Contents

1

Data Structures

1.0 Introduction

It takes time for a society to accept computers. Many societies still resist their use, but computers have passed the test of time and are being accepted at a rapid pace in all aspects of life. Computers have become an integral part of American business. They are rapidly entering the fields of education, inventory control, retail sales, government affairs; they are scanning legal briefs to aid justice, helping law enforcement, making smart judgments on monetary affairs, arranging airline schedules, and so on. These are only a fraction of the ways that computers are being used, and their potential can be bound only by the limitations on man's imagination.

Hardware technologies have played vital roles in our ability to use electronic properties to process information, but the software and data processing aspects have not developed at the same speed. One of the main reasons for the inability of software and data processing development to keep pace with hardware development has been the novelty and complexity of the subject. The rapid expansion of computer applications to unrelated and widely divergent subject areas has introduced tremendous challenge to both computer users and researchers. In the beginning, computers were used to process only numerical information, but the majority of our information is nonnumerical in nature and very little is known about its description and processing. If data relevant to one business are described using certain primitives, then it is not clear whether the same set of primitives can be adequately used to describe the data relevant to another business. Is there any one technique for describing data for all business applications? Are the techniques used for data description in the field of business applications adequate for describing data in the field of education or legal matters, etc.? Very few basic questions associated with nonnumerical information processing have been answered up to now.

The information explosion has certainly not helped the situation. Our knowledge has been doubling every ten years and most of this new knowledge is being recorded in books. Libraries were the first to face the problem and the first to start work in the area of automatic information processing. Pioneering work done by many researchers in the area of library automation laid the foundations of nonnumerical information processing. Most of their work relates to retrieving relevant documents, indexing, and answering some very simple questions contained in the material of the book, etc. The last problem is actually the first attempt to solve complex problems in the area of nonnumerical data processing.

Even before good reliable automated library systems could be developed, computers were used to store complex structured data from many diversified applications. Then computers from different regions were linked to form computer networks so that data could be shared by users of computers in the different regions. Thus, the need for developing techniques for describing data structures in an efficient manner has become more important than ever before.

In the last ten years, many information systems have been developed for storing and retrieving nonnumerical information. These information systems were based on the extension of operating systems which were developed for processing numerical information. Many query languages have been developed for processing existing information-retrieval systems and are widely used, but very little is really known about their capabilities. To develop good query languages, good data descriptions must be developed. Most query languages are not completely dependent on the data description; some depend on other aspects of the computer system. A truly data-independent query language depends on the primitives of the data description alone. In such situations, it may be possible to assess the capabilities of different query languages based on the same data description. Developing good theories for data description, then, is the first step toward laying the foundation for developing powerful query languages, information-retrieval systems, and broadening the applications of computers to new and challenging fields.

In this book we shall use many basic mathematical concepts to describe data, characterize the hardware to store data, and develop techniques for storing, retrieving, and modifying data. The next section will be devoted to basic mathematical concepts which will be used throughout the book.

1.1 Basic Mathematical Concepts

In this section some of the basic mathematical concepts relevant to the materials covered in the book are discussed. In addition, more specific concepts relevant to a chapter are covered in the chapter itself. The basic materials covered in this chapter can be found in many textbooks (see Stoll, 1963; Riordan, 1968; Harary, 1969; Peterson, 1972); hence, unless considered beneficial to the readers, proofs of theorems or derivations of results are not given.

1.1.1 Set Theory

A set is made up of objects called members or elements. Given an element, it can be determined whether it is a member of a set or not. If a is a member of a set A, then it is symbolically represented as

$$a \in A.$$

If a is not a member of A, then it is represented as

$$a \notin A.$$

If $a_1, a_2, \ldots, a_n$ are all the members of the set A, then the set is denoted by

$$\{a_1, a_2, \ldots, a_n\} \equiv A.$$

If a statement $P(x)$ defines a set A, i.e., the elements of A are those objects for which $P(x)$ is true, then it is symbolically represented as

$$\{x \mid P(x)\}.$$

Two sets are equal iff (if and only if) they have the same elements. The equality of two sets A and B is denoted by

$$A = B,$$

and the inequality is denoted by

$$A \neq B.$$

If A and B are two sets, then A is *included in* B is symbolized by

$$A \subseteq B$$

iff each member of A is a member of B. In such a situation, it is customary to say A is a *subset* of B. The same symbolic representation ($A \subseteq B$) can also be stated as B *includes* A. If the set A is included in B but not equal to B, then it is called *properly included* and is denoted by

$$A \subset B.$$

In such cases, A is also called a *proper subset* of B. Among the basic properties of inclusion are

$$A \subseteq B,$$
$$A \subseteq B \quad \text{and} \quad B \subseteq C \Rightarrow A \subseteq C \qquad (\Rightarrow \text{ is the notation for implies}),$$
$$A \subseteq B \quad \text{and} \quad B \subseteq A \Rightarrow A = B.$$

There can be only one set with no elements. It is called the *empty set* and it is denoted by the symbol $\varnothing$. If A has n elements, then the number of subsets of A including the empty subset is 2^n.

The set of all objects which are members of either A or B is called the *union* of the sets A and B. It is symbolized by

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

For example, $\{a_1, b_1, a_2\} \cup \{a_1, b_2\} = \{a_1, b_1, a_2, b_2\}$. The set of all objects which are members of both A and B is called the *intersection* of the sets A and B. It is symbolized by

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

For example,

$$\{a_1, b_1, a_2\} \cap \{a_1, b_2\} = \{a_1\}.$$

Two sets are *disjoint* if their intersection is the empty set, i.e., $A \cap B = \varnothing$. A collection of sets is called a *disjoint collection* iff each distinct pair of its member sets are disjoint. A *partition* of a set A is a disjoint collection of nonempty and distinct subsets of A such that the union of the subsets is equal to A. For example, $\{a_1, a_2\}$, $\{a_3\}$, and $\{b_1, b_2\}$ is a partition of $\{a_1, a_2, a_3, b_1, b_2\}$.

The *absolute complement* of a set A is the set of all elements which do not belong to the set and is denoted by $\overline{A}$. Thus,

$$\overline{A} = \{x \mid x \notin A\}.$$

The *relative complement* of B with respect to A is the set of elements of A which do not belong to B. It is symbolized as

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

The *symmetric difference* of sets A and B is symbolized by $A + B$ and is defined by

$$A + B = (A - B) \cup (B - A).$$

In a certain discussion if all sets under consideration are subsets of a set, say Ω, then Ω is called the *universal set*. Thus $\overline{A} = \Omega - A$.

The associative, commutative, and distributive laws are true for $\cup$ and $\cap$. Consider any subsets A, B, and C; then the following are true:

ASSOCIATIVE LAWS

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

COMMUTATIVE LAWS

$$A \cup B = B \cup A, \qquad A \cap B = B \cap A.$$

DISTRIBUTIVE LAWS

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

These equalities follow immediately:

$$A \cup \varnothing = A, \qquad A \cap \Omega = A;$$
$$A \cup \overline{A} = \Omega, \qquad A \cap \overline{A} = \varnothing.$$

The associative, commutative, and distributive laws can easily be generalized to any finite number of sets. The generalization of the associative and commutative laws are left to the readers. The generalization of the distributive laws are given by

$$A \cup (B_1 \cap B_2 \cap \cdots \cap B_n) = (A \cup B_1) \cap (A \cup B_2) \cap \cdots \cap (A \cup B_n),$$
$$A \cap (B_1 \cup B_2 \cup \cdots \cup B_n) = (A \cap B_1) \cup (A \cap B_2) \cup \cdots \cup (A \cap B_n).$$

There are three more laws which are also of importance in set theory:

IDEMPOTENT LAWS

$$A \cup A = A \qquad \text{and} \qquad A \cap A = A.$$

ABSORPTION LAWS

$$A \cup (A \cap B) = A \qquad \text{and} \qquad A \cap (A \cup B) = A.$$

DeMORGAN LAWS

$$\overline{A \cup B} = \overline{A} \cap \overline{B} \qquad \text{and} \qquad \overline{A \cap B} = \overline{A} \cup \overline{B}.$$

Examples of *self-dual* are

$$\overline{\overline{A}} = A \qquad \text{and} \qquad \text{if } A \cup B = \Omega \text{ and } A \cap B = \varnothing, \text{ then } B = \overline{A}.$$

1.1.2 Relations

If an object x is related to another object y by a *relation* ρ, then symbolically we write $x \rho y$ or $\langle x, y\rangle \in \rho$. Examples of relations are $x = y$, $x < y$, $x \equiv y$, etc. x and y are called the arguments of the relation. This is called a binary relation because it has two arguments. A binary relation can also be considered as an ordered pair. Suppose $x \in X$ and $y \in Y$; then ρ is a subset of the *cartesian product space* $X \times Y$ (see Stoll, 1963). The elements of the relation ρ are all pairs which satisfy $x \rho y$ in $X \times Y$. We call x the first coordinate and y the second coordinate of the ordered pair $\langle x, y\rangle$. A relation is called *transitive* if $x \rho y$ and $y \rho z$ implies $x \rho z$.

Just as the concept of ordered pair can be extended to ordered n-tuples $\langle x_1, x_2, \ldots, x_n\rangle$, similarly the concept of relations can be extended to n-ary relations. The n-ary relation R is a subset of the product space $X_1 \times X_2 \times \cdots \times X_n$ where X_i, $i = 1, 2, \ldots, n$, are sets. The elements of R are ordered n-tuples which satisfy the relation R. X_j is called the jth *domain* of R. For example, if $R \equiv$ the equality of the arguments in a 5-ary space over integers, then the elements of R are (1, 1, 1, 1, 1), (2, 2, 2, 2, 2), (3, 3, 3, 3, 3), ...

Since relations are sets, all the usual operations of set theory are applicable to them. However, relations are not closed under all set theoretic operations; for example, the union of a binary relation and a ternary relation is not a relation and neither is the intersection between them.

There are some operations which are applicable to relations but not to general sets. Given a relation R, a new relation can be obtained by interchanging any two domains. The new relation is called the *permutation* of the previous relation. In general, $n!$ permutation relations can be obtained from a relation with n arguments.

If the domain X_n is eliminated from the relation R and the duplicate elements removed, then the resulting set is a relation in $n - 1$ arguments. It is called the *projection* of R on $X_1 \times X_2 \times \cdots \times X_{n-1}$.

Let $I = \{i_1, i_2, \ldots, i_k\}$ for $k \leq n$ be a list of k indices; then the selection operator Π_I when operated on R gives a k-ary relation $\Pi_I(R)$ whose jth domain is X_{i_j}. The selection operator Π_I can be used for permutation as well as projection.

The concept of joining two relations to obtain another relation is more difficult. It can be defined in a roundabout manner as follows: given a relation R_1 on $X_1 \times X_2 \times \cdots \times X_k$ and another relation R_2 on $X_k \times X_{k+1} \times \cdots \times X_n$, they are joinable if there exists an n-ary relation R_3 on $X_1 \times X_2 \times \cdots \times X_n$ such that $\Pi_{12\cdots k}(R_3) = R_1$ and $\Pi_{k\cdots n}(R_3) = R_2$. Then R_3 is called the join of R_1 and R_2. Joins can also be defined on multiple domains.

Given two relations R_1 and R_2, if there exists a join R_3 of R_1 and R_2 such that $\Pi_{12\cdots k-1, k+1\cdots n}(R_3) = X_k$, then X_k is called the *composition* of R_1 and R_2.

1.1.3 Functions

A *function* is an n-ary relation such that no two distinct members have the same set of first $n-1$ coordinates. Thus, f is a function iff it satisfies the following conditions:

(i) The members of f are ordered tuples.
(ii) If $\langle x_1, x_2, \ldots, x_{n-1}, x_n\rangle$ and $\langle x_1, x_2, \ldots, x_{n-1}, x_n'\rangle$ are members of f, then $x_n = x_n'$.

If $\langle x_1, x_2, \ldots, x_n\rangle \in f$, then it is symbolically written as $\langle x_1, x_2, \ldots, x_{n-1}\rangle f x_n$. If f is a function and $\langle x_1, x_2, \ldots, x_{n-1}, x_n\rangle \in f$ such that $\langle x_1, x_2, \ldots, x_{n-1}\rangle f x_n$, then $\langle x_1, x_2, \ldots, x_{n-1}\rangle$ is called the argument of f. x_n is called the value of f at $\langle x_1, x_2, \ldots, x_{n-1}\rangle$ or the image of $\langle x_1, x_2, \ldots, x_{n-1}\rangle$ under f, etc. The terms mapping and transformation are synonyms for function. The symbol $\langle x_1, x_2, \ldots, x_{n-1}\rangle \rightarrow x_n$ is also used to symbolize the same function. Henceforth the symbol X will denote the argument domain of f, and Y will denote the range of x_n. Examples of a function are $\langle 1, 3, 7\rangle$, $\langle$Dick, Harry, Tom$\rangle$, $\langle$book, pen, student$\rangle$. The image space of a function does not have to be one domain; it may contain multiple domains. The image of a tuple can be a tuple so long as multiple tuples are not mapped into the same tuple. Thus $X \rightarrow Y$ is a function when $X = \{\langle x_1, x_2, \ldots, x_{n_1}\rangle\}$ and $Y = \{\langle y_1, y_2, \ldots, y_{n_2}\rangle\}$.

A function f is *into* Y iff the range of f, say R_f, is a subset of Y and f is *onto* Y iff $R_f = Y$. The symbol $f(x)$ also denotes the image of x under f.

A function is called *one-to-one* if it maps distinct elements onto distinct elements, i.e.,

$$x \neq x' \Rightarrow f(x) \neq f(x') \qquad \text{and} \qquad f(x) = f(x') \Rightarrow x = x'.$$

An example is $f(x) = x + 1$, where x is a real number.

The *composition* of two functions f and g written symbolically as

$$g \circ f$$

is the set

$$\{\langle x, z\rangle \mid \text{there is a } y \text{ such that } xfy \text{ and } ygz\}.$$

Functional composition is not a commutative operation, but it is an associative operation. The function which is defined by composition from the functions $f_1, f_2, \ldots, f_n$ is a unique function and is designated by

$$f_1 \circ f_2 \circ \cdots \circ f_n.$$

If f is a one-to-one function, then the function resulting from f by interchanging the argument and the value coordinates of each member of f is called the *inverse function.* It is symbolized as

$$f^{-1}.$$

If f^{-1} exists, then its domain is the range of f, its range is the domain of f, and $x = f^{-1}(y)$ iff $y = f(x)$.

1.1.4 Binominal Coefficients

The binominal expansion discovered by Newton is given by

$$(1 + x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k,$$

where $\binom{n}{k}$ is the number of ways in which k objects can be chosen from n objects and is called the binominal coefficient. It can be expressed as ratios of products or factorials as

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots(1)} = \frac{n!}{k!(n-k)!} \quad \text{for integer } n \geq \text{integer } k > 0.$$

The binominal coefficient can be decomposed as

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

Some basic equalities and sums of binominal coefficients which are useful for the materials covered in this book are

$$\binom{n}{k} = \binom{n}{n-k}, \qquad \binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1},$$

$$\binom{n}{k}\binom{k}{l} = \binom{n}{l}\binom{n-l}{k-l} = \binom{n}{k-l}\binom{n-k+l}{l} \quad \text{for integer } l \geq 0,$$

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n,$$

$$\sum_{i=0}^{m} \binom{n+i}{i} = \binom{n+m+1}{m} \qquad \text{for integer } m \geq 0,$$

$$\binom{n}{k} = \sum_{i=0}^{K} \binom{n-1-i}{k-i}, \qquad \text{where} \quad K = \min(k, n-1),$$

$$\sum_{i=0}^{m} \binom{i}{k} = \binom{m+1}{k+1},$$

$$\sum_{i=m}^{\min(n,\, l+m)} \binom{n}{i}\binom{l}{m-i} = \binom{n+l}{m}, \qquad \sum_{i=0}^{\min(n,\, l-m)} \binom{n}{i}\binom{l}{m+i} = \binom{n+l}{n+m}.$$

1.1.5 Graph Theory

Graph theory is the branch of mathematics which studies the properties of diagrams which can be constructed on a plane using points and line segments. A point is referred to as a *vertex* or *node* or *point*. A line is used to connect two nodes. A line with ends connecting the same node is called a *loop*. If the line does not have any sense of direction associated with it, then it is referred to as an *edge*. A line is called an *arc* if it has a sense of direction associated with it. Diagrams which are constructed using nodes and edges are called *undirected graphs*. A graph can be specified by identifying its node collection $\{v_i\}$ and edge collection $\{e_i\}$. Figure 1.1.5.1 is an example of an undirected graph.

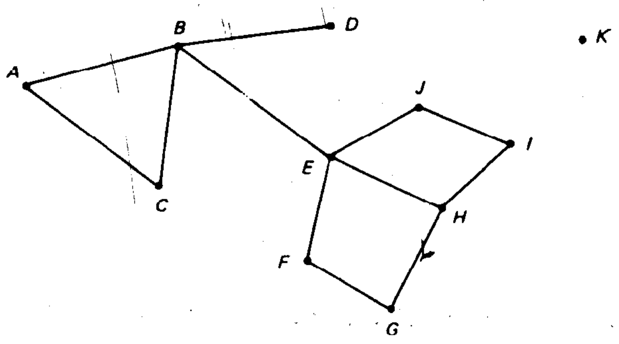

FIG. 1.1.5.1. An undirected graph.

In Fig. 1.1.5.1, A, B, C, D, E, F, G, H, I, J, and K are the nodes of the nondirected graph. The lines AB, AC, BC, BD, BE, EF, EJ, EH, FG, GH, HI, and IJ are the edges of the undirected graph. K is an isolated point of the graph.

The edges of a graph may have labels as in Fig. 1.1.5.2. Such graphs are called *lagraphs*.

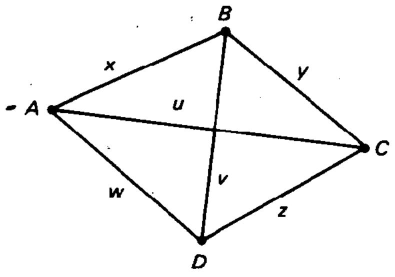

Fig. 1.1.5.2. A graph to illustrate labels and intersection of edges.

In Fig. 1.1.5.2, the edges u and v intersect, but their intersection point is not a node of the graph.

A graph with arcs is called a *directed graph* or *digraph*. Figure 1.1.5.3 is an example of a directed graph.

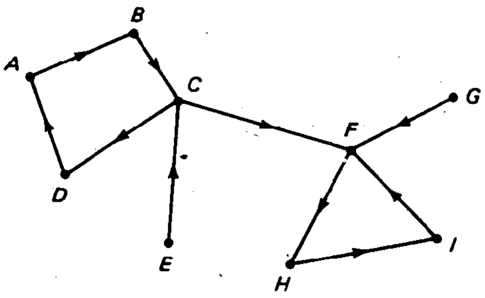

FIG. 1.1.5.3. Directed graph (digraph).

In Fig. 1.1.5.3 the arcs AB, BC, DA, CD, EC, CF, FH, HI, IF, and GF are elements of the graph, but it would be wrong to say that arcs like BA or AD are also elements of this graph.

If a graph has multiple edges or arcs connecting two specified nodes, then it is called *multigraph.* The two diagraphs in Fig. 1.1.5.4 illustrate multigraphs.

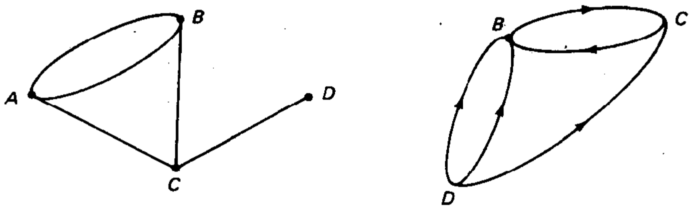

FIG. 1.1.5.4. Multigraphs.

In multigraphs, loops with one node are not allowed. If such loops appear in a graph, it is called *pseudograph.* Figure 1.1.5.5 gives an example of a pseudograph.

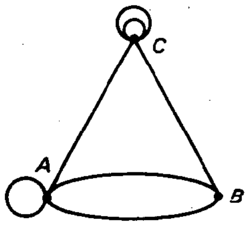

FIG. 1.1.5.5. Pseudograph.

The *valency* or *degree* of a node in a graph is the number of edges incident on the node. The valency of A in Fig. 1.1.5.3 is 2. There are many interesting theorems that can be mathematically proven for graphs. One such theorem due to Euler states: "The sum of the valencies of the nodes of a graph is equal to twice the number of lines." In a directed graph the number of arcs coming into a node is called its *in-degree* and the number of arcs going away from the node is called its *out-degree.*

We have given diagrammatic representations of different types of graphs, but diagrams should not be mistaken for definitions of graphs. A graph is distinguished by its nodes and lines only. Two graphs are equivalent if they have the same nodes and lines. Consider a graph with nodes $\{A, B, C, D\}$

and edges $\{AB, BC, CD\}$. Then the two diagrams of this graph given in Fig. 1.1.5.6 are equivalent. If a piece of a graph is separated from a graph, then each piece is referred to as a *subgraph*. A subgraph consists of a subset of the vertices of the original graph and all the edges or arcs connecting these vertices which are present in the original graph. Figure 1.1.5.7 illustrates two subgraphs obtained from the graph presented in Fig. 1.1.5.3.

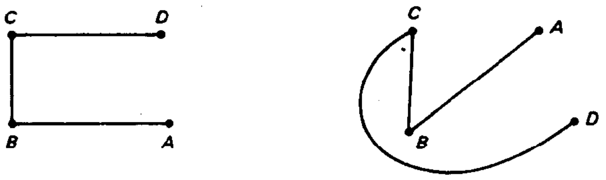

FIG. 1.1.5.6. Equivalent graphs.

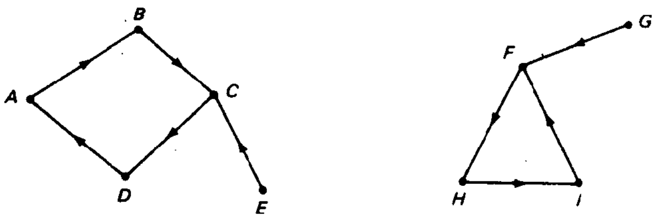

FIG. 1.1.5.7. Two subgraphs of Fig. 1.1.5.3.

A part of a graph is called *partial graph*. Figure 1.1.5.8 gives a partial graph of Fig. 1.1.5.3.

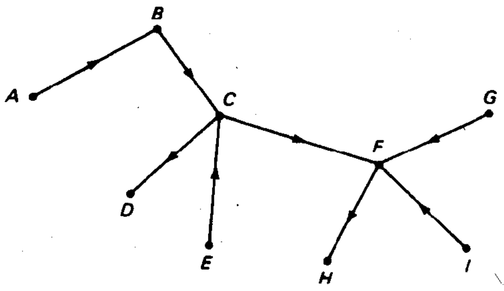

FIG. 1.1.5.8. A partial graph of Fig. 1.1.5.3.

A *complete graph* is a graph in which there is a line between every two pairs of nodes. Figure 1.1.5.9 illustrates a complete graph.

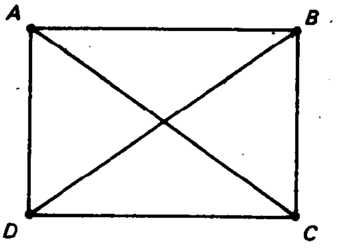

FIG. 1.1.5.9. Complete graph.

In a complete graph with n nodes, each node has a valency of $n-1$; hence there are $\binom{n}{2}$ lines. The *complement graph* of a given graph is obtained by constructing a complete graph on the nodes and then deleting all the lines of the original graph. Figure 1.1.5.10 illustrates a graph and its complement.

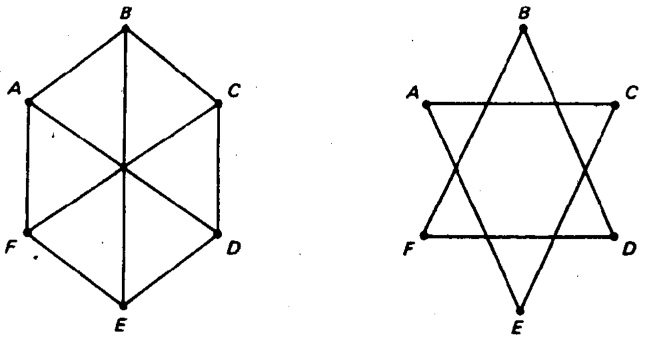

FIG. 1.1.5.10. A graph and its complement.

A graph G is called a *bigraph* or *bipartite graph* if its vertices can be partitioned into two subsets V_1 and V_2 such that every line of G joins V_1 with V_2. Figure 1.1.5.11 is an example of a bigraph.

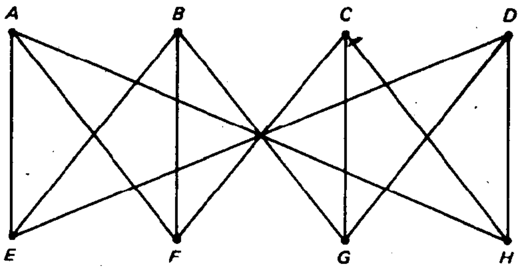

FIG. 1.1.5.11. Bigraph.

Let $N = \{S_1, S_2, \ldots, S_n\}$ be a collection of distinct nonempty subsets of a set S, whose union is S. The *intersection graph* of N is obtained by considering each element of N as a node and defining lines between S_i and S_j whenever $S_i \cap S_j \neq \emptyset$ for $i \neq j$. The intersection graph has many important applications in organizing information in an optimum manner in a computer storage.

A sequence of edges which form a route between two nodes is called an *edge sequence*. In Fig. 1.1.5.1 the edges AB, BE, and EH form an edge sequence. The multiplicity of an edge is the number of times it appears in an edge sequence. A *cycle* is a closed edge sequence. In Fig. 1.1.5.1 ABC is a cycle. An edge sequence with multiplicity one is called a *chain*.

A route in a digraph is an *arc sequence*. In an arc sequence the terminal node of one arc is the initial node of another arc, i.e., we follow the arrow to go from the "start" of the arc sequence to the "end." A *path* is an arc sequence and contains no duplicate arc. A path can be labeled by writing the nodes in it in a sequential manner. In Fig. 1.1.5.3 $ABCFH$ is a path, but $ABCE$ is not a path. An *elementary path* repeats no nodes. A *loop* is an arc sequence with the same "start" and "finish." In Fig. 1.1.5.3 $FHIF$ is a loop.

A *tree* is a graph with a chain between any two nodes but without any cycles. Examples of trees are given in Fig. 1.1.5.12. The number of edges in a tree is one less than the number of nodes. If one edge is deleted from a tree, the graph is no longer connected. If an edge is added between two nonadjacent nodes, then exactly one cycle is formed. If the edges are replaced by arcs, we get a *directed tree*. The tree with $2n - 1$ nodes in which $n - 2$ nodes have valency 3 and one node has valency 2 is called a *binary tree*. These trees, especially the directed ones, play an important role in searching information in a computer.

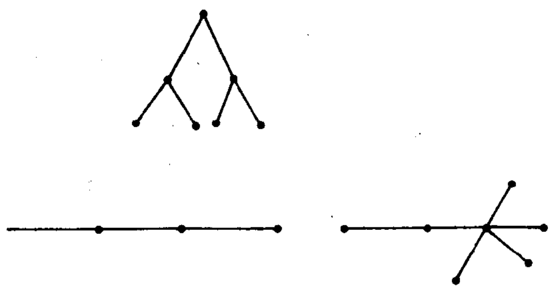
FIG. 1.1.5.12. Trees.

A *network* is a graph with values assigned to each line. In most situations these values are real numbers, but in this book we shall also be dealing with nonnumerical values.