0

INTRODUCTION

We begin with an overview of those areas in the theory of computation that we present in this course. Following that, you'll have a chance to learn and/or review some mathematical concepts that you will need later.

0.1

AUTOMATA, COMPUTABILITY, AND COMPLEXITY

This book focuses on three traditionally central areas of the theory of computation: automata, computability, and complexity. They are linked by the question:

What are the fundamental capabilities and limitations of computers?

This question goes back to the 1930s when mathematical logicians first began to explore the meaning of computation. Technological advances since that time have greatly increased our ability to compute and have brought this question out of the realm of theory into the world of practical concern.

In each of the three areas—automata, computability, and complexity—this question is interpreted differently, and the answers vary according to the interpretation. Following this introductory chapter, we explore each area in a separate part of this book. Here, we introduce these parts in reverse order because starting from the end you can better understand the reason for the beginning.

COMPLEXITY THEORY

Computer problems come in different varieties; some are easy, and some are hard. For example, the sorting problem is an easy one. Say that you need to arrange a list of numbers in ascending order. Even a small computer can sort a million numbers rather quickly. Compare that to a scheduling problem. Say that you must find a schedule of classes for the entire university to satisfy some reasonable constraints, such as that no two classes take place in the same room at the same time. The scheduling problem seems to be much harder than the sorting problem. If you have just a thousand classes, finding the best schedule may require centuries, even with a supercomputer.

What makes some problems computationally hard and others easy?

This is the central question of complexity theory. Remarkably, we don't know the answer to it, though it has been intensively researched for the past 35 years. Later, we explore this fascinating question and some of its ramifications.

In one of the important achievements of complexity theory thus far, researchers have discovered an elegant scheme for classifying problems according to their computational difficulty. It is analogous to the periodic table for classifying elements according to their chemical properties. Using this scheme, we can demonstrate a method for giving evidence that certain problems are computationally hard, even if we are unable to prove that they are.

You have several options when you confront a problem that appears to be computationally hard. First, by understanding which aspect of the problem is at the root of the difficulty, you may be able to alter it so that the problem is more easily solvable. Second, you may be able to settle for less than a perfect solution to the problem. In certain cases finding solutions that only approximate the perfect one is relatively easy. Third, some problems are hard only in the worst case situation, but easy most of the time. Depending on the application, you may be satisfied with a procedure that occasionally is slow but usually runs quickly. Finally, you may consider alternative types of computation, such as randomized computation, that can speed up certain tasks.

One applied area that has been affected directly by complexity theory is the ancient field of cryptography. In most fields, an easy computational problem is preferable to a hard one because easy ones are cheaper to solve. Cryptography is unusual because it specifically requires computational problems that are hard, rather than easy, because secret codes should be hard to break without the secret key or password. Complexity theory has pointed cryptographers in the direction of computationally hard problems around which they have designed revolutionary new codes.

COMPUTABILITY THEORY

During the first half of the twentieth century, mathematicians such as Kurt Gödel, Alan Turing, and Alonzo Church discovered that certain basic problems cannot be solved by computers. One example of this phenomenon is the problem of determining whether a mathematical statement is true or false. This task is the bread and butter of mathematicians. It seems like a natural for solution by computer because it lies strictly within the realm of mathematics. But no computer algorithm can perform this task.

Among the consequences of this profound result was the development of ideas concerning theoretical models of computers that eventually would help lead to the construction of actual computers.

The theories of computability and complexity are closely related. In complexity theory, the objective is to classify problems as easy ones and hard ones, whereas in computability theory the classification of problems is by those that are solvable and those that are not. Computability theory introduces several of the concepts used in complexity theory.

AUTOMATA THEORY

Automata theory deals with the definitions and properties of mathematical models of computation. These models play a role in several applied areas of computer science. One model, called the *finite automaton*, is used in text processing, compilers, and hardware design. Another model, called the *context-free grammar*, is used in programming languages and artificial intelligence.

Automata theory is an excellent place to begin the study of the theory of computation. The theories of computability and complexity require a precise definition of a *computer*. Automata theory allows practice with formal definitions of computation as it introduces concepts relevant to other nontheoretical areas of computer science.

0.2 MATHEMATICAL NOTIONS AND TERMINOLOGY

As in any mathematical subject, we begin with a discussion of the basic mathematical objects, tools, and notation that we expect to use.

SETS

A ***set*** is a group of objects represented as a unit. Sets may contain any type of object, including numbers, symbols, and even other sets. The objects in a set are called its ***elements*** or ***members***. Sets may be described formally in several ways. One way is by listing a set's elements inside braces. Thus the set

$$\{7, 21, 57\}$$

contains the elements 7, 21, and 57. The symbols $\in$ and $\notin$ denote set membership and nonmembership. We write $7 \in \{7, 21, 57\}$ and $8 \notin \{7, 21, 57\}$. For two sets A and B, we say that A is a ***subset*** of B, written $A \subseteq B$, if every member of

A also is a member of B. We say that A is a ***proper subset*** of B, written $A \subsetneq B$, if A is a subset of B and not equal to B.

The order of describing a set doesn't matter, nor does repetition of its members. We get the same set by writing $\{57, 7, 7, 7, 21\}$. If we do want to take the number of occurrences of members into account we call the group a ***multiset*** instead of a set. Thus $\{7\}$ and $\{7, 7\}$ are different as multisets but identical as sets. An ***infinite set*** contains infinitely many elements. We cannot write a list of all the elements of an infinite set, so we sometimes use the "..." notation to mean, "continue the sequence forever." Thus we write the set of ***natural numbers*** $\mathcal{N}$ as

$$\{1, 2, 3, \ldots\}.$$

The set of ***integers*** $\mathcal{Z}$ is written

$$\{\ldots, -2, -1, 0, 1, 2, \ldots\}.$$

The set with 0 members is called the ***empty set*** and is written $\emptyset$.

When we want to describe a set containing elements according to some rule, we write $\{n|\, \text{rule about } n\}$. Thus $\{n|\, n = m^2 \text{ for some } m \in \mathcal{N}\}$ means the set of perfect squares.

If we have two sets A and B, the ***union*** of A and B, written $A \cup B$, is the set we get by combining all the elements in A and B into a single set. The ***intersection*** of A and B, written $A \cap B$, is the set of elements that are in both A and B. The ***complement*** of A, written $\overline{A}$, is the set of all elements under consideration that are *not* in A.

As is often the case in mathematics, a picture helps clarify a concept. For sets, we use a type of picture called a ***Venn diagram***. It represents sets as regions enclosed by circular lines. Let the set START-t be the set of all English words that start with the letter "t." For example, in the following figure the circle represents the set START-t. Several members of this set are represented as points inside the circle.

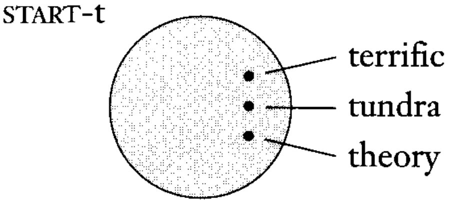


FIGURE 0.1
Venn diagram for the set of English words starting with "t"

Similarly, we represent the set END-z of English words that end with "z" in the following figure.

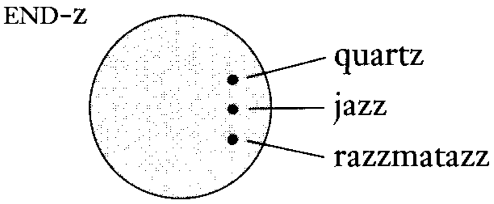


FIGURE 0.2
Venn diagram for the set of English words ending with "z"

To represent both sets in the same Venn diagram we must draw them so that they overlap, indicating that they share some elements, as shown in the following figure. For example, the word *topaz* is in both sets. The figure also contains a circle for the set START-j. It doesn't overlap the circle for START-t because no word lies in both sets.

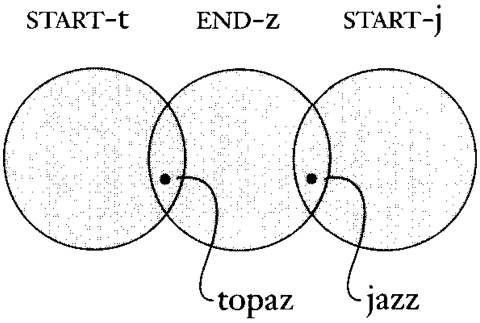


FIGURE 0.3
Overlapping circles indicate common elements

The next two Venn diagrams depict the union and intersection of sets A and B.

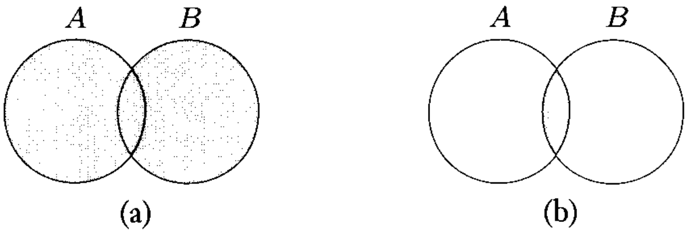


FIGURE 0.4
Diagrams for (a) $A \cup B$ and (b) $A \cap B$

SEQUENCES AND TUPLES

A ***sequence*** of objects is a list of these objects in some order. We usually designate a sequence by writing the list within parentheses. For example, the sequence 7, 21, 57 would be written

$$(7, 21, 57).$$

In a set the order doesn't matter, but in a sequence it does. Hence $(7, 21, 57)$ is not the same as $(57, 7, 21)$. Similarly, repetition does matter in a sequence, but it doesn't matter in a set. Thus $(7, 7, 21, 57)$ is different from both of the other sequences, whereas the set $\{7, 21, 57\}$ is identical to the set $\{7, 7, 21, 57\}$.

As with sets, sequences may be finite or infinite. Finite sequences often are called ***tuples***. A sequence with k elements is a ***k-tuple***. Thus $(7, 21, 57)$ is a 3-tuple. A 2-tuple is also called a ***pair***.

Sets and sequences may appear as elements of other sets and sequences. For example, the ***power set*** of A is the set of all subsets of A. If A is the set $\{0, 1\}$, the power set of A is the set $\{\, \emptyset, \{0\}, \{1\}, \{0, 1\}\, \}$. The set of all pairs whose elements are 0s and 1s is $\{\, (0, 0), (0, 1), (1, 0), (1, 1)\, \}$.

If A and B are two sets, the ***Cartesian product*** or ***cross product*** of A and B, written $A \times B$, is the set of all pairs wherein the first element is a member of A and the second element is a member of B.

EXAMPLE 0.5

If $A = \{1, 2\}$ and $B = \{x, y, z\}$,

$$A \times B = \{\, (1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\, \}.$$

We can also take the Cartesian product of k sets, $A_1, A_2, \ldots, A_k$, written $A_1 \times A_2 \times \cdots \times A_k$. It is the set consisting of all k-tuples $(a_1, a_2, \ldots, a_k)$ where $a_i \in A_i$.

EXAMPLE 0.6

If A and B are as in Example 0.5,

$$\begin{aligned} A \times B \times A = \{\, & (1, x, 1), (1, x, 2), (1, y, 1), (1, y, 2), (1, z, 1), (1, z, 2), \\ & (2, x, 1), (2, x, 2), (2, y, 1), (2, y, 2), (2, z, 1), (2, z, 2)\, \}. \end{aligned}$$

If we have the Cartesian product of a set with itself, we use the shorthand

$$\overbrace{A \times A \times \cdots \times A}^{k} = A^k.$$

EXAMPLE 0.7

The set $\mathcal{N}^2$ equals $\mathcal{N} \times \mathcal{N}$. It consists of all pairs of natural numbers. We also may write it as $\{(i,j) \mid i,j \geq 1\}$.

FUNCTIONS AND RELATIONS

Functions are central to mathematics. A ***function*** is an object that sets up an input–output relationship. A function takes an input and produces an output. In every function, the same input always produces the same output. If f is a function whose output value is b when the input value is a, we write

$$f(a) = b.$$

A function also is called a ***mapping***, and, if $f(a) = b$, we say that f maps a to b.

For example, the absolute value function abs takes a number x as input and returns x if x is positive and $-x$ if x is negative. Thus $abs(2) = abs(-2) = 2$. Addition is another example of a function, written add. The input to the addition function is a pair of numbers, and the output is the sum of those numbers.

The set of possible inputs to the function is called its ***domain***. The outputs of a function come from a set called its ***range***. The notation for saying that f is a function with domain D and range R is

$$f\colon D \longrightarrow R.$$

In the case of the function abs, if we are working with integers, the domain and the range are $\mathcal{Z}$, so we write $abs\colon \mathcal{Z} \longrightarrow \mathcal{Z}$. In the case of the addition function for integers, the domain is the set of pairs of integers $\mathcal{Z} \times \mathcal{Z}$ and the range is $\mathcal{Z}$, so we write $add\colon \mathcal{Z} \times \mathcal{Z} \longrightarrow \mathcal{Z}$. Note that a function may not necessarily use all the elements of the specified range. The function abs never takes on the value -1 even though $-1 \in \mathcal{Z}$. A function that does use all the elements of the range is said to be ***onto*** the range.

We may describe a specific function in several ways. One way is with a procedure for computing an output from a specified input. Another way is with a table that lists all possible inputs and gives the output for each input.

EXAMPLE 0.8

Consider the function $f\colon \{0,1,2,3,4\} \longrightarrow \{0,1,2,3,4\}$.

n	$f(n)$
0	1
1	2
2	3
3	4
4	0

This function adds 1 to its input and then outputs the result modulo 5. A number modulo m is the remainder after division by m. For example, the minute hand on a clock face counts modulo 60. When we do modular arithmetic we define $\mathcal{Z}_m = \{0, 1, 2, \ldots, m-1\}$. With this notation, the aforementioned function f has the form $f\colon \mathcal{Z}_5 \longrightarrow \mathcal{Z}_5$.

EXAMPLE 0.9

Sometimes a two-dimensional table is used if the domain of the function is the Cartesian product of two sets. Here is another function, $g\colon \mathcal{Z}_4 \times \mathcal{Z}_4 \longrightarrow \mathcal{Z}_4$. The entry at the row labeled i and the column labeled j in the table is the value of $g(i, j)$.

g	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

The function g is the addition function modulo 4.

When the domain of a function f is $A_1 \times \cdots \times A_k$ for some sets $A_1, \ldots, A_k$, the input to f is a k-tuple $(a_1, a_2, \ldots, a_k)$ and we call the a_i the ***arguments*** to f. A function with k arguments is called a ***k-ary function***, and k is called the ***arity*** of the function. If k is 1, f has a single argument and f is called a ***unary function***. If k is 2, f is a ***binary function***. Certain familiar binary functions are written in a special ***infix notation***, with the symbol for the function placed between its two arguments, rather than in ***prefix notation***, with the symbol preceding. For example, the addition function add usually is written in infix notation with the $+$ symbol between its two arguments as in $a + b$ instead of in prefix notation $add(a, b)$.

A ***predicate*** or ***property*** is a function whose range is $\{\text{TRUE}, \text{FALSE}\}$. For example, let $even$ be a property that is TRUE if its input is an even number and FALSE if its input is an odd number. Thus $even(4) = \text{TRUE}$ and $even(5) = \text{FALSE}$.

A property whose domain is a set of k-tuples $A \times \cdots \times A$ is called a ***relation***, a ***k-ary relation***, or a ***k-ary relation on A***. A common case is a 2-ary relation, called a ***binary relation***. When writing an expression involving a binary relation, we customarily use infix notation. For example, "less than" is a relation usually written with the infix operation symbol $<$. "Equality," written with the $=$ symbol is another familiar relation. If R is a binary relation, the statement aRb means that $aRb = \text{TRUE}$. Similarly if R is a k-ary relation, the statement $R(a_1, \ldots, a_k)$ means that $R(a_1, \ldots, a_k) = \text{TRUE}$.

EXAMPLE 0.10

In a children's game called Scissors–Paper–Stone, the two players simultaneously select a member of the set {SCISSORS, PAPER, STONE} and indicate their selections with hand signals. If the two selections are the same, the game starts over. If the selections differ, one player wins, according to the relation *beats*.

beats	SCISSORS	PAPER	STONE
SCISSORS	FALSE	TRUE	FALSE
PAPER	FALSE	FALSE	TRUE
STONE	TRUE	FALSE	FALSE

From this table we determine that SCISSORS *beats* PAPER is TRUE and that PAPER *beats* SCISSORS is FALSE.

Sometimes describing predicates with sets instead of functions is more convenient. The predicate $P\colon D \longrightarrow \{\text{TRUE}, \text{FALSE}\}$ may be written (D, S), where $S = \{a \in D|\ P(a) = \text{TRUE}\}$, or simply S if the domain D is obvious from the context. Hence the relation *beats* may be written

$$\{(\text{SCISSORS, PAPER}), (\text{PAPER, STONE}), (\text{STONE, SCISSORS})\}.$$

A special type of binary relation, called an ***equivalence relation***, captures the notion of two objects being equal in some feature. A binary relation R is an equivalence relation if R satisfies three conditions:

1. R is ***reflexive*** if for every x, xRx;
2. R is ***symmetric*** if for every x and y, xRy implies yRx; and
3. R is ***transitive*** if for every x, y, and z, xRy and yRz implies xRz.

EXAMPLE 0.11

Define an equivalence relation on the natural numbers, written $\equiv_7$. For $i, j \in \mathcal{N}$ say that $i \equiv_7 j$, if $i-j$ is a multiple of 7. This is an equivalence relation because it satisfies the three conditions. First, it is reflexive, as $i - i = 0$, which is a multiple of 7. Second, it is symmetric, as $i - j$ is a multiple of 7 if $j - i$ is a multiple of 7. Third, it is transitive, as whenever $i - j$ is a multiple of 7 and $j - k$ is a multiple of 7, then $i - k = (i - j) + (j - k)$ is the sum of two multiples of 7 and hence a multiple of 7, too.

GRAPHS

An ***undirected graph***, or simply a ***graph***, is a set of points with lines connecting some of the points. The points are called ***nodes*** or ***vertices***, and the lines are called ***edges***, as shown in the following figure.

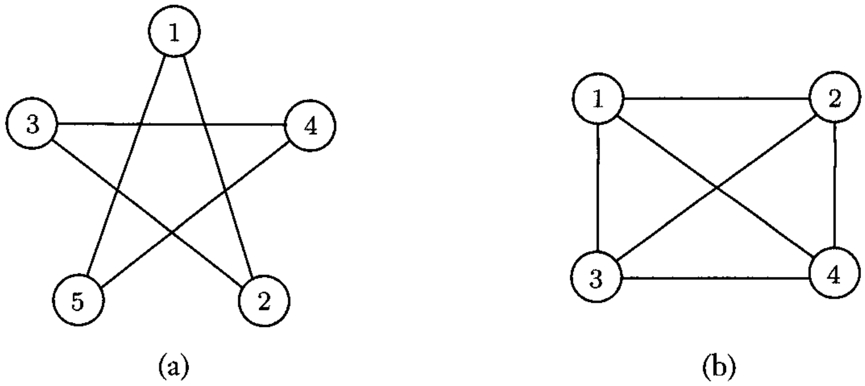


FIGURE 0.12
Examples of graphs

The number of edges at a particular node is the ***degree*** of that node. In Figure 0.12(a) all the nodes have degree 2. In Figure 0.12(b) all the nodes have degree 3. No more than one edge is allowed between any two nodes.

In a graph G that contains nodes i and j, the pair (i, j) represents the edge that connects i and j. The order of i and j doesn't matter in an undirected graph, so the pairs (i, j) and (j, i) represent the same edge. Sometimes we describe edges with sets, as in $\{i, j\}$, instead of pairs if the order of the nodes is unimportant. If V is the set of nodes of G and E is the set of edges, we say $G = (V, E)$. We can describe a graph with a diagram or more formally by specifying V and E. For example, a formal description of the graph in Figure 0.12(a) is

$$(\{1,2,3,4,5\},\ \{(1,2),\ (2,3),\ (3,4),\ (4,5),\ (5,1)\}),$$

and a formal description of the graph in Figure 0.12(b) is

$$(\{1,2,3,4\},\ \{(1,2),\ (1,3),\ (1,4),\ (2,3),\ (2,4),\ (3,4)\}).$$

Graphs frequently are used to represent data. Nodes might be cities and edges the connecting highways, or nodes might be electrical components and edges the wires between them. Sometimes, for convenience, we label the nodes and/or edges of a graph, which then is called a ***labeled graph***. Figure 0.13 depicts a graph whose nodes are cities and whose edges are labeled with the dollar cost of the cheapest nonstop air fare for travel between those cities if flying nonstop between them is possible.

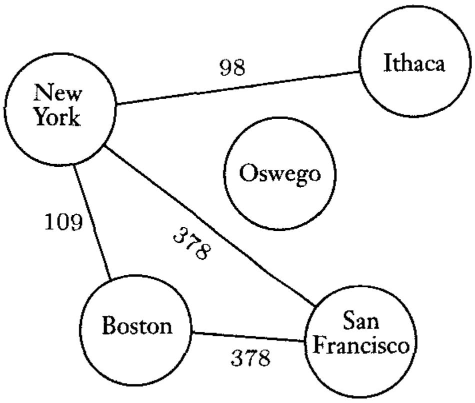


FIGURE 0.13
Cheapest nonstop air fares between various cities

We say that graph G is a ***subgraph*** of graph H if the nodes of G are a subset of the nodes of H, and the edges of G are the edges of H on the corresponding nodes. The following figure shows a graph H and a subgraph G.

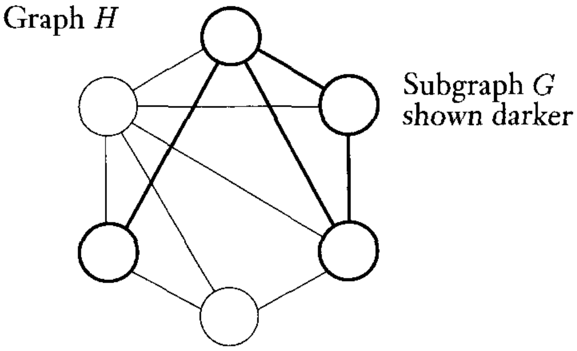


FIGURE 0.14
Graph G (shown darker) is a subgraph of H

A ***path*** in a graph is a sequence of nodes connected by edges. A ***simple path*** is a path that doesn't repeat any nodes. A graph is ***connected*** if every two nodes have a path between them. A path is a ***cycle*** if it starts and ends in the same node. A ***simple cycle*** is one that contains at least three nodes and repeats only the first and last nodes. A graph is a ***tree*** if it is connected and has no simple cycles, as shown in Figure 0.15. A tree may contain a specially designated node called the ***root***. The nodes of degree 1 in a tree, other than the root, are called the ***leaves*** of the tree.

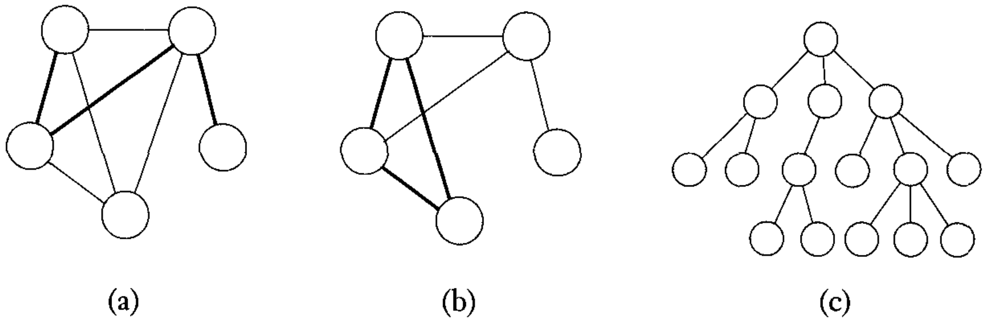


FIGURE 0.15
(a) A path in a graph, (b) a cycle in a graph, and (c) a tree

If it has arrows instead of lines, the graph is a ***directed graph***, as shown in the following figure. The number of arrows pointing from a particular node is the ***outdegree*** of that node, and the number of arrows pointing to a particular node is the ***indegree***.

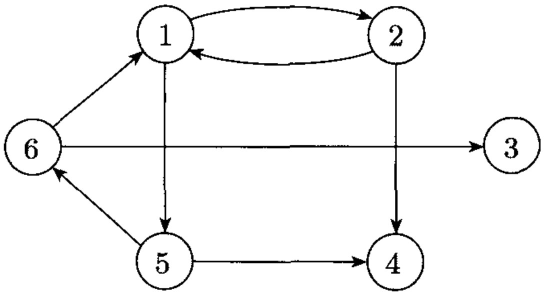


FIGURE 0.16
A directed graph

In a directed graph we represent an edge from i to j as a pair (i, j). The formal description of a directed graph G is (V, E) where V is the set of nodes and E is the set of edges. The formal description of the graph in Figure 0.16 is

$$(\{1,2,3,4,5,6\},\ \{(1,2), (1,5), (2,1), (2,4), (5,4), (5,6), (6,1), (6,3)\}).$$

A path in which all the arrows point in the same direction as its steps is called a ***directed path***. A directed graph is ***strongly connected*** if a directed path connects every two nodes.

EXAMPLE 0.17

The directed graph shown here represents the relation given in Example 0.10.

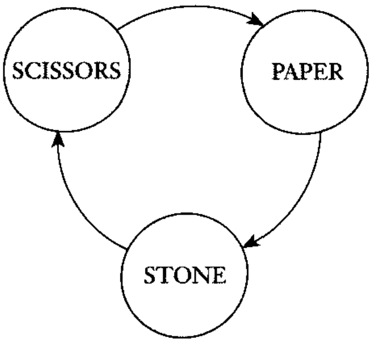


FIGURE 0.18
The graph of the relation *beats*

Directed graphs are a handy way of depicting binary relations. If R is a binary relation whose domain is $D \times D$, a labeled graph $G = (D, E)$ represents R, where $E = \{(x, y) | \, xRy\}$. Figure 0.18 illustrates this representation.

If V is the set of nodes and E is the set of edges, the notation for a graph G consisting of these nodes and edges is $G = (V, E)$.

STRINGS AND LANGUAGES

Strings of characters are fundamental building blocks in computer science. The alphabet over which the strings are defined may vary with the application. For our purposes, we define an ***alphabet*** to be any nonempty finite set. The members of the alphabet are the ***symbols*** of the alphabet. We generally use capital Greek letters Σ and Γ to designate alphabets and a typewriter font for symbols from an alphabet. The following are a few examples of alphabets.

$$\Sigma_1 = \{\texttt{0,1}\};$$

$$\Sigma_2 = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}, \texttt{g}, \texttt{h}, \texttt{i}, \texttt{j}, \texttt{k}, \texttt{l}, \texttt{m}, \texttt{n}, \texttt{o}, \texttt{p}, \texttt{q}, \texttt{r}, \texttt{s}, \texttt{t}, \texttt{u}, \texttt{v}, \texttt{w}, \texttt{x}, \texttt{y}, \texttt{z}\};$$

$$\Gamma = \{\texttt{0}, \texttt{1}, \texttt{x}, \texttt{y}, \texttt{z}\}.$$

A ***string over an alphabet*** is a finite sequence of symbols from that alphabet, usually written next to one another and not separated by commas. If $\Sigma_1 = \{\texttt{0,1}\}$, then 01001 is a string over Σ_1. If $\Sigma_2 = \{\texttt{a}, \texttt{b}, \texttt{c}, \ldots, \texttt{z}\}$, then abracadabra is a string over Σ_2. If w is a string over Σ, the ***length*** of w, written $|w|$, is the number of symbols that it contains. The string of length zero is called the ***empty string*** and is written ε. The empty string plays the role of 0 in a number system. If w

has length n, we can write $w = w_1w_2 \cdots w_n$ where each $w_i \in \Sigma$. The ***reverse*** of w, written $w^{\mathcal{R}}$, is the string obtained by writing w in the opposite order (i.e., $w_nw_{n-1} \cdots w_1$). String z is a ***substring*** of w if z appears consecutively within w. For example, cad is a substring of abracadabra.

If we have string x of length m and string y of length n, the ***concatenation*** of x and y, written xy, is the string obtained by appending y to the end of x, as in $x_1 \cdots x_my_1 \cdots y_n$. To concatenate a string with itself many times we use the superscript notation

$$\overbrace{xx\cdots x}^{k} = x^k.$$

The ***lexicographic ordering*** of strings is the same as the familiar dictionary ordering, except that shorter strings precede longer strings. Thus the lexicographic ordering of all strings over the alphabet {0,1} is

$$(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots).$$

A ***language*** is a set of strings.

BOOLEAN LOGIC

Boolean logic is a mathematical system built around the two values TRUE and FALSE. Though originally conceived of as pure mathematics, this system is now considered to be the foundation of digital electronics and computer design. The values TRUE and FALSE are called the ***Boolean values*** and are often represented by the values 1 and 0. We use Boolean values in situations with two possibilities, such as a wire that may have a high or a low voltage, a proposition that may be true or false, or a question that may be answered yes or no.

We can manipulate Boolean values with specially designed operations, called the ***Boolean operations***. The simplest such operation is the ***negation*** or **NOT** operation, designated with the symbol $\neg$. The negation of a Boolean value is the opposite value. Thus $\neg 0 = 1$ and $\neg 1 = 0$. The ***conjunction***, or **AND**, operation is designated with the symbol $\wedge$. The conjunction of two Boolean values is 1 if both of those values are 1. The ***disjunction***, or **OR**, operation is designated with the symbol $\vee$. The disjunction of two Boolean values is 1 if either of those values is 1. We summarize this information as follows.

$0 \wedge 0 = 0$	$0 \vee 0 = 0$	$\neg 0 = 1$
$0 \wedge 1 = 0$	$0 \vee 1 = 1$	$\neg 1 = 0$
$1 \wedge 0 = 0$	$1 \vee 0 = 1$	
$1 \wedge 1 = 1$	$1 \vee 1 = 1$	

We use Boolean operations for combining simple statements into more complex Boolean expressions, just as we use the arithmetic operations + and × to construct complex arithmetic expressions. For example, if P is the Boolean value representing the truth of the statement "the sun is shining" and Q represents the truth of the statement "today is Monday", we may write $P \wedge Q$ to represent the truth value of the statement "the sun is shining *and* today is Monday" and similarly for $P \vee Q$ with *and* replaced by *or*. The values P and Q are called the ***operands*** of the operation.

Several other Boolean operations occasionally appear. The ***exclusive or***, or **XOR**, operation is designated by the $\oplus$ symbol and is 1 if either but not both of its two operands are 1. The ***equality*** operation, written with the symbol $\leftrightarrow$, is 1 if both of its operands have the same value. Finally, the ***implication*** operation is designated by the symbol $\rightarrow$ and is 0 if its first operand is 1 and its second operand is 0; otherwise $\rightarrow$ is 1. We summarize this information as follows.

$$\begin{array}{lll}
0 \oplus 0 = 0 & \qquad 0 \leftrightarrow 0 = 1 & \qquad 0 \rightarrow 0 = 1 \\
0 \oplus 1 = 1 & \qquad 0 \leftrightarrow 1 = 0 & \qquad 0 \rightarrow 1 = 1 \\
1 \oplus 0 = 1 & \qquad 1 \leftrightarrow 0 = 0 & \qquad 1 \rightarrow 0 = 0 \\
1 \oplus 1 = 0 & \qquad 1 \leftrightarrow 1 = 1 & \qquad 1 \rightarrow 1 = 1
\end{array}$$

We can establish various relationships among these operations. In fact, we can express all Boolean operations in terms of the AND and NOT operations, as the following identities show. The two expressions in each row are equivalent. Each row expresses the operation in the left-hand column in terms of operations above it and AND and NOT.

$$\begin{array}{ll}
P \vee Q & \qquad \neg(\neg P \wedge \neg Q) \\
P \rightarrow Q & \qquad \neg P \vee Q \\
P \leftrightarrow Q & \qquad (P \rightarrow Q) \wedge (Q \rightarrow P) \\
P \oplus Q & \qquad \neg(P \leftrightarrow Q)
\end{array}$$

The ***distributive law*** for AND and OR comes in handy in manipulating Boolean expressions. It is similar to the distributive law for addition and multiplication, which states that $a \times (b + c) = (a \times b) + (a \times c)$. The Boolean version comes in two forms:

- $P \wedge (Q \vee R)$ equals $(P \wedge Q) \vee (P \wedge R)$, and its dual

- $P \vee (Q \wedge R)$ equals $(P \vee Q) \wedge (P \vee R)$.

Note that the dual of the distributive law for addition and multiplication does not hold in general.

SUMMARY OF MATHEMATICAL TERMS

Alphabet	A finite set of objects called symbols
Argument	An input to a function
Binary relation	A relation whose domain is a set of pairs
Boolean operation	An operation on Boolean values
Boolean value	The values TRUE or FALSE, often represented by 1 or 0
Cartesian product	An operation on sets forming a set of all tuples of elements from respective sets
Complement	An operation on a set, forming the set of all elements not present
Concatenation	An operation that sticks strings from one set together with strings from another set
Conjunction	Boolean AND operation
Connected graph	A graph with paths connecting every two nodes
Cycle	A path that starts and ends in the same node
Directed graph	A collection of points and arrows connecting some pairs of points
Disjunction	Boolean OR operation
Domain	The set of possible inputs to a function
Edge	A line in a graph
Element	An object in a set
Empty set	The set with no members
Empty string	The string of length zero
Equivalence relation	A binary relation that is reflexive, symmetric, and transitive
Function	An operation that translates inputs into outputs
Graph	A collection of points and lines connecting some pairs of points
Intersection	An operation on sets forming the set of common elements
k-tuple	A list of k objects
Language	A set of strings
Member	An object in a set
Node	A point in a graph
Pair	A list of two elements, also called a 2-tuple
Path	A sequence of nodes in a graph connected by edges
Predicate	A function whose range is {TRUE, FALSE}
Property	A predicate
Range	The set from which outputs of a function are drawn
Relation	A predicate, most typically when the domain is a set of k-tuples
Sequence	A list of objects
Set	A group of objects
Simple path	A path without repetition
String	A finite list of symbols from an alphabet
Symbol	A member of an alphabet
Tree	A connected graph without simple cycles
Union	An operation on sets combining all elements into a single set
Vertex	A point in a graph

0.3 DEFINITIONS, THEOREMS, AND PROOFS

Theorems and proofs are the heart and soul of mathematics and definitions are its spirit. These three entities are central to every mathematical subject, including ours.

Definitions describe the objects and notions that we use. A definition may be simple, as in the definition of *set* given earlier in this chapter, or complex as in the definition of *security* in a cryptographic system. Precision is essential to any mathematical definition. When defining some object we must make clear what constitutes that object and what does not.

After we have defined various objects and notions, we usually make ***mathematical statements*** about them. Typically a statement expresses that some object has a certain property. The statement may or may not be true, but like a definition, it must be precise. There must not be any ambiguity about its meaning.

A ***proof*** is a convincing logical argument that a statement is true. In mathematics an argument must be airtight, that is, convincing in an absolute sense. In everyday life or in the law, the standard of proof is lower. A murder trial demands proof "beyond any reasonable doubt." The weight of evidence may compel the jury to accept the innocence or guilt of the suspect. However, evidence plays no role in a mathematical proof. A mathematician demands proof beyond *any* doubt.

A ***theorem*** is a mathematical statement proved true. Generally we reserve the use of that word for statements of special interest. Occasionally we prove statements that are interesting only because they assist in the proof of another, more significant statement. Such statements are called ***lemmas***. Occasionally a theorem or its proof may allow us to conclude easily that other, related statements are true. These statements are called ***corollaries*** of the theorem.

FINDING PROOFS

The only way to determine the truth or falsity of a mathematical statement is with a mathematical proof. Unfortunately, finding proofs isn't always easy. It can't be reduced to a simple set of rules or processes. During this course, you will be asked to present proofs of various statements. Don't despair at the prospect! Even though no one has a recipe for producing proofs, some helpful general strategies are available.

First, carefully read the statement you want to prove. Do you understand all the notation? Rewrite the statement in your own words. Break it down and consider each part separately.

Sometimes the parts of a multipart statement are not immediately evident. One frequently occurring type of multipart statement has the form "P if and only if Q", often written "P iff Q", where both P and Q are mathematical statements. This notation is shorthand for a two-part statement. The first part is "P only if Q," which means: If P is true, then Q is true, written $P \Rightarrow Q$. The second is "P if Q," which means: If Q is true, then P is true, written $P \Leftarrow Q$. The first of these parts is the ***forward direction*** of the original statement and the second is the ***reverse direction***. We write "P if and only if Q" as $P \Longleftrightarrow Q$. To prove a statement of this form you must prove each of the two directions. Often, one of these directions is easier to prove than the other.

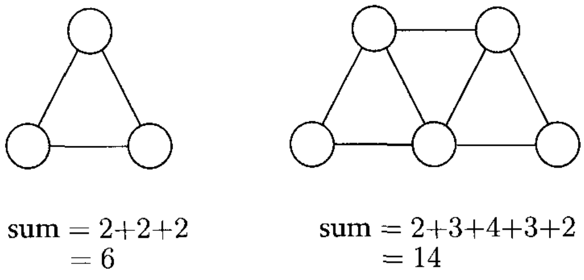
Another type of multipart statement states that two sets A and B are equal. The first part states that A is a subset of B, and the second part states that B is a subset of A. Thus one common way to prove that $A = B$ is to prove that every member of A also is a member of B and that every member of B also is a member of A.

Next, when you want to prove a statement or part thereof, try to get an intuitive, "gut" feeling of why it should be true. Experimenting with examples is especially helpful. Thus, if the statement says that all objects of a certain type have a particular property, pick a few objects of that type and observe that they actually do have that property. After doing so, try to find an object that fails to have the property, called a ***counterexample***. If the statement actually is true, you will not be able to find a counterexample. Seeing where you run into difficulty when you attempt to find a counterexample can help you understand why the statement is true.

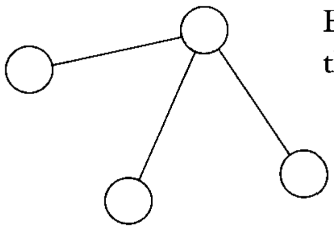
EXAMPLE 0.19

Suppose that you want to prove the statement *for every graph G, the sum of the degrees of all the nodes in G is an even number.*

First, pick a few graphs and observe this statement in action. Here are two examples.



Next, try to find a counterexample, that is, a graph in which the sum is an odd number.



Every time an edge is added, the sum increases by 2.

Can you now begin to see why the statement is true and how to prove it?

If you are still stuck trying to prove a statement, try something easier. Attempt to prove a special case of the statement. For example, if you are trying to prove that some property is true for every $k > 0$, first try to prove it for $k = 1$. If you succeed, try it for $k = 2$, and so on until you can understand the more general case. If a special case is hard to prove, try a different special case or perhaps a special case of the special case.

Finally, when you believe that you have found the proof, you must write it up properly. A well-written proof is a sequence of statements, wherein each one follows by simple reasoning from previous statements in the sequence. Carefully writing a proof is important, both to enable a reader to understand it and for you to be sure that it is free from errors.

The following are a few tips for producing a proof.

- *Be patient.* Finding proofs takes time. If you don't see how to do it right away, don't worry. Researchers sometimes work for weeks or even years to find a single proof.

- *Come back to it.* Look over the statement you want to prove, think about it a bit, leave it, and then return a few minutes or hours later. Let the unconscious, intuitive part of your mind have a chance to work.

- *Be neat.* When you are building your intuition for the statement you are trying to prove, use simple, clear pictures and/or text. You are trying to develop your insight into the statement, and sloppiness gets in the way of insight. Furthermore, when you are writing a solution for another person to read, neatness will help that person understand it.

- *Be concise.* Brevity helps you express high-level ideas without getting lost in details. Good mathematical notation is useful for expressing ideas concisely. But be sure to include enough of your reasoning when writing up a proof so that the reader can easily understand what you are trying to say.

For practice, let's prove one of DeMorgan's laws.

THEOREM 0.20

For any two sets A and B, $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

First, is the meaning of this theorem clear? If you don't understand the meaning of the symbols $\cup$ or $\cap$ or the overbar, review the discussion on page 4.

To prove this theorem we must show that the two sets $\overline{A \cup B}$ and $\overline{A} \cap \overline{B}$ are equal. Recall that we may prove that two sets are equal by showing that every member of one set also is a member of the other and vice versa. Before looking at the following proof, consider a few examples and then try to prove it yourself.

PROOF This theorem states that two sets, $\overline{A \cup B}$ and $\overline{A} \cap \overline{B}$, are equal. We prove this assertion by showing that every element of one also is an element of the other and vice versa.

Suppose that x is an element of $\overline{A \cup B}$. Then x is not in $A \cup B$ from the definition of the complement of a set. Therefore x is not in A and x is not in B, from the definition of the union of two sets. In other words, x is in $\overline{A}$ and x is in $\overline{B}$. Hence the definition of the intersection of two sets shows that x is in $\overline{A} \cap \overline{B}$.

For the other direction, suppose that x is in $\overline{A} \cap \overline{B}$. Then x is in both $\overline{A}$ and $\overline{B}$. Therefore x is not in A and x is not in B, and thus not in the union of these two sets. Hence x is in the complement of the union of these sets; in other words, x is in $\overline{A \cup B}$ which completes the proof of the theorem.

Let's now prove the statement in Example 0.19.

THEOREM 0.21

For every graph G, the sum of the degrees of all the nodes in G is an even number.

PROOF Every edge in G is connected to two nodes. Each edge contributes 1 to the degree of each node to which it is connected. Therefore each edge contributes 2 to the sum of the degrees of all the nodes. Hence, if G contains e edges, then the sum of the degrees of all the nodes of G is $2e$, which is an even number.

0.4

TYPES OF PROOF

Several types of arguments arise frequently in mathematical proofs. Here, we describe a few that often occur in the theory of computation. Note that a proof may contain more than one type of argument because the proof may contain within it several different subproofs.

PROOF BY CONSTRUCTION

Many theorems state that a particular type of object exists. One way to prove such a theorem is by demonstrating how to construct the object. This technique is a ***proof by construction***.

Let's use a proof by construction to prove the following theorem. We define a graph to be ***k*-regular** if every node in the graph has degree k.

THEOREM 0.22

For each even number n greater than 2, there exists a 3-regular graph with n nodes.

PROOF Let n be an even number greater than 2. Construct graph $G = (V, E)$ with n nodes as follows. The set of nodes of G is $V = \{0, 1, \ldots, n-1\}$, and the set of edges of G is the set

$$\begin{aligned} E = \{\, \{i, i+1\} \mid \text{ for } 0 \le i \le n-2\} &\cup \{\, \{n-1, 0\}\,\} \\ \cup\ \{\, \{i, i+n/2\} \mid \text{ for } 0 \le i \le n/2-1\}. \end{aligned}$$

Picture the nodes of this graph written consecutively around the circumference of a circle. In that case the edges described in the top line of E go between adjacent pairs around the circle. The edges described in the bottom line of E go between nodes on opposite sides of the circle. This mental picture clearly shows that every node in G has degree 3.

PROOF BY CONTRADICTION

In one common form of argument for proving a theorem, we assume that the theorem is false and then show that this assumption leads to an obviously false consequence, called a contradiction. We use this type of reasoning frequently in everyday life, as in the following example.

EXAMPLE 0.23

Jack sees Jill, who has just come in from outdoors. On observing that she is completely dry, he knows that it is not raining. His "proof" that it is not raining is that, *if it were raining* (the assumption that the statement is false), *Jill would be wet* (the obviously false consequence). Therefore it must not be raining.

Next, let's prove by contradiction that the square root of 2 is an irrational number. A number is ***rational*** if it is a fraction m/n where m and n are integers; in other words, a rational number is the *ratio* of integers m and n. For example, $2/3$ obviously is a rational number. A number is ***irrational*** if it is not rational.

THEOREM 0.24

$\sqrt{2}$ is irrational.

PROOF First, we assume for the purposes of later obtaining a contradiction that $\sqrt{2}$ is rational. Thus

$$\sqrt{2} = \frac{m}{n},$$

where both m and n are integers. If both m and n are divisible by the same integer greater than 1, divide both by that integer. Doing so doesn't change the value of the fraction. Now, at least one of m and n must be an odd number.

We multiply both sides of the equation by n and obtain

$$n\sqrt{2} = m.$$

We square both sides and obtain

$$2n^2 = m^2.$$

Because m^2 is 2 times the integer n^2, we know that m^2 is even. Therefore m, too, is even, as the square of an odd number always is odd. So we can write $m = 2k$ for some integer k. Then, substituting $2k$ for m, we get

$$\begin{aligned} 2n^2 &= (2k)^2 \\ &= 4k^2. \end{aligned}$$

Dividing both sides by 2 we obtain

$$n^2 = 2k^2.$$

But this result shows that n^2 is even and hence that n is even. Thus we have established that both m and n are even. But we had earlier reduced m and n so that they were *not* both even, a contradiction.

PROOF BY INDUCTION

Proof by induction is an advanced method used to show that all elements of an infinite set have a specified property. For example, we may use a proof by induction to show that an arithmetic expression computes a desired quantity for every assignment to its variables or that a program works correctly at all steps or for all inputs.

To illustrate how proof by induction works, let's take the infinite set to be the natural numbers, $\mathcal{N} = \{1, 2, 3, \ldots\}$, and say that the property is called $\mathcal{P}$. Our goal is to prove that $\mathcal{P}(k)$ is true for each natural number k. In other words, we want to prove that $\mathcal{P}(1)$ is true, as well as $\mathcal{P}(2)$, $\mathcal{P}(3)$, $\mathcal{P}(4)$, and so on.

Every proof by induction consists of two parts, the ***induction step*** and the ***basis***. Each part is an individual proof on its own. The induction step proves that for each $i \geq 1$, if $\mathcal{P}(i)$ is true, then so is $\mathcal{P}(i+1)$. The basis proves that $\mathcal{P}(1)$ is true.

When we have proven both of these parts, the desired result follows—namely, that $\mathcal{P}(i)$ is true for each i. Why? First, we know that $\mathcal{P}(1)$ is true because the basis alone proves it. Second, we know that $\mathcal{P}(2)$ is true because the induction step proves that, if $\mathcal{P}(1)$ is true then $\mathcal{P}(2)$ is true, and we already know that $\mathcal{P}(1)$ is true. Third, we know that $\mathcal{P}(3)$ is true because the induction step proves that, if $\mathcal{P}(2)$ is true then $\mathcal{P}(3)$ is true, and we already know that $\mathcal{P}(2)$ is true. This process continues for all natural numbers, showing that $\mathcal{P}(4)$ is true, $\mathcal{P}(5)$ is true, and so on.

Once you understand the preceding paragraph, you can easily understand variations and generalizations of the same idea. For example, the basis doesn't necessarily need to start with 1; it may start with any value b. In that case the induction proof shows that $\mathcal{P}(k)$ is true for every k that is at least b.

In the induction step the assumption that $\mathcal{P}(i)$ is true is called the ***induction hypothesis***. Sometimes having the stronger induction hypothesis that $\mathcal{P}(j)$ is true for every $j \leq i$ is useful. The induction proof still works because, when we want to prove that $\mathcal{P}(i+1)$ is true we have already proved that $\mathcal{P}(j)$ is true for every $j \leq i$.

The format for writing down a proof by induction is as follows.

Basis: Prove that $\mathcal{P}(1)$ is true.

$$\vdots$$

Induction step: For each $i \geq 1$, assume that $\mathcal{P}(i)$ is true and use this assumption to show that $\mathcal{P}(i+1)$ is true.

$$\vdots$$

Now, let's prove by induction the correctness of the formula used to calculate the size of monthly payments of home mortgages. When buying a home, many people borrow some of the money needed for the purchase and repay this loan over a certain number of years. Typically, the terms of such repayments stipulate that a fixed amount of money is paid each month to cover the interest, as well as part of the original sum, so that the total is repaid in 30 years. The formula for calculating the size of the monthly payments is shrouded in mystery, but actually is quite simple. It touches many people's lives, so you should find it interesting. We use induction to prove that it works, making it a good illustration of that technique.

First, we set up the names and meanings of several variables. Let P be the *principal*, the amount of the original loan. Let $I > 0$ be the yearly *interest rate* of the loan, where $I = 0.06$ indicates a 6% rate of interest. Let Y be the monthly payment. For convenience we define another variable M from I, for the monthly multiplier. It is the rate at which the loan changes each month because of the interest on it. Following standard banking practice we assume monthly compounding, so $M = 1 + I/12$.

Two things happen each month. First, the amount of the loan tends to increase because of the monthly multiplier. Second, the amount tends to decrease because of the monthly payment. Let P_t be the amount of the loan outstanding after the tth month. Then $P_0 = P$ is the amount of the original loan, $P_1 = MP_0 - Y$ is the amount of the loan after one month, $P_2 = MP_1 - Y$ is the amount of the loan after two months, and so on. Now we are ready to state and prove a theorem by induction on t that gives a formula for the value of P_t.

THEOREM 0.25

For each $t \geq 0$,

$$P_t = PM^t - Y\left(\frac{M^t - 1}{M - 1}\right).$$

PROOF

Basis: Prove that the formula is true for $t = 0$. If $t = 0$, then the formula states that

$$P_0 = PM^0 - Y\left(\frac{M^0 - 1}{M - 1}\right).$$

We can simplify the right-hand side by observing that $M^0 = 1$. Thus we get

$$P_0 = P,$$

which holds because we have defined P_0 to be P. Therefore we have proved that the basis of the induction is true.

Induction step: For each $k \geq 0$ assume that the formula is true for $t = k$ and show that it is true for $t = k + 1$. The induction hypothesis states that

$$P_k = PM^k - Y\left(\frac{M^k - 1}{M - 1}\right).$$

Our objective is to prove that

$$P_{k+1} = PM^{k+1} - Y\left(\frac{M^{k+1} - 1}{M - 1}\right).$$

We do so with the following steps. First, from the definition of P_{k+1} from

P_k, we know that

$$P_{k+1} = P_k M - Y.$$

Therefore, using the induction hypothesis to calculate P_k,

$$P_{k+1} = \left[PM^k - Y\left(\frac{M^k - 1}{M - 1}\right)\right] M - Y.$$

Multiplying through by M and rewriting Y yields

$$\begin{aligned} P_{k+1} &= PM^{k+1} - Y\left(\frac{M^{k+1} - M}{M - 1}\right) - Y\left(\frac{M - 1}{M - 1}\right) \\ &= PM^{k+1} - Y\left(\frac{M^{k+1} - 1}{M - 1}\right). \end{aligned}$$

Thus the formula is correct for $t = k + 1$, which proves the theorem.

Problem 0.14 asks you to use the preceding formula to calculate actual mortgage payments.

EXERCISES

0.1 Examine the following formal descriptions of sets so that you understand which members they contain. Write a short informal English description of each set.

a. $\{1, 3, 5, 7, \ldots\}$

b. $\{\ldots, -4, -2, 0, 2, 4, \ldots\}$

c. $\{n \mid n = 2m \text{ for some } m \text{ in } \mathcal{N}\}$

d. $\{n \mid n = 2m \text{ for some } m \text{ in } \mathcal{N}, \text{ and } n = 3k \text{ for some } k \text{ in } \mathcal{N}\}$

e. $\{w \mid w$ is a string of 0s and 1s and w equals the reverse of $w\}$

f. $\{n \mid n \text{ is an integer and } n = n + 1\}$

0.2 Write formal descriptions of the following sets

a. The set containing the numbers 1, 10, and 100

b. The set containing all integers that are greater than 5

c. The set containing all natural numbers that are less than 5

d. The set containing the string aba

e. The set containing the empty string

f. The set containing nothing at all

0.3 Let A be the set $\{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$ and B be the set $\{\mathtt{x}, \mathtt{y}\}$.

- **a.** Is A a subset of B?
- **b.** Is B a subset of A?
- **c.** What is $A \cup B$?
- **d.** What is $A \cap B$?
- **e.** What is $A \times B$?
- **f.** What is the power set of B?

0.4 If A has a elements and B has b elements, how many elements are in $A \times B$? Explain your answer.

0.5 If C is a set with c elements, how many elements are in the power set of C? Explain your answer.

0.6 Let X be the set $\{1, 2, 3, 4, 5\}$ and Y be the set $\{6, 7, 8, 9, 10\}$. The unary function $f\colon X \longrightarrow Y$ and the binary function $g\colon X \times Y \longrightarrow Y$ are described in the following tables.

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

g	6	7	8	9	10
1	10	10	10	10	10
2	7	8	9	10	6
3	7	7	8	8	9
4	9	8	7	6	10
5	6	6	6	6	6

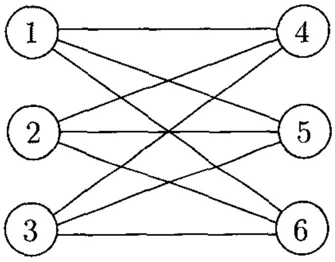
- **a.** What is the value of $f(2)$?
- **b.** What are the range and domain of f?
- **c.** What is the value of $g(2, 10)$?
- **d.** What are the range and domain of g?
- **e.** What is the value of $g(4, f(4))$?

0.7 For each part, give a relation that satisfies the condition.

- **a.** Reflexive and symmetric but not transitive
- **b.** Reflexive and transitive but not symmetric
- **c.** Symmetric and transitive but not reflexive

0.8 Consider the undirected graph $G = (V, E)$ where V, the set of nodes, is $\{1, 2, 3, 4\}$ and E, the set of edges, is $\{\{1, 2\}, \{2, 3\}, \{1, 3\}, \{2, 4\}, \{1, 4\}\}$. Draw the graph G. What is the degree of node 1? of node 3? Indicate a path from node 3 to node 4 on your drawing of G.

0.9 Write a formal description of the following graph.



PROBLEMS

0.10 Find the error in the following proof that $2 = 1$.
Consider the equation $a = b$. Multiply both sides by a to obtain $a^2 = ab$. Subtract b^2 from both sides to get $a^2 - b^2 = ab - b^2$. Now factor each side, $(a + b)(a - b) = b(a - b)$, and divide each side by $(a - b)$, to get $a + b = b$. Finally, let a and b equal 1, which shows that $2 = 1$.

0.11 Find the error in the following proof that all horses are the same color.
CLAIM: In any set of h horses, all horses are the same color.
PROOF: By induction on h.

Basis: For $h = 1$. In any set containing just one horse, all horses clearly are the same color.

Induction step: For $k \geq 1$ assume that the claim is true for $h = k$ and prove that it is true for $h = k + 1$. Take any set H of $k + 1$ horses. We show that all the horses in this set are the same color. Remove one horse from this set to obtain the set H_1 with just k horses. By the induction hypothesis, all the horses in H_1 are the same color. Now replace the removed horse and remove a different one to obtain the set H_2. By the same argument, all the horses in H_2 are the same color. Therefore all the horses in H must be the same color, and the proof is complete.

0.12 Show that every graph with 2 or more nodes contains two nodes that have equal degrees.

^{A*}**0.13** **Ramsey's theorem.** Let G be a graph. A ***clique*** in G is a subgraph in which every two nodes are connected by an edge. An ***anti-clique***, also called an ***independent set***, is a subgraph in which every two nodes are not connected by an edge. Show that every graph with n nodes contains either a clique or an anti-clique with at least $\frac{1}{2}\log_2 n$ nodes.

A0.14 Use Theorem 0.25 to derive a formula for calculating the size of the monthly payment for a mortgage in terms of the principal P, interest rate I, and the number of payments t. Assume that, after t payments have been made, the loan amount is reduced to 0. Use the formula to calculate the dollar amount of each monthly payment for a 30-year mortgage with 360 monthly payments on an initial loan amount of $100,000 with a 5% annual interest rate.

SELECTED SOLUTIONS

0.13 Make space for two piles of nodes, A and B. Then, starting with the entire graph, repeatedly add each remaining node x to A if its degree is greater than one half the number of remaining nodes and to B otherwise, and discard all nodes to which x isn't (is) connected if it was added to A (B). Continue until no nodes are left. At most half of the nodes are discarded at each of these steps, so at least $\log_2 n$ steps will occur before the process terminates. Each step adds a node to one of the piles, so one of the piles ends up with at least $\frac{1}{2}\log_2 n$ nodes. The A pile contains the nodes of a clique and the B pile contains the nodes of an anti-clique.

0.14 We let $P_t = 0$ and solve for Y to get the formula: $Y = PM^t(M-1)/(M^t-1)$. For $P = \$100,000$, $I = 0.05$, and $t = 360$ we have $M = 1 + (0.05)/12$. We use a calculator to find that $Y \approx \$536.82$ is the monthly payment.