \text{ and } n \ge 2 \end{cases}$$

Exercises 48–55 involve this version of Ackermann's function.

48. Find these values of Ackermann's function.

a) $A(1, 0)$ **b)** $A(0, 1)$
c) $A(1, 1)$ **d)** $A(2, 2)$

49. Show that $A(m, 2) = 4$ whenever $m \ge 1$.

50. Show that $A(1, n) = 2^n$ whenever $n \ge 1$.

51. Find these values of Ackermann's function.

a) $A(2, 3)$ ***b)** $A(3, 3)$

***52.** Find $A(3, 4)$.

****53.** Prove that $A(m, n+1) > A(m, n)$ whenever m and n are nonnegative integers.

***54.** Prove that $A(m+1, n) \ge A(m, n)$ whenever m and n are nonnegative integers.

55. Prove that $A(i, j) \ge j$ whenever i and j are nonnegative integers.

56. Use mathematical induction to prove that a function F defined by specifying $F(0)$ and a rule for obtaining $F(n+1)$ from $F(n)$ is well defined.

57. Use strong induction to prove that a function F defined by specifying $F(0)$ and a rule for obtaining $F(n+1)$ from the values $F(k)$ for $k = 0, 1, 2, \ldots, n$ is well defined.

58. Show that each of these proposed recursive definitions of a function on the set of positive integers does not produce a well-defined function.

a) $F(n) = 1 + F(\lfloor n/2 \rfloor)$ for $n \ge 1$ and $F(1) = 1$.

b) $F(n) = 1 + F(n-3)$ for $n \ge 2$, $F(1) = 2$, and $F(2) = 3$.

c) $F(n) = 1 + F(n/2)$ for $n \ge 2$, $F(1) = 1$, and $F(2) = 2$.

d) $F(n) = 1 + F(n/2)$ if n is even and $n \ge 2$, $F(n) = 1 - F(n-1)$ if n is odd, and $F(1) = 1$.

e) $F(n) = 1 + F(n/2)$ if n is even and $n \ge 2$, $F(n) = F(3n-1)$ if n is odd and $n \ge 3$, and $F(1) = 1$.

59. Show that each of these proposed recursive definitions of a function on the set of positive integers does not produce a well-defined function.

a) $F(n) = 1 + F(\lfloor (n+1)/2 \rfloor)$ for $n \ge 1$ and $F(1) = 1$.

b) $F(n) = 1 + F(n-2)$ for $n \ge 2$ and $F(1) = 0$.

c) $F(n) = 1 + F(n/3)$ for $n \ge 3$, $F(1) = 1$, $F(2) = 2$, and $F(3) = 3$.

d) $F(n) = 1 + F(n/2)$ if n is even and $n \ge 2$, $F(n) = 1 + F(n-2)$ if n is odd, and $F(1) = 1$.

e) $F(n) = 1 + F(F(n-1))$ if $n \ge 2$ and $F(1) = 2$.

Exercises 60–62 deal with iterations of the logarithm function. Let $\log n$ denote the logarithm of n to the base 2, as usual. The function $\log^{(k)} n$ is defined recursively by

$$\log^{(k)} n = \begin{cases} n & \text{if } k = 0 \\ \log(\log^{(k-1)} n) & \text{if } \log^{(k-1)} n \text{ is defined} \\ & \text{and positive} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The **iterated logarithm** is the function $\log^* n$ whose value at n is the smallest nonnegative integer k such that $\log^{(k)} n \le 1$.

60. Find each of these values:

a) $\log^{(2)} 16$

b) $\log^{(3)} 256$

c) $\log^{(3)} 2^{65536}$

d) $\log^{(4)} 2^{2^{65536}}$

61. Find the value of $\log^* n$ for each of these values of n:

a) 2 **b)** 4 **c)** 8 **d)** 16
e) 256 **f)** 65536 **g)** 2^{2048}

62. Find the largest integer n such that $\log^* n = 5$. Determine the number of decimal digits in this number.

Exercises 63–65 deal with values of iterated functions. Suppose that $f(n)$ is a function from the set of real numbers, or positive real numbers, or some other set of real numbers, to the set of real numbers such that $f(n)$ is monotonically increasing [that is, $f(n) < f(m)$ when $n < m$) and $f(n) < n$ for all n in the domain of f.] The function $f^{(k)}(n)$ is defined recursively by

$$f^{(k)}(n) = \begin{cases} n & \text{if } k = 0 \\ f(f^{(k-1)}(n)) & \text{if } k > 0. \end{cases}$$

Furthermore, let c be a positive real number. The **iterated function** f_c^* is the number of iterations of f required to reduce its argument to c or less, so that $f_c^*(n)$ is the smallest nonnegative integer k such that $f^k(n) \leq c$.

63. Let $f(n) = n - a$, where a is a positive integer. Find a formula for $f^{(k)}(n)$. What is the value of $f_0^*(n)$ when n is a positive integer?

64. Let $f(n) = n/2$. Find a formula for $f^{(k)}(n)$. What is the value of $f_1^*(n)$ when n is a positive integer?

65. Let $f(n) = \sqrt{n}$. Find a formula for $f^{(k)}(n)$. What is the value of $f_2^*(n)$ when n is a positive integer?

3.5 Recursive Algorithms

INTRODUCTION

Sometimes we can reduce the solution to a problem with a particular set of input to the solution of the same problem with smaller input values. For instance, the problem of finding the greatest common divisor of two positive integers a and b where $b > a$ can be reduced to finding the greatest common divisor of a pair of smaller integers, namely, b **mod** a and a, since $\gcd(b \textbf{ mod } a, a) = \gcd(a, b)$. When such a reduction can be done, the solution to the original problem can be found with a sequence of reductions, until the problem has been reduced to some initial case for which the solution is known. For instance, for finding the greatest common divisor, the reduction continues until the smaller of the two numbers is zero, since $\gcd(a, 0) = a$ when $a > 0$.

We will see that algorithms that successively reduce a problem to the same problem with smaller input are used to solve a wide variety of problems.

DEFINITION 1 An algorithm is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input.

Links

We will describe several different recursive algorithms in Examples 1, 2, 4, 5, and 6. The first example shows how a recursive algorithm can be constructed to evaluate a function from its recursive definition.

EXAMPLE 1 Give a recursive algorithm for computing a^n where a is a nonzero real number and n is a nonnegative integer.

Solution: We can base a recursive algorithm on the recursive definition of a^n. This definition states that $a^{n+1} = a \cdot a^n$ for $n > 0$ and the initial condition $a^0 = 1$. To find a^n, successively use the recursive condition to reduce the exponent until it becomes zero. We give this procedure in Algorithm 1. ◀