

bigger set. For our argument can be slightly modified to show that any interval, no matter how small in length, of the real numbers has a higher cardinal number than the natural numbers.

3.8 IMPLICATIONS OF THE THEORY OF TRANSFINITE NUMBERS TO COMPUTING

The denumerability of the natural numbers implies immediately the denumerability of all finite strings made up of symbols from some finite alphabet. Suppose, for example, that we employ a 256-character alphabet, of which the blank is one character. Then, any finite string of symbols, or word, formed from this alphabet can be considered to be a base 256 representation of one of the natural numbers. This gives an immediate one-to-one correspondence between all such strings and the natural numbers, and, therefore, these strings form a denumerable class. This implies that the class of all possible books written in the English language, the class of all possible programs that can be written in any given programming language, the class of all descriptions of numbers, as in the discussion in Chapter I of Berry's paradox, the class of all possible human thoughts (if we assume that any thought can be expressed as a string of English words—which assumption is, perhaps, doubtful) are all denumerable classes. To be practical, there is, of course, some upper bound imposed by the real world on the size of each of these classes, and they are, in fact, finite rather than infinite. But, even if we ignore the restraints of a finite world and finite man, these classes are no more than denumerably infinite.

On the other hand, the class of all possible total functions defined with domain equal to the natural numbers and range contained within the natural numbers is nondenumerably infinite. There are several ways of proving this, but one rather immediate proof can be given by following the "diagonalization" argument that we used to show the nondenumerability of the real numbers in the interval from 0 to 1.

Assume that all such functions comprise a denumerable class. We then can list them in tabular form, identifying each function by the string of its values at arguments 1, 2, 3, Suppose, for example, this gives us the form in Fig. 3.14.

Now, paralleling what was done in the case of the real numbers, we form a new function on the natural numbers by writing a string that differs in every position from the corresponding element of the above array such as

$$f: 2, 7, 38, 100, \dots$$

f is certainly not the same as any of the f_i ($i = 1, 2, 3, \dots$), since it must differ at one or more arguments from every f_i . Therefore, no such list of all total functions exists; they comprise a nondenumerable class.

f_1 :	3,	28,	2,	7,	...
f_2 :	57,	9,	3,	1,	...
f_3 :	2,	118,	46,	5,	...
f_4 :	5,	62,	3,	7,	...
\vdots	\vdots	\vdots	\vdots	\vdots	

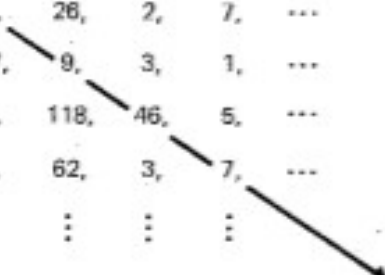


Figure 3.14

On the other hand, the set of possible programs, in any given programming language, that can be written to compute functions is, from our earlier remarks, a denumerable class. Therefore, there exist functions on the natural numbers that cannot be computed by any program.

This corresponds to a property of the real numbers to which we have alluded. There are only a denumerably infinite number of ways of describing individual real numbers, using any finite vocabulary. For example, "the ratio of the circumference of a circle to its diameter," "the square root of the biggest, in absolute value, of the roots of the equation

$$(x \star \star 3) - 17 \star (x \star \star 2) + 35 \star x - 15,735 = 0,"$$

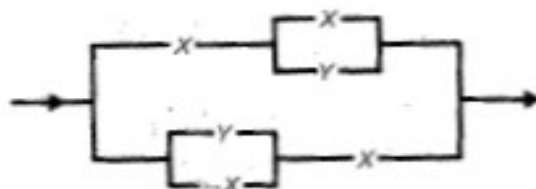
"the limit as $n \rightarrow \infty$ of $(1 + 1/n) \star \star n$," are examples of such descriptions. We have seen, however, that there are (accepting the argument given) a non-denumerably infinite number of such real numbers. Therefore, the great majority, in some special sense, of these real numbers can individually never be identified, named, or described in any manner whatsoever.

This property of the real numbers is reminiscent of the claim by the philosopher Ludwig Wittgenstein that there are facts that cannot be enunciated or expressed. They can only be "known" in a sense by their effects. As he writes in his *Tractatus Logico-Philosophicus*, "There are indeed things that cannot be put into words. They make themselves manifest," and again, "What we cannot speak about, we must pass over in silence." However, in mathematics, we certainly do not remain silent about the totality of real numbers.

EXERCISES

3.1 Draw a circuit, employing switches to represent Boolean variables, that realizes the Boolean expression $X \mid (Y \& Z)$. Write the truth table for this expression.

3.2 Given the following switching circuit where switches (not shown in detail) represent the indicated Boolean variables:



- give a Boolean expression that represents the circuit.
 - write a truth table for the result of (a) and give the disjunctive normal form for the corresponding Boolean expression.
- 3.3 Let A, B, C be three bits. Write a table that defines for each of the 8 possible combinations of A, B, C that bit D which when added (arithmetically) to the algebraic sum of A, B, C gives an even sum (D is called an "even parity bit"). Write a Boolean expression for D in terms of A, B, C using the disjunctive normal form. Draw a logical circuit (using AND and OR gates and inverters) with inputs A, B, C that will give D as output.
- 3.4 Let A, B, C be three bits. Write the table that shows for each combination of A, B, C the corresponding value of the "majority bit" m where m is 0 or 1, according to which value occurs more frequently among A, B, C . Draw a logical circuit with inputs A, B, C that will give m as output.
- 3.5 Design a "black box" with three inputs and a single output, employing logical gates and inverters, that gives an output signal when at least one and at most two inputs carry signals (or are equal to 1).
- 3.6 If the universe is taken to be the first 1,000 positive integers and $\{P(i) \mid i = 1, 2, \dots, M\}$ is an array of M positive integers ($\leq 1,000$) stored in memory, write a PL/I subroutine (or use any other suitable programming language) for finding the set P' , the complement of the set P (denote P' by Q in the program).
- 3.7 Let $\{P(i) \mid i = 1, 2, \dots, M\}$ and $\{Q(i) \mid i = 1, 2, \dots, N\}$ be two arrays of integers stored in memory with $M, N \leq 1,000$. Write a PL/I subroutine (or use any other suitable programming language) for finding the sets R , defined as $P \cup Q$, and S , defined as $P \cap Q$.
- 3.8 If $N(A)$ denotes the number of elements in the set A , we can write:
 $N(A \cup B) = N(A) + N(B) - N(A \cap B)$. Justify this equation. Applying this identity twice we calculate:

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

In a survey it is reported that of 1,000 programmers, 650 habitually flowchart their programs, 788 are skilled COBOL programmers, 675 are men, 278 of the women are skilled COBOL programmers, 440 programmers both habitually flowchart and are

skilled in COBOL, 210 women habitually flowchart, and 166 women are both skilled in COBOL and habitually flowchart. Would you accept these data as being accurately reported?

3.9 The set theory formula $(A \cup B)' = A' \cap B'$ is the analog of what statement of the propositional calculus? Prove, by truth tables, this theorem of Boolean logic to which the set formula corresponds. Generalize this formula to give the complement of the union of n sets.

3.10 Let A be the range of the function $f(x) = x^2 + 3$, B the domain of the function $g(x) = x - 25$ where the universe is the set of nonnegative integers $\{0, 1, 2, \dots\}$. List the elements of the set $A \cap B'$.

3.11 Illustrate the following set equality by using Venn diagrams and shading the region corresponding to each side of the equation:

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

3.12 Let $A - B$ be the set of those elements that are in set A but not in set B . Express $A - B$ in terms of the set operations \cup , \cap , $'$ and show by Venn diagrams that

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

3.13 Given any set S formed from the sets A, B, C by the set operations \cup , \cap , $'$, explain the analog of the disjunctive normal form for S in terms of the Venn diagram. Take as an example

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

3.14 With the same given conditions as in exercise 3.13, express S in the *conjunctive* normal form, i.e., as the intersection of sets which are individually unions of selected sets and of the complements of selected sets. (Hint: Express S' using the disjunctive normal form, then take the complement to obtain an expression for S , using the result of Exercise 3.9.) Cf. Exercise 2.30.

3.15 Given any set theory equation of the form $S_1 = S_2$, where S_1, S_2 are set expressions formed from A, B, C by using \cup , \cap , $'$, show that this is a true equation for arbitrary A, B, C if it is true for any assignment of Φ, U to A, B, C . (Hint: Assuming the hypothesis, show that if any point x is in S_1 , it must be in S_2 and conversely, by using the disjunctive normal forms of S_1, S_2 , as in Exercise 3.13.)

3.16 Write down the set P , defined as the set of all subsets (the power set) of $\{1, 2, 3\}$. If the set P is taken as the universe, what is the complement of the set $B: \{(1, 2), (2, 3)\}$?

3.17 If the universe is the set of natural numbers and $S = \{x \mid x^2 > 5x\}$, $T = \{y \mid \neg(y^2 - 2y = 0)\}$ what is $T' \cup S$?

3.18 If the universe is the set of natural numbers, what is the set Q defined by

$$a. Q = \{n \mid \neg \exists m(2m = n)\}?$$

$$b. Q = \{m \mid \forall z[(z \neq 5) \rightarrow ((\exists k(m = z \cdot k)) \rightarrow (z = 1))]\}?$$

3.19 Prove that the power set of a given set S has greater cardinal number than S . (Hint: Use the method of *reductio ad absurdum*. Assume that the elements of the set

of all subsets *can* be put into one-to-one correspondence with the elements of S . Then construct a subset of S to which there cannot possibly correspond any element of S .)

3.20 Write a program in any suitable language that will generate *all* pairs of natural numbers (make the substantial and unrealistic simplifying assumption of ignoring the limited word size of any real computer). This result also implies the denumerability of the set of all pairs of natural numbers.

3.21 Show that the points of any two line segments of different length can be put into one-to-one correspondence (i.e., the segments have the same cardinality). Show, also, that any line segment has as many points as an infinite line.

3.22 Following the discussion of the denumerability of the rational numbers we can, in accordance with the following suggested scheme, establish an effective one-to-one correspondence between all *pairs* of nonnegative integers and *single* nonnegative integers.

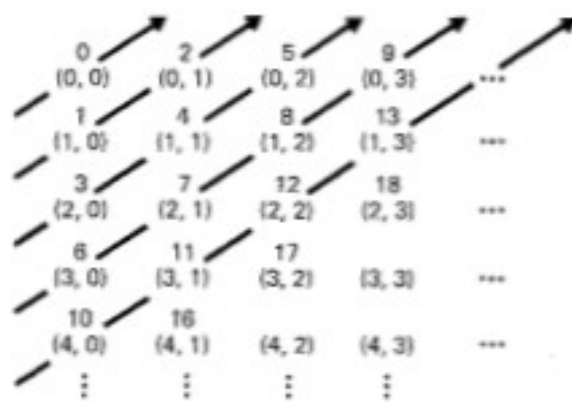


Figure 3.15

Let the pair (x, y) correspond in this fashion to the number z . Write algebraic expressions that give z as a function of x, y and conversely that give x, y as functions of z .

APPENDIX THE RUDIMENTS OF GRAPH THEORY

Graph theory is a part of combinatorial mathematics, that branch of mathematics that concerns the study of finite sets and problems of enumeration, arrangement, or design. The theory is not, strictly speaking, usually consid-

ered to be a part of the foundations of mathematics. Yet a case can be made for so classifying it. In any event, the basic ideas of graph theory are of great importance in the computer sciences, especially in formal linguistics and automata theory. One quickly infers this from the number of illustrations, in works in these areas, that consist of sets of nodes connected in various ways. For these reasons, we include this material as an appendix.

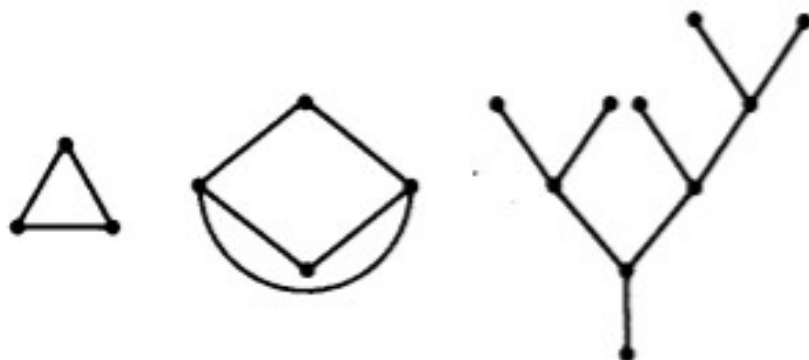
As with all of the mathematics we cite, there is a very extensive literature devoted to this subject. However, it is the basic ideas that are important to the computer scientist, the representation as graphs of a number of structures that appear in computing. We briefly describe the fundamental definitions and vocabulary and give some examples of the representation of graphs and of algorithms related to them.

3.9 SOME FUNDAMENTAL NOTIONS OF GRAPH THEORY

We have repeatedly emphasized that all items represented in a digital computer are strings of binary digits and, therefore, are representations of the natural numbers. This is not, however, very restrictive in applications, and many different kinds of "data structures" are mapped into the natural numbers and thus represented in the computer. One frequently occurring structure in computer science is the class of *graphs*. We are here using this somewhat overworked term with still another meaning.

Graph theory concerns the study of structures that can be drawn in the plane consisting of points, or nodes, and of path segments, not necessarily straight lines, that connect some of these nodes (there may be more than one path segment joining two nodes, and we may have a path segment extending from a node back to itself). We shall consider only finite graphs, i.e., graphs with a finite number of nodes and a finite number of path segments.

Examples.



We shall in this section introduce some of the fundamental definitions and notation of graph theory. We shall not go into the extensive body of results that are included in numerous treatises on this subject. For example, see [1, 2, 4, 5, 7].

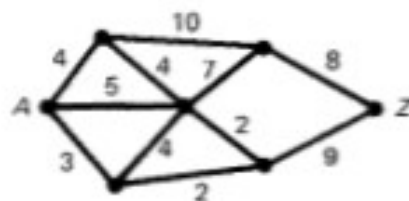
If all the path segments are oriented, or directed (like one-way streets), the graph is called a directed graph or, more briefly, a *digraph*.

Examples.



In a digraph each node has an "indegree" (number of incoming path segments) and an "outdegree" (number of outgoing path segments). In computing, most applications of graphs involve digraphs.

Path segments may have numeric "weights" associated with them.



Problem. Find the "shortest" (least sum of weights) path from *A* to *Z*.

The weight associated with a path segment from a node *P* to a node *Q* may indicate the distance from a physical point *P* to a point *Q*, but in general it can have a meaning and a measure that are unrelated to this distance. If, for example, *P* and *Q* denote work stations in some assembly line procedure, the associated weight might indicate the time, or the cost, for a component being manufactured to reach station *Q* after arriving at *P*.

Graphs describe the connectivity of the nodes. They do not have properties stemming from the measures of the segments and angles that might appear in a particular drawing of them. Thus, the four graphs in Fig. 3.16 are equivalent. Each indicates that the four given nodes are connected to each other in the same manner. (Such graphs are said to be *isomorphic*.)

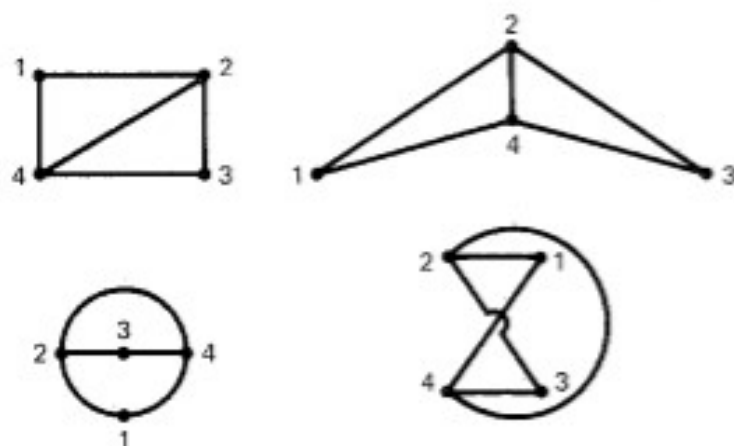
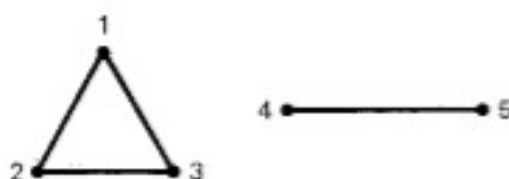


Figure 3.16

Two graphs G, G' are isomorphic if there is a one-to-one onto mapping of the nodes of G to the nodes of G' such that there is a path segment connecting nodes u, v in G if and only if there is a path segment connecting the images u', v' of these nodes in G' . If G, G' are digraphs, the corresponding path segments in G' must be oriented in the same way as those in G .)

A graph is *connected* if every two distinct nodes are joined by a chain of path segments, or simply by a *path*; otherwise, it is said to be disconnected.

The following graph is *not* connected. That is, it includes 5 nodes with connectivity as shown. There is, for example, no path from node 1 to node 4.



A digraph is *strongly connected* if every node can be reached from every other node by a directed path, i.e., a path whose every segment is traced according to its given orientation.

A directed *tree* (Fig. 3.17) is a connected digraph in which every node, except one, has indegree 1. The exceptional or, as it is often called, the "distinguished" node is called the *root* and has indegree 0. Nodes with zero outdegree are called terminal nodes or *leaves*. In an unweighted tree the

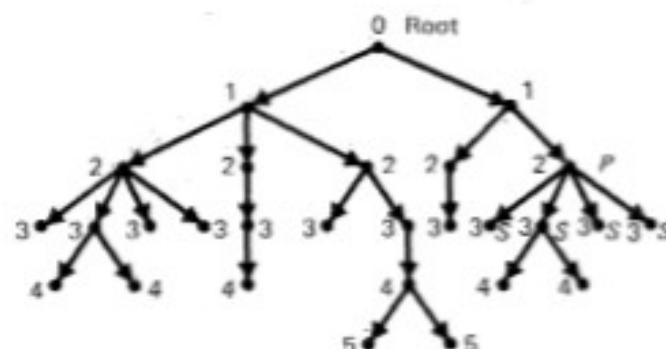


Figure 3.17

length of the (unique) path from the root to any node is taken to be the number of segments in the path and is called the *level* of the node. Usual trees are drawn with the root at the top. An undirected tree is simply a directed tree without the path segment orientations indicated. However, a tree always has an implied orientation given by tracing a path from the root to each of its nodes.

A directed path in a digraph that leads from a node back to that same node is called a *cycle*. A directed tree has no cycles and is an example of an *acyclic* graph. A cycle is *simple* if all its nodes are distinct (i.e., no node appears twice, except for the starting node). If a simple cycle contains all the nodes of a digraph it is called *Hamiltonian* (Fig. 3.18). A path is said to be *simple* if it does not contain any cycles; it is *Hamiltonian* if it contains all the nodes of the digraph. We also speak of cycles and Hamiltonian paths in undirected graphs where these terms have the obvious meanings.

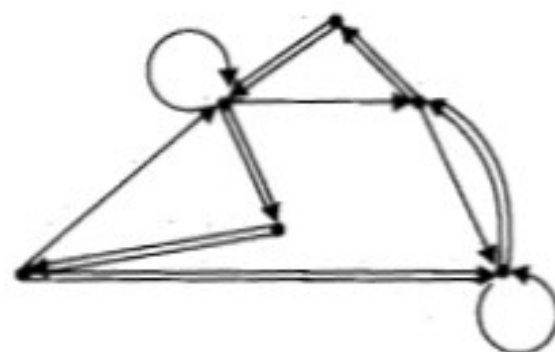


Figure 3.18

The "traveling salesman" problem of determining the shortest path that passes through each of a given set of cities is that of determining a Hamiltonian path of minimum length. (The digraph in this case is weighted, each path segment having its length as its weight.)

The problem of the "knight's tour" in chess consists of moving the knight starting from any given square so that in 64 moves it visits every square on the board and returns to the starting square. In the language of graphs, it is the problem of finding a Hamiltonian cycle in the graph whose nodes are the 64 squares of the board and whose path segments join every pair of squares that are at a knight's move from each other.

A *subtree* of a given tree is the tree obtained by taking any node of the tree as a root and including all path segments and nodes that can be reached by traveling down from that node. If we "prune" a subtree from a given tree, the remaining structure is still a tree.

Family trees are often used in genealogy to describe the lineage of individuals. A tree that shows some of the ancestors of a given individual is a graph that satisfies the conditions for being a tree (assuming no marriages of cousins), the given individual being at the root of the tree. The representation of the descendants of an individual is also a tree if only one parent of each individual in the lineage is shown. If two parents appear for any individual, the representation is a digraph but not a tree.

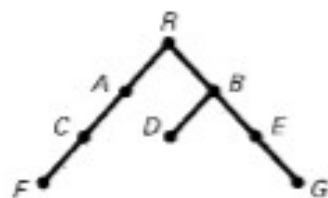
By analogy with family trees we speak of the relationship between the members of certain pairs of nodes as that between a parent, or father, node and a child, or son, node. (It is difficult to root out all evidence of sexism in the literature.) Thus, in Fig. 3.17 node P is the father of four sons labeled S .

A tree is a natural generalization of an ordered list. In a list, every element has as its predecessor the element immediately to its left. If we generalize the notion of ordering so that every element in the list is considered a successor of some preceding element, not necessarily the one immediately to its left, we are led to the notion of a tree.

For example, consider the following list in which each element is to be considered a successor of the element written in parentheses next to it.

$$R, A(R), B(R), C(A), D(B), E(B), F(C), G(E).$$

The tree corresponding to this list with its special ordering is:



3.11 THE ENCODING OF GRAPHS IN THE COMPUTER

It is easy to represent a graph by a list of symbols that can then be represented in memory. We can, for example, represent a digraph by listing each path segment extending from node a to node b as the ordered pair (a, b) .

Thus, such a representation of the following graph G is indicated:



Trees can be represented more compactly. We can, for example, use a prefix or postfix notation corresponding to the prefix or postfix representation for algebraic expressions and their trees, as we have implied in the preceding discussion of the tree structure of expressions.

Consider the tree in Fig. 3.21 where the level number of each node is written in parentheses next to it.

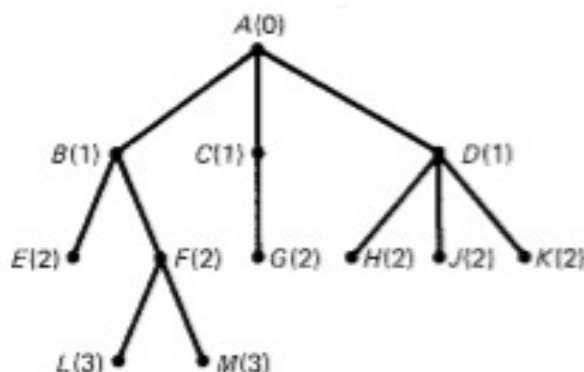
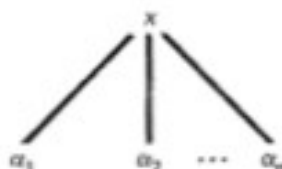


Figure 3.21

A postfix string for this tree is:

ELMFBGCHJKDA

That is, the string $\alpha_1 \alpha_2 \dots \alpha_n x$ represents the tree:



and the notation is extended in obvious recursive fashion to arbitrary trees. (The node x above corresponds to an operation symbol in expressions. The postfix notation applies even if the operation is not binary, but we must then be able to determine the number of operands for a given operation.)

A digraph can also be represented by an "adjacency matrix." A *matrix* in mathematics is a rectangular array of elements, often written as in the following array A of 3 rows and 4 columns.

$$A: \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

Each element a_{ij} ($i = 1, 2, 3; j = 1, 2, 3, 4$) in the matrix has in this example two subscripts, the first indicating the row i and the second the column j that contain that element. The matrix A is said to be a 3×4 matrix, indicating the numbers of rows and of columns in that order. These numbers define the "dimensions" of the matrix.

An adjacency matrix is a square matrix in which each row (and column) corresponds to one of the nodes of the graph. We take $a_{ij} = 1$ if there is a path segment directed from node i to node j , and $a_{ij} = 0$ if there is none.

Thus, the adjacency matrix for the graph G above is

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

If the path segments have weights, we can write down an adjacency matrix that contains these weights as entries, not just 0 and 1. If there is more than one path segment from a node i to a node j , we might wish to indicate the number of such segments in the (i, j) position of the matrix.

In algebra, the addition or subtraction of two matrices of the same dimensions is defined in the following way. If A , with elements $\{a_{ij}\}$, is an $l \times m$ matrix and B , with elements $\{b_{ij}\}$, has the same dimensions, then the matrix $C = A \pm B$ is defined to be that $l \times m$ matrix whose elements $\{c_{ij}\}$ are given by $c_{ij} = a_{ij} \pm b_{ij}$.

As an example we have

$$\begin{pmatrix} 1 & 0 & -2 \\ 3 & 2 & 2 \\ 0 & -1 & 3 \end{pmatrix} + \begin{pmatrix} 2 & 1 & 0 \\ 4 & -3 & 2 \\ 1 & 0 & 4 \end{pmatrix} = \begin{pmatrix} 3 & 1 & -2 \\ 7 & -1 & 4 \\ 1 & -1 & 7 \end{pmatrix}$$

Multiplication of matrices is defined as follows. Let A with elements $\{a_{ij}\}$ be an $l \times m$ matrix and B with elements $\{b_{ij}\}$ be an $m \times n$ matrix. That is, A has the same number of columns as B has rows. The product $C = A \times B$ is then defined to be an $l \times n$ matrix whose elements $\{c_{ij}\}$ are given by the following equation:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{im}b_{mj},$$

or, as it is usually written,

$$c_{ij} = \sum_{k=1}^m a_{ik}b_{kj}.$$

Thus, as an example, we can compute the following product of two square matrices of order 3:

$$\begin{pmatrix} 1 & 0 & -2 \\ 3 & 2 & 2 \\ 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 4 & -3 & 2 \\ 1 & 0 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -8 \\ 16 & -3 & 12 \\ -1 & 3 & 10 \end{pmatrix}$$

Similar matrix operations are useful in dealing with adjacency matrices. Suppose that A is the $n \times n$ adjacency matrix corresponding to a graph H with n nodes. We define the product $A \cdot A = A^2$ to be that matrix which is obtained by following a multiplication rule similar to the one used in algebra but in which *Boolean* addition (the OR) and *Boolean* multiplication (the AND) are used in place of ordinary addition and multiplication. We write then

$$(A^2)_{ij} = \alpha_{ij} = (a_{i1} \& a_{1j}) \vee (a_{i2} \& a_{2j}) \vee \dots \vee (a_{in} \& a_{nj}).$$

What meaning can be attached to this element α_{ij} in the i, j position of the square of the adjacency matrix? Note that $\alpha_{ij} = 1$ if any one of the products $(a_{ik} \& a_{kj})$ ($k = 1, 2, \dots, n$) is equal to 1. That is, $\alpha_{ij} = 1$ if there is any *two*-segment path joining node i to node j , and $\alpha_{ij} = 0$ otherwise.

To illustrate the computation, let G also represent the adjacency matrix for the graph G above. We then have:

$$G^2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \text{ Boolean } \times \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

This shows that, for example, there is a two-segment path from node 2 to node 4 but none from node 2 to node 3.

The Boolean analog of the algebraic sum $A + A^2$ is the matrix $A \vee A^2$ defined to be that matrix whose element in the i, j position is $a_{ij} \vee a_{ij}^2$. If, then, we form the matrix $G \vee G^2$, its elements will indicate whether in graph G node i can be joined to node j by a path of length 1 or of length 2. That is

$$(G \vee G^2)_{ij} = G_{ij} \vee (G^2)_{ij}.$$

In our example

$$G \vee G^2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \text{ OR } \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

We see that a one- or two-segment path connects any ordered pair of the nodes 1, 2, 4.

If there are n nodes, then the matrix $A \vee A^2 \vee A^3 \vee \dots \vee A^n$ is the *path matrix*. The element in the (i, j) position is 1 if there is a path of any length connecting node i to node j and it is 0 otherwise. (It is sufficient to include paths up to length n because a longer path passing through n nodes must contain a cycle that can be removed, leading to a shorter path.)

If ordinary addition and multiplication are used in forming the powers of the adjacency matrix, the elements of these matrix powers denote the number of paths of given length (corresponding to the power) that join node i to node j .

3.12 EXAMPLE OF AN ALGORITHMIC SOLUTION TO A GRAPH-RELATED PROBLEM—THE MINIMAL-PATH PROBLEM

We return to a problem we have already illustrated—that of finding the shortest path between two nodes in a weighted digraph.

Consider the problem of finding the shortest path from node 1 to node 8 in the digraph in Fig. 3.22:

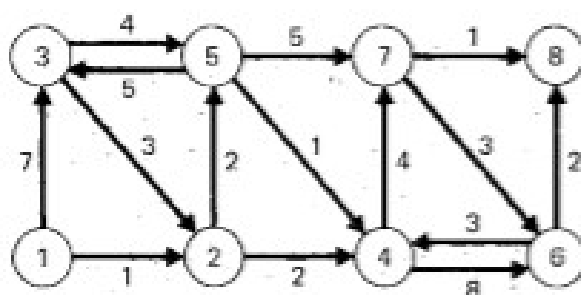


Figure 3.22

We shall solve the problem by following an iterative procedure. The object at each stage of this procedure will be to find additional nodes for which we know the minimal path from node 1. When node 8 appears as one of these additional nodes, we shall have the solution to the problem.

At step 1 we include node 1, for we know the minimal path from node 1 to node 1. It has length 0. Thus, the initial set of nodes for which we know the solution to the minimal path problem is $M_1 = \{1\}$.

We shall continue to generate increasing sets of nodes M_1, M_2, M_3, \dots . At each stage we shall, as follows, add nodes for which we know the solution to finding the shortest path from node 1. After determining all the nodes in set M_k (M_1 offers no problem), we examine all new nodes that are within one path segment of any node in M_k . For each of these new nodes compute the length of a path from node 1 as follows. Suppose the new node N is one path segment from a node P in M_k . Add the length of this path segment to the known minimal distance from node 1 to P . Do this for all possible new nodes. Choose as new members of the set M_{k+1} , where $M_{k+1} \supset M_k$, those nodes that are at the minimal distance, computed in this manner, from node 1. That is, the minimum is to be taken with respect to all of the new nodes that are being considered.

The sets M_1, M_2, \dots are strictly increasing in the number of their elements. Since there are only a finite number of nodes in the digraph, every node must eventually appear in one of the M 's, and we shall then know the solution for that node.

Continuing beyond step 1 we obtain the following sets of nodes for the given problem. We also indicate the minimal path length d_k from node 1 to

each of the *new* nodes in the set M_k (that is, to each node of M_k that is not in M_{k-1}).

$$\begin{array}{llll} M_1 = \{1\}, & d_1 = 0 & M_4 = \{1, 2, 3, 4, 5, 7\}, & d_4 = 7 \\ M_2 = \{1, 2\}, & d_2 = 1 & M_5 = \{1, 2, 3, 4, 5, 7, 8\}, & d_5 = 8 \\ M_3 = \{1, 2, 4, 5\}, & d_3 = 3 & & \end{array}$$

The solution to the given problem of finding the shortest path length from node 1 to node 8 is seen to be $d_5 = 8$.

We leave it to the reader to complete the argument that the stated procedure does what we claim and that it always succeeds in determining the shortest path length between a given node and every other node in the digraph.

APPENDIX EXERCISES

A.1 Are the two graphs drawn in (a) isomorphic? in (b)?



A.2 Draw all trees containing seven nodes, showing only trees that are *not* isomorphic.

A.3 Draw a digraph with nodes identified by the numbers 1, 2, 3, 4, 5, 6, 10, 12, 15, 18, 30 so that there is a directed path from node i to node j if and only if i is a divisor of j . Avoid superfluous path segments.

A.4 Draw a directed (from the root) tree of eight nodes where each node is identified by one of the following strings: AN, ANT, ANTIC, CANT, CANTÉR, CANTALOUPE, FRANTIC, MAN, and a directed path from node i to node j exists if and only if j contains i as a substring.

A.5 Draw a digraph whose eight nodes are identified by the subsets of $\{a, b, c\}$ and in which a directed path from node i to node j exists if and only if i is a subset of j . Avoid superfluous path segments.

A.6 Let x, y be three-bit binary words. Consider the relation R defined by: xRy if and only if x, y differ in precisely one position and the sum of the bits in y exceeds the sum of the bits in x . Represent R by a digraph in which each node corresponds to a three-bit number and a directed path segment extends from node x to node y if and only if xRy .

A.7 Show that in any digraph (i) the sum of the indegrees at all nodes, (ii) the sum of the outdegrees at all nodes, and (iii) the number of path segments are all equal.

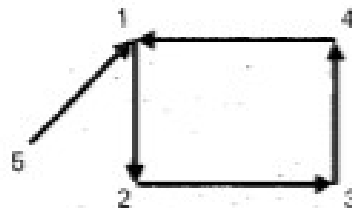
A.8 Show that if there is a directed path in a given digraph from a node u to a node v , then there is a *simple* directed path from u to v .

A.9 Given the "weighted" adjacency matrix M :

$$\begin{pmatrix} 5 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 8 \\ 0 & 4 & 1 & 0 \end{pmatrix}$$

Draw a weighted digraph G of four nodes corresponding to M . Is G connected? Is it strongly connected?

A.10 Given the following digraph G :



Is G connected? Is it strongly connected? Write down the adjacency matrix A for G . Find, using A , that matrix which indicates for each pair of nodes i, j the number of directed paths from i to j that include precisely two path segments. Find the path matrix for G .

A.11 Let the M nodes of a digraph G be N_1, N_2, \dots, N_M . Suppose that the graph is specified by being given the set of path segments $\{(N_i, N_j)\}$ where (N_i, N_j) denotes a path segment directed from node N_i to node N_j (there may be, as we have noted, more than one path segment from N_i to N_j).

- Write a program in any suitable language to give as output the indegree of every node.
- Write a program to form the adjacency matrix of G and to compute the path matrix of G .

A.12 Write a program in any suitable language to find the shortest path between two nodes of a given weighted digraph G . Assume that the input data consists of the weighted adjacency matrix A of G ; that is, $A(I, J)$ is the length of the directed path segment joining node I to node J . However, if $A(I, J) = 0$, this is taken to mean that there is no directed path segment from I to J . Use the method outlined in the text.

REFERENCES

- Berge, C., *The Theory of Graphs and Its Applications*. Translated by Alison Doig. London: Methuen, 1962.
- Bellman, R., K. L. Cooke, and J. A. Lockett, *Algorithms, Graphs, and Computers*. New York: Academic Press, 1970.

3. Date, C. J., *An Introduction to Database Systems*. Reading, Massachusetts: Addison-Wesley, 1975.
4. Harary, F., *Graph Theory*. Reading, Massachusetts: Addison-Wesley, 1969.
5. Harris, B., ed. *Graph Theory and its Applications*. New York: Academic Press, 1970.
6. Kennedy, K., and J. Schwartz, "An Introduction to the Set Theoretical Language SETL." *Comp. and Math. with Appl.*, 1 (1975).
7. Ore, O., *Theory of Graphs*. Providence, Rhode Island: Amer. Math. Soc. Colloq. Publ., 38, 1962.
8. Schwartz, J. T., *Set Theory as a Language for Program Specification and Programming*. New York: Courant Institute of Mathematical Sciences, Lecture Notes.
9. Schwartz, J. T., *An Interim Report on the SETL Project, Installment 1: Generalities*. New York: Courant Institute of Mathematical Sciences, Lecture Notes, February 1973.