GRAPHS: THEORY AND ALGORITHMS

K. THULASIRAMAN
M. N. S. SWAMY
Concordia University
Montreal, Canada

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York / Chichester / Brisbane / Toronto / Singapore

This page intentionally left blank

GRAPHS: THEORY AND ALGORITHMS

This page intentionally left blank

GRAPHS: THEORY AND ALGORITHMS

K. THULASIRAMAN
M. N. S. SWAMY
Concordia University
Montreal, Canada

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York / Chichester / Brisbane / Toronto / Singapore

Library of Congress Cataloging in Publication Data:

Thulasiraman, K.

Graphs: theory and algorithms / K. Thulasiraman and M.N.S. Swamy.

p. cm.

"A Wiley-Interscience publication."

Includes bibliographical references and index.

ISBN 0-471-51356-3

1. Graph theory. 2. Electric networks. 3. Algorithms.

I. Swamy, M. N. S. II. Title.

QA166.T58 1992 91-34930

511'.5--dc20 CIP

10 9 8 7 6 5 4 3 2 1

Dedicated to Our Parents and Teachers

अज्ञानतिमिरान्धस्य ज्ञानाञ्जनशलाकया ।
चक्षुरुन्मीलितं येन तस्मै श्रीगुरवे नमः ॥

from *Vishwasara Tantra*

Salutations to the Guru who with the collyrium stick of knowledge has opened the eyes of one blinded by the disease of ignorance.

This page intentionally left blank

CONTENTS

This page intentionally left blank

PREFACE

In the past two decades graph theory has come to stay as a powerful analytical tool in the understanding and solution of large complex problems that arise in the study of engineering, computer, and communication systems. While its origin is traced to Euler's solution in 1735 of the Königsberg bridge problem, its first application to a problem in physical science did not occur until 1847, when Kirchhoff developed the theory of trees for its application in the study of electrical networks. The elegance with which the graph of an electrical network captures the structural relationships between the voltage and current variables of the network has led to equally elegant contributions to electrical network theory. One such condition is Tellegen's theorem, the application of which in the computation of network sensitivities is now well recognized. The theory of network flows developed by Ford and Fulkerson in 1956 was the first major application of graph theory to operations research. This theory provides the main link between graph theory and operations research and continues to be a fascinating topic of further research. Computer and communication systems are among the recent additions to the growing list of application areas of graph theory. Motivated by applications in the design of interconnection networks for these systems, in recent years there has been a great deal of interest in the design of graphs having specified topological properties such as distance, connectivity, and regularity. Fascinated by the challenges encountered in the design of efficient algorithms for graph problems, theoretical computer scientists have developed in the past two decades a large number of interesting and deep graph algorithms adding to the richness of graph theory. Theoretical computer scientists have also identified the class of graph problems for which "efficient" algorithms are not likely to exist, giving birth to the theory of NP-Completeness. This is indeed a significant contribution of computer science to graph theory.

Every time a new area of application of graph theory emerged, the need arose for the introduction and study of new concepts or a further study of several known concepts. This continuous interaction has immensely contributed to the recent explosion of graph theory, which was fairly dormant for more than a century after its origin. Thus graph theory is now a vast subject with several fascinating branches of its own: enumerative graph theory, extremal graph theory, random graph theory, algorithmic graph theory, and so on.

As its name implies, this book is on graph theory and graph algorithms. It is addressed to students in engineering, computer science, and mathematics. Our choice of topics has been motivated by their relevance to applications. Thus we attempt to provide a unified and an in-depth treatment of those topics in graph theory and graph algorithms that we believe to be fundamental in nature and that occur in most applications. Broadly speaking, the book may be considered as consisting of two parts dealing with graph theory and graph algorithms in that order.

In the first ten chapters we discuss the theory of graphs. The topics discussed include trees, circuits, cutsets, Hamiltonian and Eulerian graphs, directed graphs, matrices of a graph, planarity, connectivity, matching, and coloring. We have also included an introduction to matroid theory. Among the matroid topics presented are Minty's self-dual axiom system, which makes obvious the duality between circuits and cutsets of a graph, the arc coloring lemma, the greedy algorithm, and its intimate relationship with matroids.

The last two chapters of the book deal with graph algorithms. In Chapter 11 we discuss several algorithms which are basic in the sense that they serve as building blocks in designing more complex algorithms. In most cases the algorithms of Chapter 11 are based on results and concepts presented in earlier chapters. In certain cases we also introduce and discuss new concepts such as branching and graph reducibility. In Chapter 12 we develop the theory of network flows. We start with the maximum flow minimum cut theorem of Ford and Fulkerson and then proceed to develop several algorithms for the maximum flow problem, culminating with the recent work of Goldberg and Tarjan. In this chapter we also show how the network flow technique can be used to develop connectivity and matching algorithms as well as prove Menger's theorems on connectivities. We conclude Chapter 12 with a brief introduction to the theory of NP-Completeness. While developing the algorithms of Chapters 11 and 12 we pay particular attention to the proof of correctness and complexity analysis of the algorithms.

The book can be used to organize different courses to suit the needs of different groups of students. The first ten chapters contain adequate material for a one-semester course on graph theory at the senior or beginning graduate level. The authors have taught for several years a course on graph theory with system applications based on the first seven chapters and a selection of topics from the remaining chapters. The last two chapters and appropriate background material selected from the other chapters can serve as the core of a course on algorithmic graph theory. These two chapters can also serve as supplemental material for a general course on design and analysis of algorithms.

Several colleagues and students have assisted us in the writing of this book. Raghu Prasad Chalasani, Concordia University; Joseph Cheriyan, Cornell University; Anindya Das, University of Montreal; Andrew Goldberg, Stanford University; R. Jayakumar, Concordia University; V. Krishnamoorthy, Anna University, Madras (India); and N. Srinivasan, University of Madras deserve special thanks. We are greatful to Anindya Das, Joseph Cheriyan, and Andrew Goldberg for their careful reading of the last chapter of the book and drawing our attention to recent developments on the maximum flow problem.

It is a pleasure to thank the following organizations for their support to our research leading to the preparation of the book: Natural Sciences and Engineering Research Council of Canada; Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR), Quebec; Bell Northern Research Laboratory, Ottawa; Centre de Recherche Informatique de Montréal, Montréal; German National Science Foundation; and the Japan Society for Promotion of Science.

Finally we thank our wives—Santha Thulasiraman and Leela Swamy—and our children for their patience and understanding during the entire period of our efforts.

K. THULASIRAMAN
M. N. S. SWAMY

It is probably fair to say, and has been said before by many others, that graph theory began with Euler's solution in 1735 of the class of problems suggested to him by the Königsberg bridge puzzle. But had it not started with Euler, it would have started with Kirchhoff in 1847, who was motivated by the study of electrical networks; had it not started with Kirchhoff, it would have started with Cayley in 1857, who was motivated by certain applications to organic chemistry, or perhaps it would have started earlier with the four-color map problem, which was posed to De Morgan by Guthrie around 1850. And had it not started with any of the individuals named above, it would almost surely have started with someone else, at some other time. For one has only to look around to see "real-world graphs" in abundance, either in nature (trees, for example) or in the works of man (transportation networks, for example). Surely someone at some time would have passed from some real-world object, situation, or problem to the abstraction we call graphs, and graph theory would have been born.

D. R. Fulkerson
(From Preface to *Studies in Graph Theory*, *Part II*,
The Mathematical Association of America, 1975)

CHAPTER 1

BASIC CONCEPTS

We begin our study with an introduction in this chapter to several basic concepts in the theory of graphs. A few results involving these concepts will be established. These results, while illustrating the concepts, will also serve to introduce the reader to certain techniques commonly used in proving theorems in graph theory.

1.1 SOME BASIC DEFINITIONS

A *graph* $G = (V, E)$ consists of two sets: a finite set V of elements called *vertices* and a finite set E of elements called *edges*. Each edge is identified with a pair of vertices. If the edges of a graph G are identified with ordered pairs of vertices, then G is called a *directed* or an *oriented* graph. Otherwise G is called an *undirected* or a *nonoriented* graph. Our discussions in the first four chapters of this book are concerned with undirected graphs.

We use the symbols $v_1, v_2, v_3, \ldots$ to represent the vertices and the symbols $e_1, e_2, e_3, \ldots$ to represent the edges of a graph. The vertices v_i and v_j associated with an edge e_l are called the *end vertices* of e_l. The edge e_l is then denoted as $e_l = (v_i, v_j)$. Note that while the elements of E are distinct, more than one edge in E may have the same pair of end vertices. All edges having the same pair of end vertices are called *parallel edges*. Further, the end vertices of an edge need not be distinct. If $e_l = (v_i, v_i)$, then the edge e_l is called a *self-loop* at vertex v_i. A graph is called a *simple graph* if it has no parallel edges or self-loops. A graph G is of *order n* if its vertex set has n elements.

A graph with no edges is called an *empty graph*. A graph with no vertices (and hence no edges) is called a *null graph*.

Pictorially a graph can be represented by a diagram in which a vertex is represented by a dot or a circle and an edge is represented by a line segment connecting the dots or the circles, which represent the end vertices of the edge. For example, if

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

and

$$E = \{e_1, e_2, e_3, e_4, e_5\}\ ,$$

such that

$$e_1 = (v_1, v_2)\ ,$$
$$e_2 = (v_1, v_4)\ ,$$
$$e_3 = (v_5, v_6)\ ,$$
$$e_4 = (v_1, v_2)\ ,$$
$$e_5 = (v_5, v_5)\ ,$$

then the graph $G = (V, E)$ is represented as in Fig. 1.1. In this graph e_1 and e_4 are parallel edges and e_5 is a self-loop.

An edge is said to be *incident on* its end vertices. Two vertices are *adjacent* if they are the end vertices of some edge. If two edges have a common end vertex, then these edges are said to be *adjacent*.

For example, in the graph of Fig. 1.1, edge e_1 is incident on vertices v_1 and v_2; v_1 and v_4 are two adjacent vertices, while e_1 and e_2 are two adjacent edges.

The number of edges incident on a vertex v_i is called the *degree* of the vertex, and it is denoted by $d(v_i)$. Sometimes the degree of a vertex is also

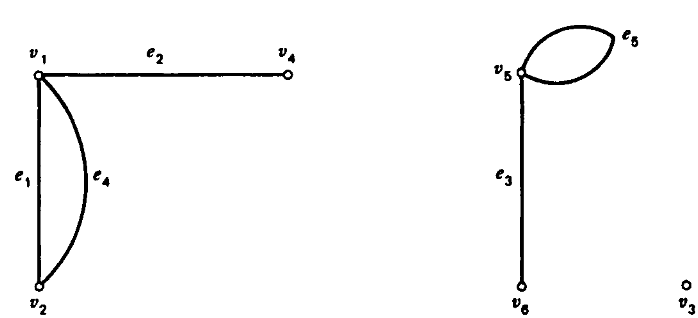

Figure 1.1. Graph $G = (V, E)$. $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$; $E = \{e_1, e_2, e_3, e_4, e_5\}$.

referred to as its *valency*. A vertex of degree 1 is called a *pendant vertex*. The only edge incident on a pendant vertex is called a *pendant edge*. A vertex of degree 0 is called an *isolated vertex*. By definition, a self-loop at a vertex v_i contributes 2 to the degree of v_i. $\delta(G)$ and $\Delta(G)$ denote, respectively, the minimum and maximum degrees in G.

In the graph G of Fig. 1.1

$$\begin{aligned} d(v_1) &= 3\,, \\ d(v_2) &= 2\,, \\ d(v_3) &= 0\,, \\ d(v_4) &= 1\,, \\ d(v_5) &= 3\,, \\ d(v_6) &= 1\,. \end{aligned}$$

Note that v_3 is an isolated vertex, v_4 and v_6 are pendant vertices, and e_2 is a pendant edge. For G it can be verified that the sum of the degrees of the vertices is equal to 10, whereas the number of edges is equal to 5. Thus the sum of the degrees of the vertices of G is equal to twice the number of edges of G and hence an even number. It may be further verified that in G the number of vertices of odd degree is also even. These interesting results are not peculiar to the graph of Fig. 1.1. In fact, they are true for all graphs as the following theorems show.

Theorem 1.1. The sum of the degrees of the vertices of a graph G is equal to $2m$, where m is the number of edges of G.

Proof. Since each edge is incident on two vertices, it contributes 2 to the sum of the degrees of the graph G. Hence all the edges together contribute $2m$ to the sum of the degrees of G. ■

Theorem 1.2. The number of vertices of odd degree in any graph is even.

Proof. Let the number of vertices in a graph G be equal to n. Let, without any loss of generality, the degrees of the first r vertices $v_1, v_2, \ldots, v_r$ be even and those of the remaining $n - r$ vertices be odd. Then

$$\sum_{i=1}^{n} d(v_i) = \sum_{i=1}^{r} d(v_i) + \sum_{i=r+1}^{n} d(v_i)\,. \tag{1.1}$$

By Theorem 1.1, the sum on the left-hand side of (1.1) is even. The first sum on the right-hand side is also even because each term in this sum is even. Hence the second sum on the right-hand side should be even. Since each term in this sum is odd, it is necessary that there be an even number of terms in this sum. In other words, $n - r$, the number of vertices of odd degree, should be even. ■

1.2 SUBGRAPHS AND COMPLEMENTS

Consider a graph $G = (V, E)$. $G' = (V', E')$ is a *subgraph* of G if V' and E' are, respectively, subsets of V and E such that an edge (v_i, v_j) is an E' only if v_i and v_j are in V'. G' will be called a *proper subgraph* of G if either E' is a proper subset of E or V' is a proper subset of V. If all the vertices of a graph G are present in a subgraph G' of G, then G' is called a *spanning subgraph* of G.

For example, consider the graph G shown in Fig. 1.2*a*. The graph G' shown in Fig. 1.2*b* is a subgraph of G. Its vertex set is $\{v_1, v_2, v_4, v_5\}$. In fact, it is a proper subgraph of G. The graph G'' of Fig. 1.2*c* is a spanning subgraph of G.

Some of the vertices in a subgraph may be isolated vertices. For example, the graph G''' shown in Fig. 1.2*d* is a subgraph of G with an isolated vertex.

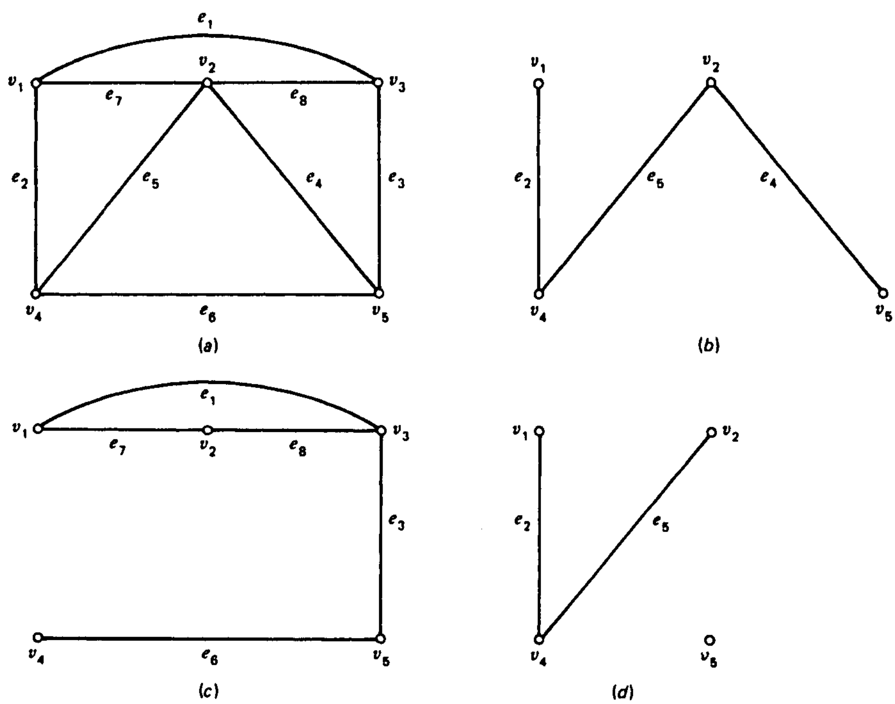

Figure 1.2. A graph and some of its subgraphs. (*a*) Graph G. (*b*) Subgraph G'. (*c*) Subgraph G''. (*d*) Subgraph G'''.

If a subgraph $G' = (V', E')$ of a graph G has no isolated vertices, then it can be seen from the definition of a subgraph that every vertex in V' is the end vertex of some edge in E'. Thus in such a case, E' uniquely specifies V' and hence the subgraph G'. The subgraph G' is then called the *induced subgraph of G on the edge set E'* (or simply *edge-induced subgraph of G*) and is denoted as $\langle E' \rangle$.

Note that the vertex set V' of $\langle E' \rangle$ is the smallest subset of V containing all the end vertices of the edges in E'. The subgraphs G' and G'' of Fig. 1.2*b* and *c* are edge-induced subgraphs of the graph G of Fig. 1.2*a*, whereas G''' shown in Fig. 1.2*d* is not an edge-induced subgraph.

Next we define a vertex-induced subgraph.

Let V' be a subset of the vertex set V of a graph $G = (V, E)$. Then the subgraph $G' = (V', E')$ is the *induced subgraph of G on the vertex set V'* (or simply *vertex-induced subgraph of G*) if E' is a subset of E such that edge (v_i, v_j) is in E' if and only if v_i and v_j are in V'. In other words, if v_i and v_j are in V', then every edge in E having v_i and v_j as its end vertices should be in E'. Note that, in this case, V' completely specifies E' and thus the subgraph G'. Hence the vertex-induced subgraph $G' = (V', E')$ is denoted simply as $\langle V' \rangle$. As an example, the graph shown in Fig. 1.3 is a vertex-induced subgraph of the graph G of Fig. 1.2*a*.

Note that the edge set E' of the vertex-induced subgraph on the vertex set V' is the largest subset of E such that the end vertices of all of its edges are in V'.

Unless otherwise stated, an *induced subgraph* will refer to a vertex-induced subgraph.

A subgraph G' of a graph G is said to be a *maximal subgraph* of G with respect to some property P if G' has the property P and G' is not a proper subgraph of any other subgraph of G having the property P.

A subgraph G' of a graph G is said to be a *minimal subgraph* of G with respect to some property P if G' has the property P and no subgraph of G having the property P is a proper subgraph of G'.

Maximal and *minimal subsets* of a set with respect to a property are defined in a similar manner.

For example, the vertex set V' of an edge-induced subgraph $\langle E' \rangle$ of a graph $G = (V, E)$ is a minimal subset of V containing the end vertices of all

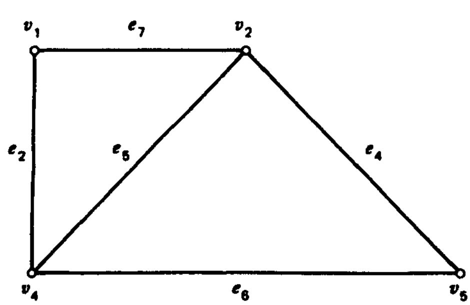

Figure 1.3. A vertex-induced subgraph of the graph G of Fig. 1.2*a*.

the edges of E'. On the other hand, the edge set E' of a vertex-induced subgraph $\langle V' \rangle$ is the maximal subset of E such that the end vertices of all of its edges are in V'.

Later we shall see that a "component" (Section 1.4) of a graph G is a maximal "connected" subgraph of G, and a "spanning tree" (Chapter 2) of a connected graph G is a minimal "connected" spanning subgraph of G.

Next we define the complement of a graph.

Graph $\overline{G} = (V, E')$ is called the *complement* of a simple graph $G = (V, E)$ if the edge (v_i, v_j) is in E' if and only if it is not in E. In other words two vertices v_i and v_j are adjacent in $\overline{G}$ if and only if they are not adjacent in G. A graph and its complement are shown in Fig. 1.4. As another example, consider the graph G shown in Fig. 1.5a. In this graph there is an edge between every pair of vertices. Hence in the complement $\overline{G}$ of G there will

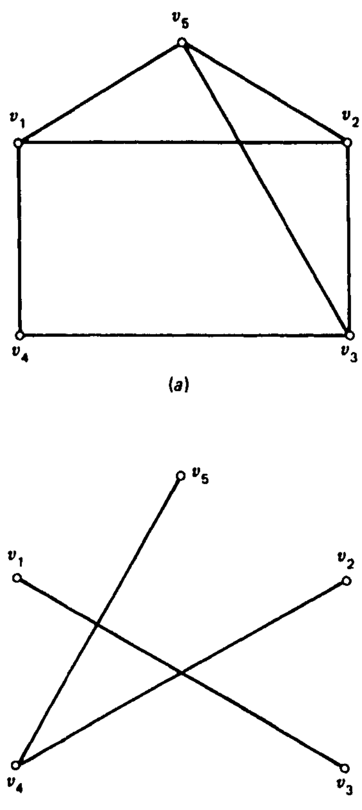

Figure 1.4. A graph and its complement. (a) Graph G. (b) Graph $\overline{G}$, complement of G.

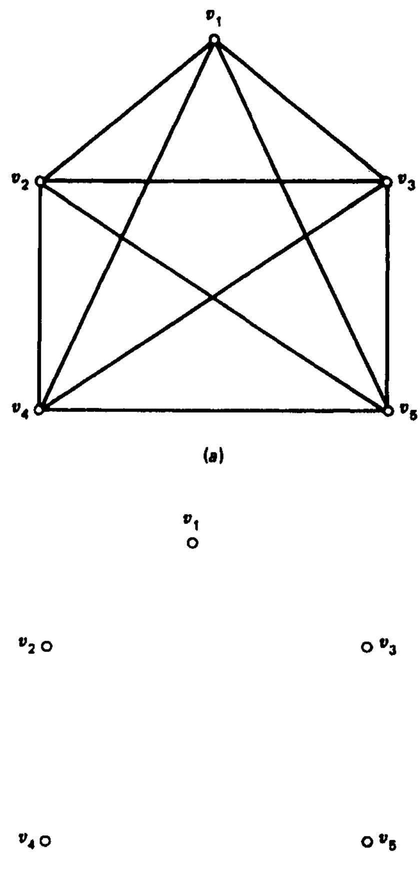

Figure 1.5. A graph and its complement. (a) Graph G. (b) Graph $\overline{G}$, complement of G.

be no edge between any pair of vertices; that is, $\overline{G}$ will contain only isolated vertices. This is shown in Fig. 1.5*b*.

Let $G' = (V', E')$ be a subgraph of a graph $G = (V, E)$. The subgraph $G'' = (V, E - E')$ of G is called the *complement of G' in G*. For example, in Fig. 1.2, subgraph G'' is the complement of G' in the graph G.

The following example illustrates some of the ideas presented thus far. Suppose we want to prove the following:

At any party with six people there are three mutual acquaintances or three mutual nonacquaintances.

Representing people by the vertices of a graph and acquaintance relationship among the people by edges connecting the corresponding vertices, we can see that the above assertion can also be stated as follows:

In any simple graph G with six vertices there are three mutually adjacent vertices or three mutually nonadjacent vertices.

In view of the definition of the complement of a graph, we see that the above statement is equivalent to the following:

For any simple graph G with six vertices, G or $\overline{G}$ contains three mutually adjacent vertices.

To prove this, we may proceed as follows:

Consider any vertex v of a simple graph G with six vertices. Note that if v is not adjacent to three vertices in G, then it will be adjacent to three vertices in $\overline{G}$. So, without any loss of generality, we may assume that, in G, v is adjacent to some three vertices v_1, v_2, and v_3. If any two of these vertices, say v_1 and v_2, are adjacent in G, then the vertices v, v_1, and v_2 are mutually adjacent in G, and the assertion is proved.

If no two of the three vertices v_1, v_2, and v_3 are adjacent in G, then it means that v_1, v_2, and v_3 are mutually nonadjacent in G. Hence, by the definition of a complement, the vertices v_1, v_2, and v_3 are mutually adjacent in $\overline{G}$, and the assertion is again proved.

1.3 WALKS, TRAILS, PATHS, AND CIRCUITS

A *walk* in a graph $G = (V, E)$ is a finite alternating sequence of vertices and edges $v_0, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_k$ beginning and ending with vertices such that v_{i-1} and v_i are the end vertices of the edge e_i, $1 \le i \le k$. Alternately, a walk can be considered as a finite sequence of vertices $v_0, v_1, v_2, \ldots, v_k$, such that (v_{i-1}, v_i), $1 \le i \le k$, is an edge in the graph G. This walk is usually called a v_0–v_k walk with v_0 and v_k referred to as the *end* or *terminal vertices* of this walk. All other vertices are *internal vertices* of this walk. Note that in a walk, edges and vertices can appear more than once.

A walk is *open* if its end vertices are distinct; otherwise it is *closed*.

In the graph G of Fig. 1.6, the sequence $v_1, e_1, v_2, e_2, v_3, e_8, v_6, e_9, v_5, e_7, v_3, e_{11}, v_6$ is an open walk, whereas the sequence $v_1, e_1, v_2, e_3, v_5, e_7, v_3, e_2, v_2, e_1, v_1$ is a closed walk.

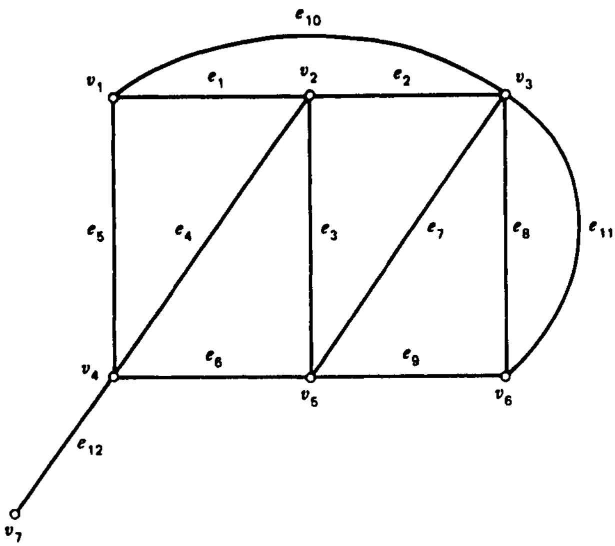

Figure 1.6. Graph G.

A walk is a *trail* if all its edges are distinct. A trail is *open* if its end vertices are distinct; otherwise, it is *closed*. In Fig. 1.6, $v_1, e_1, v_2, e_2, v_3, e_8, v_6, e_{11}, v_3$ is an open trail, whereas $v_1, e_1, v_2, e_2, v_3, e_7, v_5, e_3, v_2, e_4, v_4, e_5, v_1$ is a closed trail.

An open trail is a *path* if all its vertices are distinct.

A closed trail is a *circuit* if all its vertices except the end vertices are distinct.

For example, in Fig. 1.6 the sequence v_1, e_1, v_2, e_2, v_3 is a path, whereas the sequence $v_1, e_1, v_2, e_3, v_5, e_6, v_4, e_5, v_1$ is a circuit.

An edge of a graph G is said to be a *circuit edge* of G if there exists a circuit in G containing the edge. Otherwise the edge is called a *noncircuit edge*. In Fig. 1.6, all edges except e_{12} are circuit edges.

The number of edges in a path is called the *length of the path*. Similarly the *length of a circuit* is defined.

A path is *even* if it is of even length; otherwise it is *odd*. Similarly even and odd circuits are defined.

The *distance* between two vertices u and v in G, denoted by $d(u, v)$, is the length of the shortest u–v path in G. If no such path exists, then we define $d(u, v)$ to be infinite. The *diameter* of G, denoted by diam(G), is the maximum distance between any two vertices of G.

The following properties of paths and circuits should be noted:

1. In a path the degree of each vertex that is not an end vertex is equal to 2; the end vertices have degrees equal to 1.
2. In a circuit every vertex is of degree 2, and so of even degree. The converse of this statement, namely, the edges of a subgraph in which

every vertex is of even degree form a circuit, is not true. A more general question is discussed in Chapter 3.

3. In a path the number of vertices is one more than the number of edges, whereas in a circuit the number of edges is equal to the number of vertices.

1.4 CONNECTEDNESS AND COMPONENTS OF A GRAPH

An important concept in graph theory is that of connectedness.

Two vertices v_i and v_j are said to be *connected* in a graph G if there exists a v_i–v_j path in G. A vertex is connected to itself.

A graph G is *connected* if there exists a path between every pair of vertices in G.

For example, the graph of Fig. 1.6 is connected.

Consider a graph $G = (V, E)$ which is not connected. Then the vertex set V of G can be partitioned[†] into subsets $V_1, V_2, \ldots, V_p$ such that the vertex-induced subgraphs $\langle V_i \rangle$, $i = 1, 2, \ldots, p$, are connected and no vertex in subset V_i is connected to any vertex in subset V_j, $j \neq i$. The subgraphs $\langle V_i \rangle$, $i = 1, 2, \ldots, p$, are called the *components* of G. It may be seen that a component of a graph G is a maximal connected subgraph of G; that is, a component of G is not a proper subgraph of any other connected subgraph of G.

For example, the graph G of Fig. 1.7 is not connected. Its four components G_1, G_2, G_3, and G_4 have vertex sets $\{v_1, v_2, v_3\}$, $\{v_4, v_5\}$, $\{v_6, v_7, v_8\}$, and $\{v_9\}$, respectively.

Note that an isolated vertex by itself should be treated as a component since, by definition, a vertex is connected to itself. Further, note that if a graph G is connected, it has only one component that is the same as G itself.

We next consider some properties of connected graphs.

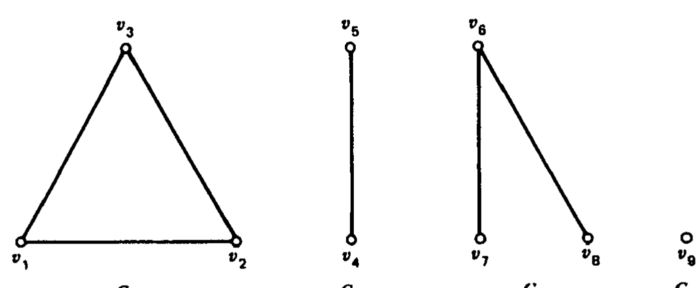

Figure 1.7. Graph G with components G_1, G_2, G_3, and G_4.

[†] A set V is said to be *partitioned* into subsets $V_1, V_2, \ldots, V_p$ if $V_1 \cup V_2 \cup \cdots \cup V_p = V$ and $V_i \cap V_j = \emptyset$ for all i and j, $i \neq j$. $\{V_1, V_2, \ldots, V_p\}$ is then called a *partition* of V.

Theorem 1.3. In a connected graph, any two longest paths have a common vertex.

Proof. Consider any two longest paths P_1 and P_2 in a connected graph G. Let P_1 be denoted by the vertex sequence $v_0, v_1, v_2, \ldots, v_k$ and P_2 by the sequence $v_0', v_1', v_2', \ldots, v_k'$.

Assume that P_1 and P_2 have no common vertex. Since the graph G is connected, then for some $i, 0 \leq i \leq k$ and some $j, 0 \leq j \leq k$ there exists a v_i–v_j' path P_a such that all the vertices of P_a other than v_i and v_j' are different from those of P_1 and P_2. The paths P_1, P_2, and P_a may be as shown in Fig. 1.8. Let

$$t_1 = \text{length of } v_0\text{–}v_i \text{ path } P_{11},$$
$$t_2 = \text{length of } v_i\text{–}v_k \text{ path } P_{12},$$
$$t_1' = \text{length of } v_0'\text{–}v_j' \text{ path } P_{21},$$
$$t_2' = \text{length of } v_j'\text{–}v_k' \text{ path } P_{22},$$
$$t_a = \text{length of path } P_a .$$

The paths P_{11}, P_{12}, P_{21}, and P_{22} are also shown in Fig. 1.8. Note that

$$t_1 + t_2 = t_1' + t_2' = \text{length of a longest path in } G$$

and

$$t_a > 0 .$$

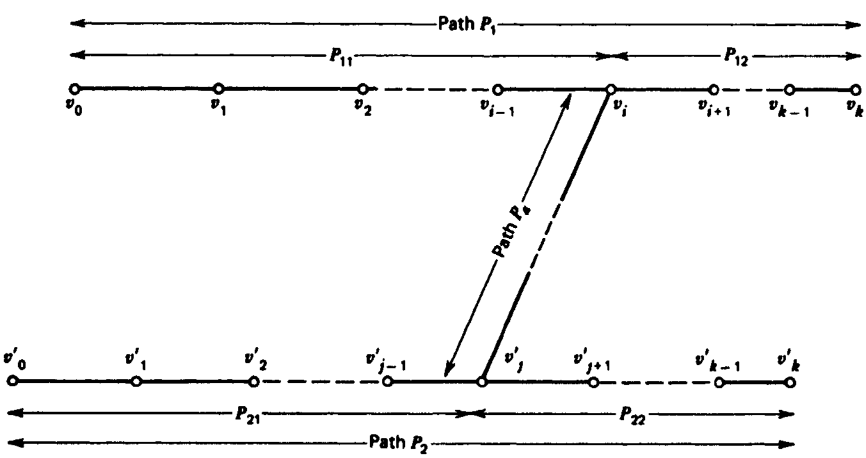

Figure 1.8. Paths P_1, P_2 and P_a.

Without any loss of generality, let

$$t_1 \geq t_2$$

and

$$t_1' \geq t_2'$$

so that

$$t_1 + t_1' \geq t_1 + t_2 = t_1' + t_2' .$$

Now it may be verified that the paths P_{11}, P_a, and P_{21} together constitute a v_0–v_0' path with its length equal to $t_1 + t_1' + t_a > t_1 + t_2$ because $t_a > 0$. This contradicts that $t_1 + t_2$ is the length of a longest path in G. ■

The following theorem is a very useful one; it is used often in the discussions of the next chapter. In this theorem as well as in the rest of the book, we abbreviate $\{x\}$ to x whenever it is clear that we are referring to a set rather than an element.

Theorem 1.4. If a graph $G = (V, E)$ is connected, then the graph $G' = (V, E - e)$ that results after removing a circuit edge e is also connected. ■

We leave the proof of this theorem as an exercise.

1.5 OPERATIONS ON GRAPHS

In this section we introduce a few operations involving graphs. The first three operations are binary operations involving two graphs, and the last four are unary operations, that is, operations defined with respect to a single graph.

Consider two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The *union* of G_1 and G_2, denoted as $G_1 \cup G_2$, is the graph $G_3 = (V_1 \cup V_2, E_1 \cup E_2)$; that is, the vertex set of G_3 is the union of V_1 and V_2, and the edge set of G_3 is the union of E_1 and E_2.

For example, two graphs G_1 and G_2 and their union are shown in Fig. 1.9*a*, *b* and *c*.

The *intersection* of G_1 and G_2, denoted as $G_1 \cap G_2$, is the graph $G_3 = (V_1 \cap V_2, E_1 \cap E_2)$. That is, the vertex set of G_3 consists of only those vertices present in both G_1 and G_2, and the edge set of G_3 consists of only those edges present in both G_1 and G_2.

The intersection of the graphs G_1 and G_2 of Fig. 1.9*a* and 1.9*b* is shown in Fig. 1.9*d*.

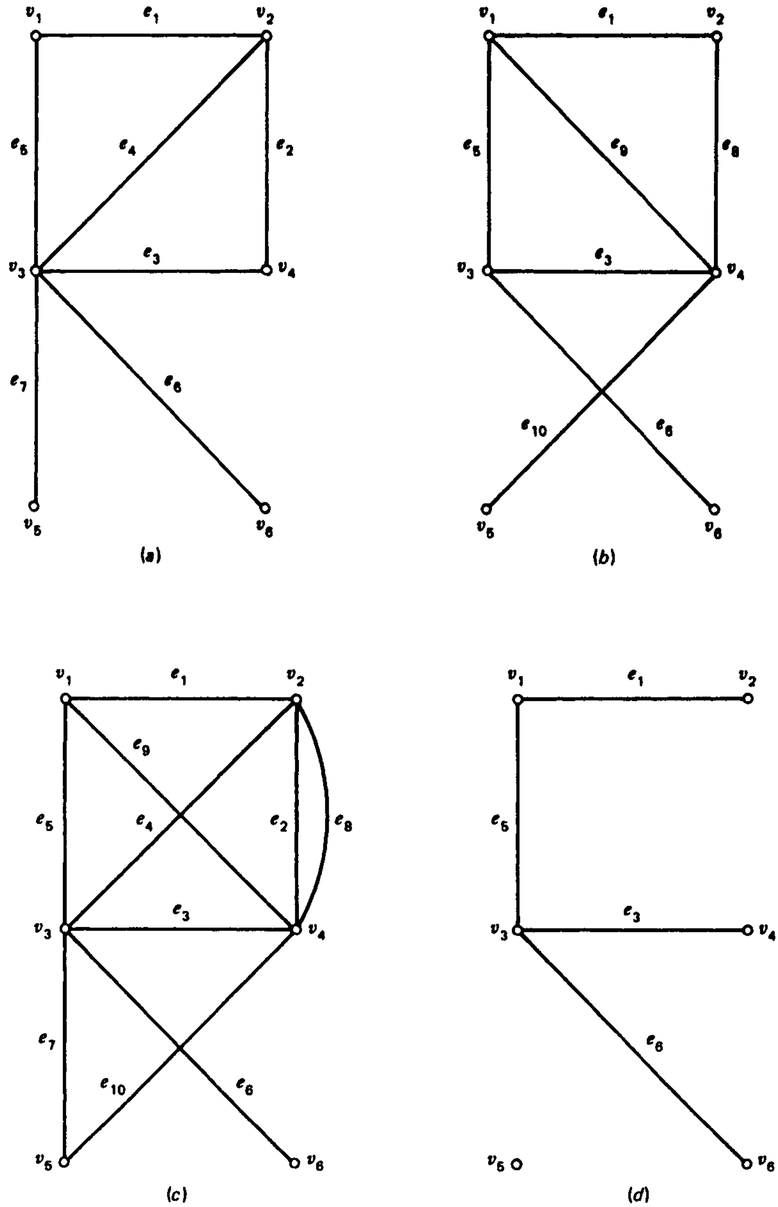

Figure 1.9. Union, intersection, and ring sum operations on graphs. (*a*) Graph G_1. (*b*) Graph G_2. (*c*) $G_1 \cup G_2$. (*d*) $G_1 \cap G_2$. (*e*) $G_1 \oplus G_2$.