A FIRST LOOK AT GRAPH THEORY

John Clark and Derek Allan Holton
Department of Mathematics and Statistics
University of Otago
New Zealand

World Scientific
Singapore • New Jersey • London • Hong Kong

Published by

World Scientific Publishing Co. Pte. Ltd.

P O Box 128, Farrer Road, Singapore 912805

USA office: Suite 1B, 1060 Main Street, River Edge, NJ 07661

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Cataloging-in-Publication Data

Clark, John.
A first look at graph theory / John Clark and Derek Allan Holton.
xv, 330 p.; 21.5 cm.
Includes bibliographical references and index.
ISBN 9810204892 ISBN 9810204906 (pbk)
1. Graph theory. I. Holton, Derek Allan, 1941– II. Title.
III. Title: Graph theory.
QA166.C56 1991
511'.5--dc20 91-14633
CIP

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

First published 1991
First reprint 1996
Second reprint 1998

Printed in Singapore.

To our long-suffering wives,
Austina and Marilyn,
with sincere thanks for their patience,
and to
John and Alan Clark,
who should have seen much more of their father
during their summer holidays.

Preface

With the increasing use of computers in society there has been a dramatic growth in all aspects of computer education. At university level, computer science students quickly learn that their subject has many facets, some of which are better appreciated when the student has an appropriate mathematical background. Unfortunately, much of this mathematical background is not the sort found easily in the mathematical education of earlier generations.

Much of the theory of computer science uses an area of mathematics loosely described as "discrete mathematics", this term chosen to emphasise its contrast with the "continuous mathematics" of the more traditional calculus courses. Discrete mathematics covers many topics and this book takes a first look at one of these — Graph Theory. This topic has a surprising number of applications, not just to computer science but to many other sciences (physical, biological and social), engineering and commerce.

From what we have said so far, the reader may have got the impression that this is a book mainly for computer scientists. Not so. Graph Theory is at last being acknowledged as an important subject in the undergraduate *mathematics* curriculum. Perhaps one should expect a more theoretical treatment here than in the computer science setting. However we feel that a blend of the theory with some of its many varied applications is highly desirable for both disciplines — for those mainly concerned with the applications of graphs, the theory helps to strengthen the ideas and point the way to independent applications; conversely, the applications of graph theory to "real world" situations reinforces the theoretical aspects and illustrates one of the many ways in which mathematics is applied. As a result this text is a mixture of both theory and applications and can be used by both the serious mathematics student, her computer science cousin or any other relation keen to learn about one of the most rapidly growing areas of modern mathematics.

The book began in 1985 as a set of notes for a second year course of 40 one-hour lectures in the Department of Mathematics at the University of Otago. The students attending the course, then and in subsequent years, were mainly a mixture of computer science majors and mathematics majors. Not all topics covered in the book were dealt with in these lectures and, indeed, some theoretical aspects may be quite difficult for the average second year university student. However, Graph Theory is particularly suited to selective study and hopefully our treatment here provides material for the individual teacher to tailor to their course's requirements.

We have also provided a plentiful supply of exercises. These are of varying difficulty. Some deal with the algorithmic aspects of the text, some on the theoretical while others introduce some new but related ideas. We encourage the reader to do as many as possible — mathematics is not a spectator sport and is best appreciated with active participation!

We wish to express our sincere thanks to several people involved in the preparation of this book. Jane Hill helped enormously with the typing and typesetting, John Marshall proofread an early version of the text and also made numerous suggestions on style, Gordon Yau prepared some of the diagrams, Maree Watson typed portions of the manuscript, Graeme McKinstry provided very useful LaTeX expertise while Mark Borrie gave frequently-needed computer assistance.

However our main thanks go to our families for their patience during the last few months of the book's preparation.

A Note to the Reader

One of the beauties of Graph Theory is that it depends very little on other branches of mathematics. However in our text we do occasionally rely on the reader having what is often called "mathematical maturity". This means an ability on behalf of the reader to understand and appreciate a mathematical argument or proof. This ability is something that usually is not acquired overnight but is the outcome of an ongoing exposure to mathematics and its accompanying logic. Hopefully, the reader's mathematical maturity will grow as he progresses through the text. If so, then we will have achieved one of the goals of the book.

On a more concrete level, we will assume that the reader knows about the principle of mathematical induction. This is dealt with in many undergraduate textbooks. In particular, the text by Mott, Kandel and Baker, mentioned in the section on Further Reading, has a nice treatment of this.

We also assume that the reader is familiar with the notion of a set. We use $\emptyset$ to denote the empty set and, for two sets A and B, we denote the set difference, consisting of all elements which belong to A but not B, by $A - B$.

Each of the ten chapters of the book is split up into numbered sections, with, for example, Section 2.5 denoting the fifth section of Chapter 2. At the end of most sections there is a collection of numbered exercises. For example, Exercise 2.5.3 refers to the third exercise accompanying Section 2.5. Within each chapter, results such as theorems and corollaries are also numbered consecutively. For example, Theorem 4.3 is the third result of Chapter 4 and it is followed by Corollary 4.4.

The end of a proof of a theorem or corollary is shown by the symbol $\square$.

We have referred to several books and articles throughout the text and details of these are given in the bibliography at the end of the book. Such a reference is given by a number in square brackets with, for example, [8] referring to the eighth item in the bibliography.

Further Reading

We have been much influenced in both our choice and treatment of topics by other Graph Theory texts. Of these we first mention Bondy and Murty's "Graph Theory with Applications" [7]. Sadly, this excellent text is currently out of print, but hopefully your library has a copy. We also learned much from Wilson's enjoyable "Introduction to Graph Theory" [65]. (We have just become aware of, but not yet seen, a new text by Wilson [66], coauthored with Watkins, and on the basis of [65], we feel we can highly recommend it!) On a more advanced level there is also the recent text by Gould [28], "Graph Theory", which provides an up-to-date treatment of the theory with an algorithmic flavour. For an advanced authorative account of the theoretical side of the subject we refer the reader to Behzad, Chartrand and Lesniak-Forster's "Graphs and Digraphs" [4], while Chartrand's "Graphs as Mathematical Models" [13] and Ore's "Graphs and Their Uses" [49] provide more elementary treatments of both the theory and its applications. (Ore's book has just been re-issued in a new edition updated by Wilson.)

For a more algorithmic flavour, we recommend Smith's "Network Optimisation Practise" [57] and Albertson and Hutchinson's "Discrete Mathematics with Algorithms" [1]. We also refer the reader to the more advanced Gibbon's "Algorithmic Graph Theory" [26] and Sysło, Deo and Kowalik's "Discrete Optimization Algorithms with Pascal Programs" [58].

There are several recent texts on the more general area of discrete mathematics. Of those that have substantial treatments of graphs, we mention Mott, Kandel and Baker's "Discrete Mathematics for Computer Scientists" [45] and Polimeni and Straight's "Foundations of Discrete Mathematics" [50]. Finally, as excellent sources of numerous applications of graphs, there are the two books, both titled "Applied Combinatorics", by Roberts [55] and by Tucker [61].

Contents

A FIRST LOOK AT GRAPH THEORY

Chapter 1

An Introduction to Graphs

1.1 The Definition of a Graph

In a hockey league there are eight teams, which we denote by S, T, U, V, W, X, Y and Z. After a few weeks of the season the following games have been played:

S has played X and Z,	T has played W, X and Z
U has played Y and Z,	V has played W and Y,
W has played T, V and Y,	X has played S and T,
Y has played U, V and W, and	Z has played S, T and U.

We may illustrate this situation by either of the two diagrams of Figure 1.1, where the teams are represented by (large) dots and two such dots are joined by a line whenever the corresponding teams have played each other.

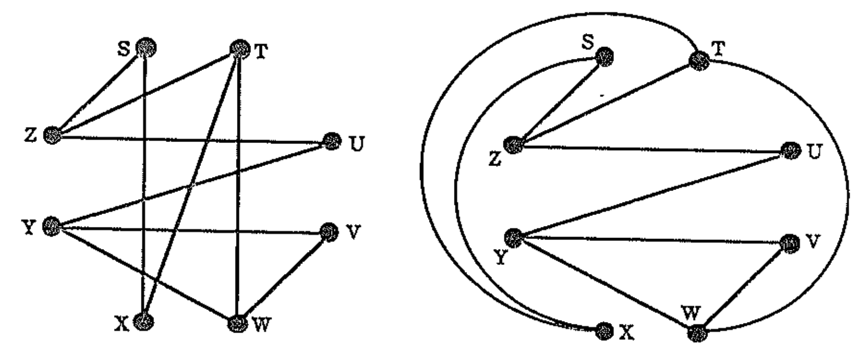


Figure 1.1: Games played in the hockey league.

In the diagram on the left the dots have been joined using straight lines, while in the other diagram three of the lines used are not straight. But since we are simply interested in which games have been played, the manner in which a pair of dots is joined is of no importance and so it does not matter whether the lines are straight or not.

The diagrams may be used to describe other situations. For example, the eight dots may represent eight people, with the lines joining a pair of dots if the two people know each other. Or, the dots could be communication centres with the lines denoting communication links. Indeed, as we will see later, many real-world situations can conveniently be described by means of such drawings, which we call **graphs**.

A graph, then, can be thought of as a drawing or diagram consisting of a collection of vertices (dots or points) together with edges (lines) joining certain pairs of these vertices, as in the diagrams of Figure 1.1. Notice that in the left-hand diagram there are intersecting edges (lines), for example, SX and ZU; the points of intersection of these edges are not however vertices (points) of our graph. To avoid any confusion that could arise in this way, the vertices in any drawing of a graph will normally be drawn as large dots. However, as we will now see from its formal definition, from a mathematical point of view a graph does not need to be drawn.

> A **graph** $G = (V(G), E(G))$ consists of two finite sets:
> $V(G)$, the **vertex set** of the graph, often denoted by just V, which is a nonempty set of elements called **vertices**, and
> $E(G)$, the **edge set** of the graph, often denoted by just E, which is a possibly empty set of elements called **edges**,
> such that each edge e in E is assigned an unordered pair of vertices (u, v), called the **end vertices** of e.

Thus for both graphs of Figure 1.1, the vertex set is

$$V = \{S, T, U, V, W, X, Y, Z\},$$

the edge set E has 10 edges and these edges are assigned the unordered pairs of vertices

$$(S, X), (S, Z), (T, W), (T, X), (T, Z), (U, Y), (U, Z), (V, W), (V, Y), (W, Y).$$

> Vertices are also sometimes called **points**, **nodes**, or just **dots**.
> If e is an edge with end vertices u and v then e is said to **join** u and v.
> Note that the definition of a graph allows the possibility of the edge e having identical end vertices, i.e., it is possible to have a vertex u joined to itself by an edge — such an edge is called a **loop**.

We now give two examples to illustrate the above definitions.

Example 1. Let $G = (V, E)$ where
$V = \{a, b, c, d, e\}$, $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$,
and the ends of the edges are given by:

$$e_1 \leftrightarrow (a, b),\quad e_2 \leftrightarrow (b, c),\quad e_3 \leftrightarrow (c, c),\quad e_4 \leftrightarrow (c, d),$$
$$e_5 \leftrightarrow (b, d),\quad e_6 \leftrightarrow (d, e),\quad e_7 \leftrightarrow (b, e),\quad e_8 \leftrightarrow (b, e).$$

We can then represent G diagrammatically as in Figure 1.2.

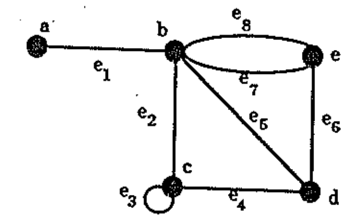


Figure 1.2: A graph G with five vertices and eight edges.

Example 2. Let H be the graph (V, E) where

$$V = \{1, 2, 3, 4, 5, 6\},\quad E = \{a, b, c, d, e, f, g, h\},$$

and the ends of the edges are given by:

$$a \leftrightarrow (1, 2),\quad b \leftrightarrow (1, 1),\quad c \leftrightarrow (2, 3),\quad d \leftrightarrow (3, 4),$$
$$e \leftrightarrow (2, 4),\quad f \leftrightarrow (3, 4),\quad g \leftrightarrow (1, 4),\quad h \leftrightarrow (4, 5).$$

We can represent H diagrammatically as in Figure 1.3.

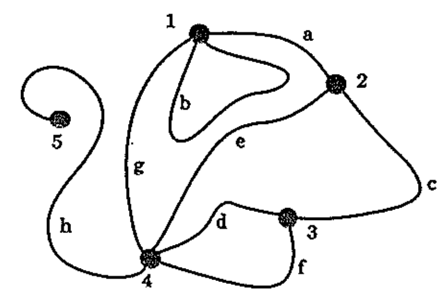


Figure 1.3: A graph H with five vertices and eight edges.

1.2 Graphs as Models

We now give some problems in which graphs provide a natural mathematical model. At this stage we only describe the problems and delay detailed discussion of their solution to later chapters.

Problem 1. Suppose that the graph of Figure 1.4 represents a network of telephone lines and poles. We are interested in the network's vulnerability to accidental disruption. We want to identify those lines and poles that must stay in service to avoid disconnecting the network.

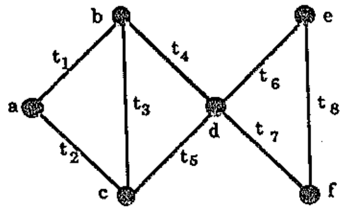


Figure 1.4: A network of telephone lines.

There is no single line whose disruption (removal) will disconnect the graph (network), but the graph will become disconnected if we remove the two lines represented by the edges t_4 and t_5, for example. When it comes to poles, the network is more vulnerable since there is a single vertex, vertex d, whose removal disconnects the graph. This illustrates the notions of edge connectivity and vertex connectivity of a graph, discussed in Sections 8.3 and 2.6.

We may also want to find a smallest possible set of edges needed to connect the six vertices. There are several examples of such minimal sets. One is

$$\{t_1, t_3, t_5, t_6, t_7\}.$$

In fact, as we will see in Chapter 2 when we look at trees and connected graphs, any such minimal set will have precisely 5 edges.

Problem 2. Suppose that we have five people A, B, C, D, E and five jobs a, b, c, d, e and some of these people are qualified for certain jobs. Is there a feasible way of allocating one job to each person, or to show that no such matching up of jobs and people is possible? We can represent this situation by a graph having a vertex for each person and a vertex for each job, and edges joining people up to jobs for which they are qualified. Does there exist a feasible matching of people to jobs for the graph of Figure 1.5?

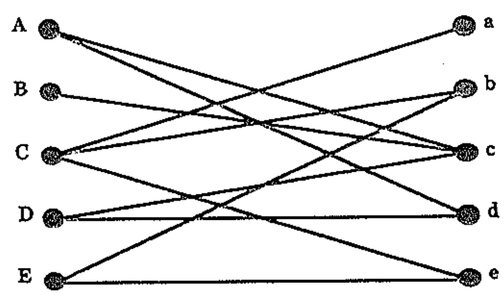


Figure 1.5: A job applications graph.

The answer is no. The reason can be found by considering people A, B, and D. These three people as a set are collectively qualified for only two jobs, c and d, hence there

is no feasible matching possible for these three people, much less all five people. We will look at an algorithm for finding a feasible matching, if any exists, later in Section 4.2.

Problem 3. Suppose a salesman's territory includes several cities with highways connecting certain pairs of these cities. His job requires him to visit each city personally. Is it possible for him to schedule a round trip by car enabling him to visit each specified city exactly once? We can represent the transportation system involved by a graph G whose vertices correspond to the cities and such that two vertices are joined by an edge if and only if a highway connects the corresponding cities (and does not pass through any other specified city). The graphs G_1 and G_2 of Figure 1.6 denote two such salesmen's territories. The desired schedule is possible for G_1 but not for G_2. (In G_1, starting at vertex u_1, we can visit each vertex and arrive back at u_1 by taking the edges e_1, e_2, e_3, e_4, and e_6 in turn.)

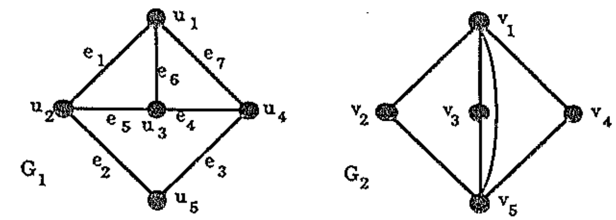


Figure 1.6: Two travelling salesmen's territories.

Problem 4. Suppose we have three houses each of which have to be supplied with electricity, gas and water. Is it possible to connect each utility with each of the three houses without the lines or mains crossing?

We can represent the connection of the three houses to the three utilities by the graph of Figure 1.7. Here we have a graph with six vertices, three of which represent the houses (denoted by H_1, H_2, H_3), the other three represent the utilities (denoted by E, G, H), and an edge joins two vertices if and only if one vertex denotes a house and the other vertex a utility.

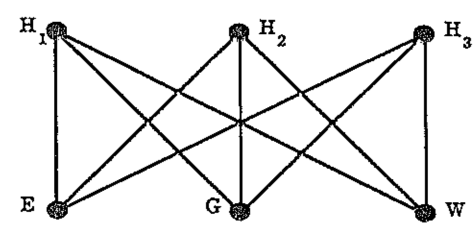


Figure 1.7: The three utilities graph.

The problem then is as to whether or not we can draw this graph in such a way that no two edges intersect. The answer is no — we will see why when we look later at *planar* graphs in Chapter 5.

Problem 5. Six radio broadcasting companies $C_1, \dots, C_6$ have applied to the Minister of Broadcasting for frequency channels. If two companies' transmitters are within 200 kilometres of each other they can not be assigned the same frequency since there will be too much interference. The Minister wishes to assign as small a number of different frequencies as possible, taking this interference into account.

To illustrate the problem, let G be the graph of Figure 1.8 with vertex set $C_1, \dots, C_6$ with two vertices C_i and C_j joined by an edge if and only if the two companies C_i and C_j have their transmitters less than 200 kilometres apart. As a particular example, consider the following graph G where, for example, C_2's transmitter is within 200 kilometres of those of C_1, C_3, C_4 and C_5.

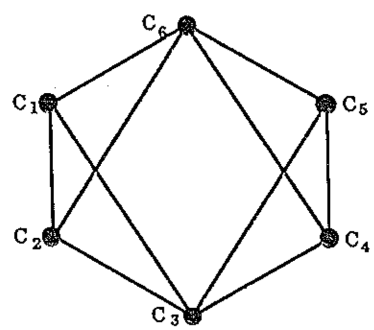


Figure 1.8: Radio transmitters and their interference graph.

Now suppose we assign different colours to the vertices of G in such a way that no two vertices of the same colour are joined by an edge. Then, thinking of the colours as representing the frequency channels, the Minister wants to find the minimum number of colours with which the vertices can be coloured in this way. For our example, the answer is 3 — we can colour C_1 and C_4 red, C_2 and C_5 blue, and C_3 and C_6 yellow.

Problem 6. A company has branches in each of six cities $C_1, C_2, \dots, C_6$. The airfare for a direct flight from C_i to C_j is given by the (i, j)th entry of the following matrix (where ∞ indicates that there is no direct flight). For example the fare from C_1 to C_4 is \$40, from C_2 to C_3 is \$15.

$$\begin{bmatrix} 0 & 50 & \infty & 40 & 25 & 10 \\ 50 & 0 & 15 & 20 & \infty & 25 \\ \infty & 15 & 0 & 10 & 20 & \infty \\ 40 & 20 & 10 & 0 & 10 & 25 \\ 25 & \infty & 20 & 10 & 0 & 55 \\ 10 & 25 & \infty & 25 & 55 & 0 \end{bmatrix}$$

The company is interested in computing a table of cheapest fares between pairs of cities. (Even if there is a direct flight between two cities this may not be the cheapest route.) We can first represent the situation by a *weighted* graph, i.e., a graph with "weights" attached to the edges according to the airfares, as in Figure 1.9.

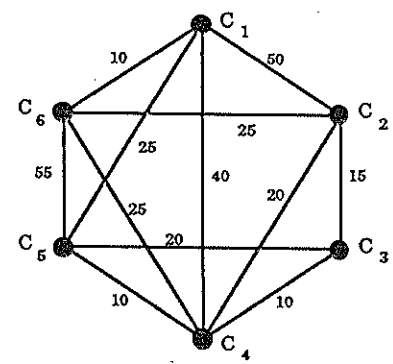


Figure 1.9: The weighted graph of airfares for direct flights between six cities.

The problem can then be solved using *Dijkstra's algorithm* — see Section 2.5. This type of problem is called a *shortest path problem.*

1.3 More Definitions

> Let G be a graph. If two (or more) edges of G have the same end vertices then these edges are called **parallel.**

For example, the edges e_7 and e_8 of the graph of Figure 1.2 are parallel.

> A vertex of G which is not the end of any edge is called **isolated.**
> Two vertices which are joined by an edge are said to be **adjacent** or **neighbours.**
> The set of all neighbours of a fixed vertex v of G is called the **neighbourhood set** of v and is denoted by $N(v)$.

Thus, in the graph of Figure 1.9, C_1 and C_4 are adjacent but C_2 and C_5 are not. The neighbourhood set $N(C_3)$ of C_3 is $\{C_2, C_4, C_5\}$.

> A graph is called **simple** if it has no loops and no parallel edges.

Much of the graphs that we consider will be simple. (The reader should be warned that other authors may differ from us in their use of the term "graph". For example, in some texts a graph which is not simple is called a **multigraph.**)

It is often the case that two graphs have the same structure, differing only in the way their vertices and edges are labelled or only in the way they are represented geometrically. For many purposes, we can regard the two graphs as *essentially* the same. This essential likeness has a special name and we now define this formally.

> A graph $G_1 = (V_1, E_1)$ is said to be **isomorphic** to the graph $G_2 = (V_2, E_2)$ if there is a one-to-one correspondence between the vertex sets V_1 and V_2 and a one-to-one correspondence between the edge sets E_1 and E_2 in such a way that if e_1 is an edge with end vertices u_1 and v_1 in G_1 then the corresponding edge e_2 in G_2 has its end points the vertices u_2 and v_2 in G_2 which correspond to u_1 and v_1 respectively.
> Such a pair of correspondences is called a **graph isomorphism.**

In other words, the graphs G_1 and G_2 are isomorphic if the vertices of G_1 can be paired off with the vertices of G_2 and the edges of G_1 can be paired off with the edges of G_2 in such a way that the ends of paired off edges are paired off. Thus G_1 is really just the same graph as G_2, apart from a possible change in how the vertices and edges are named (or a possible redrawing of the graphs).

Figure 1.10 shows five fairly obvious pairs of isomorphic graphs.

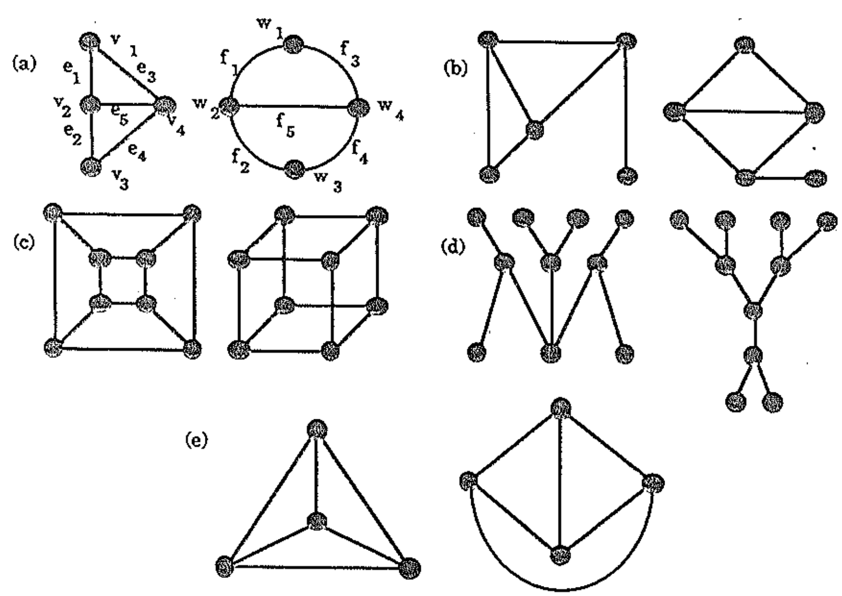


Figure 1.10: Isomorphic pairs of graphs.

Figure 1.11 gives some examples to illustrate that often it can be quite difficult to determine if two graphs are isomorphic.

In Figure 1.11 (a) an isomorphism is given by the following one-to-one correspondence of vertices:

$$u_1 \leftrightarrow v_1,\ u_2 \leftrightarrow v_3,\ u_3 \leftrightarrow v_5,\ u_4 \leftrightarrow v_2,\ u_5 \leftrightarrow v_4, u_6 \leftrightarrow v_6$$

(Note how u_1, u_2, u_3 are only joined to u_4, u_5, u_6 and similarly v_1, v_3, v_5 are only joined to v_2, v_4, v_6.) Isomorphisms for Figure 1.11 (b) and (c) are shown by the labelling of the vertices.

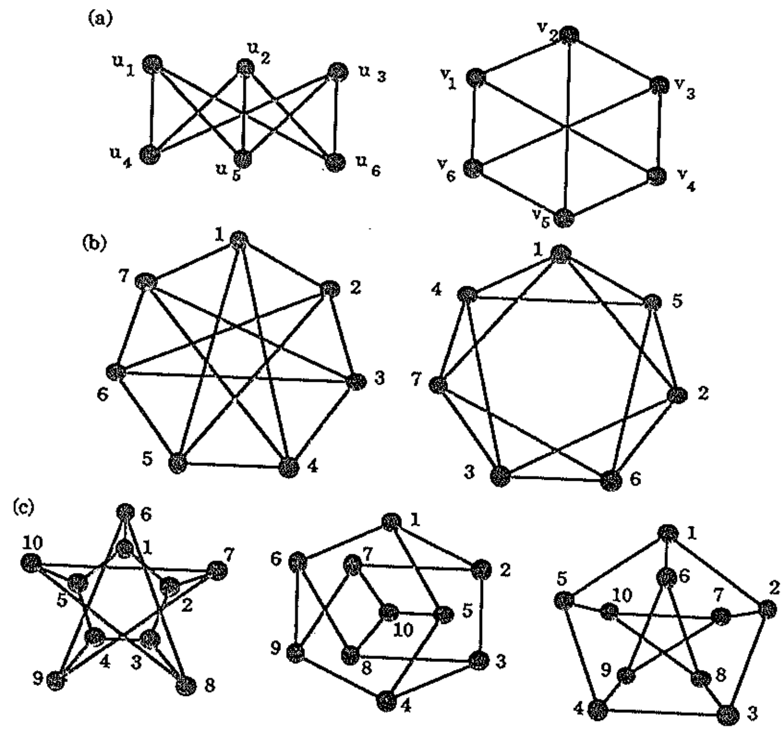


Figure 1.11: Some more groups of isomorphic graphs.

The problem of determining when two graphs are isomorphic gets harder as the number of vertices and edges of the graphs get larger. For example, while there are only 4 non-isomorphic simple graphs on three vertices and 11 on 4 vertices there are 1044 non-isomorphic simple graphs on just seven vertices.

Clearly if two graphs G_1 and G_2 are isomorphic then they must have

(i) the same number of vertices and (ii) the same number of edges.

However, as we will now see, conditions (i) and (ii) are not sufficient. The graph G of Figure 1.12 has the same number of vertices and edges as the graphs in Figure 1.11 (a) but G is not isomorphic to these graphs.

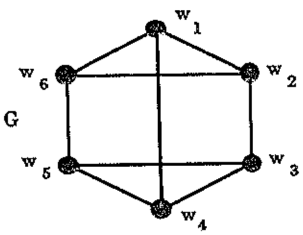


Figure 1.12

A **complete** graph is a simple graph in which each pair of distinct vertices is joined by an edge.

Thus, a graph with n vertices is complete if it has as many edges as possible provided there are no loops and no parallel edges.
If the complete graph has vertices $v_1, \dots, v_n$ then the edge set can be given by

$$E = \{(v_i, v_j) : v_i \neq v_j;\ i, j = 1, \dots, n.\}$$

It follows that the graph has $\frac{1}{2}n(n-1)$ edges (since there are $n-1$ edges incident with each of the n vertices v_i, so a total of $n \times (n-1)$, but divide by 2 since $(v_j, v_j) = (v_j, v_i)$).
Given any two complete graphs with the same number of vertices, n, then they are isomorphic. In fact any pairing off of the vertices gives a corresponding pairing off of the edges and hence an isomorphism. For this reason we speak of *the* complete graph on n vertices. It is denoted by K_n.
Figure 1.13 shows $K_1, \dots, K_6$.

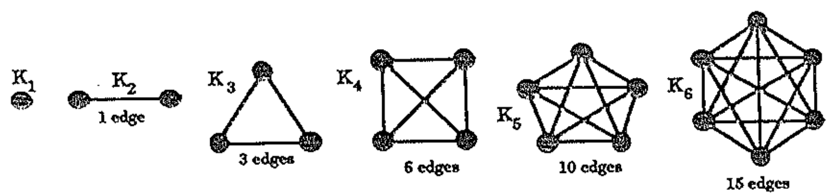


Figure 1.13: The complete graphs on at most six vertices.

An **empty** (or **trivial**) graph is a graph with no edges.

Let G be a graph. If the vertex set V of G can be partitioned into two nonempty subsets X and Y (i.e., $X \cup Y = V$ and $X \cap Y = \emptyset$) in such a way that each edge of G has one end in X and one end in Y then G is called **bipartite**. The partition $V = X \cup Y$ is called a **bipartition** of G.
A **complete bipartite graph** is a simple bipartite graph G, with bipartition $V = X \cup Y$, in which every *vertex* in X is joined to *every* vertex of Y. If X has m vertices and Y has n vertices, such a graph is denoted by $K_{m,n}$.

Any complete bipartite graph with a bipartition into two sets of m and n vertices is isomorphic to $K_{m,n}$ — in fact any pairing off of the two sets of m vertices together with any pairing off of the two sets of n vertices will give an isomorphism. In particular, $K_{m,n}$ is (of course) isomorphic to $K_{n,m}$.
Since each of the m vertices in the partition set X of $K_{m,n}$ is adjacent to each of the n vertices in the partition set Y, $K_{m,n}$ has $m \times n$ edges.
Note that there is now an unfortunate ambiguity in the use of the word *complete*, since a complete bipartite graph will not in general be complete. Indeed, as the reader should easily verify, the only complete bipartite graph which is complete is $K_{1,1}$.
Figure 1.14 shows two bipartite graphs. They are not complete bipartite. However, the graphs of Figure 1.15 *are* complete bipartite.

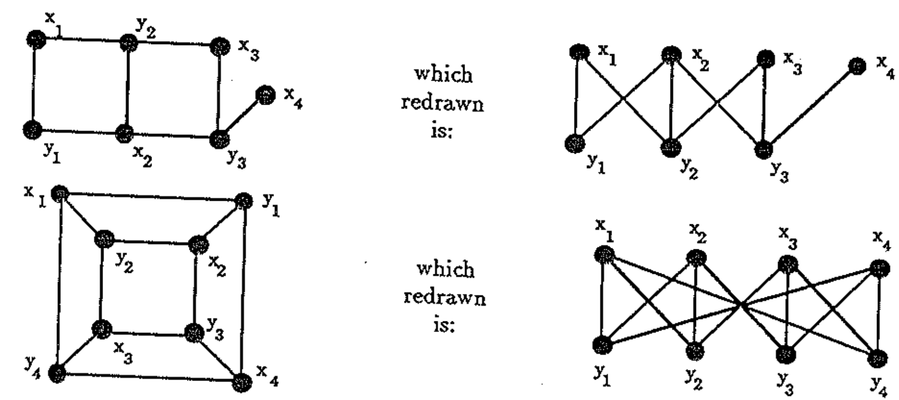


Figure 1.14: Some bipartite graphs

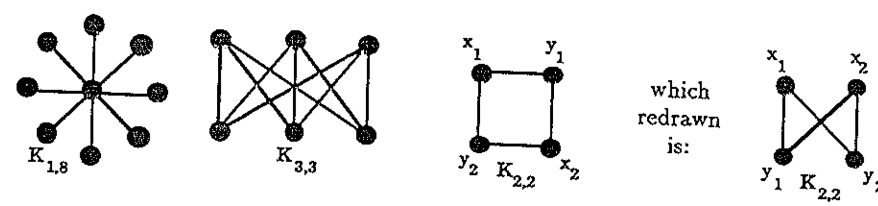


Figure 1.15: Some complete bipartite graphs

Exercises for Section 1.3

1.3.1 Make a list of drawings of all the graphs with n vertices and e edges for all n, e with $n + e \leq 6$. The list should not include any isomorphic pairs of graphs. Determine which of the graphs on your list are simple. (You should get a total of 65 graphs of which 14 are simple.)

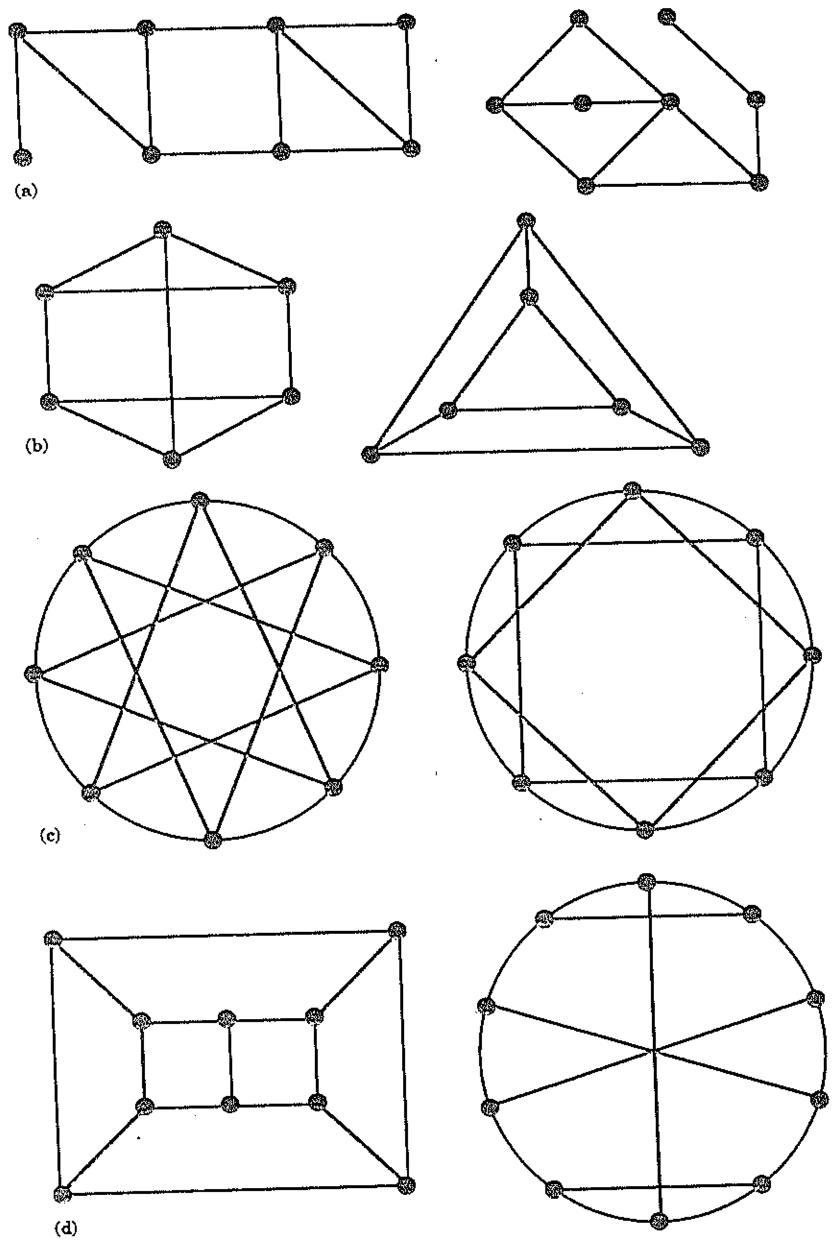


Figure 1.16: Which of these are isomorphic pairs?

1.3.2 Determine which of the pairs of graphs in Figure 1.16 are isomorphic pairs. (Give an argument justifying your answer.)

1.3.3 In each collection of three graphs shown in Figure 1.17 there is exactly one isomorphic pair. Find each pair and justify that your answer is correct.

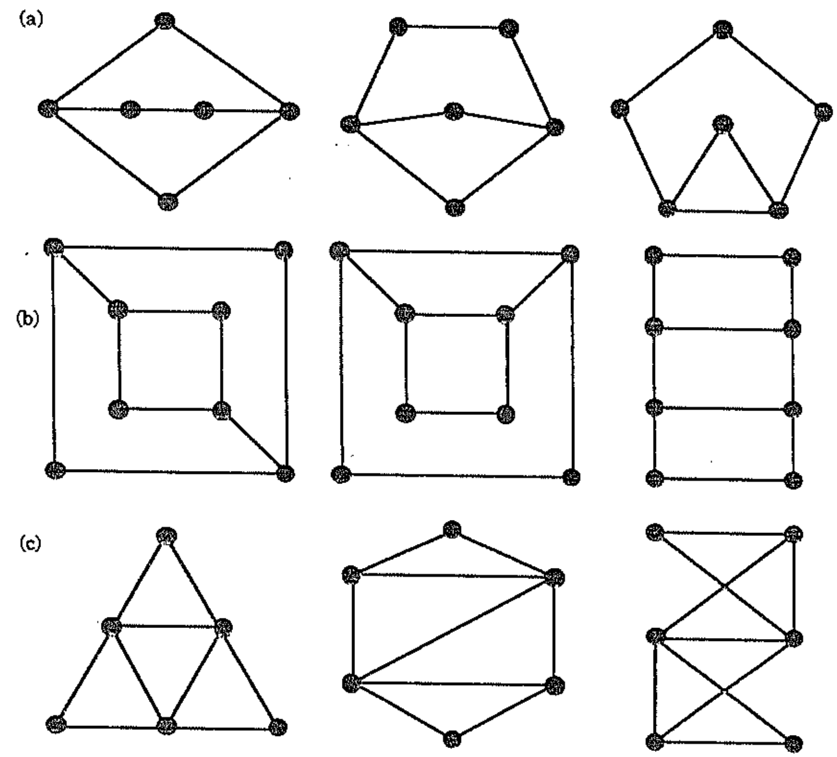


Figure 1.17: Find the odd one out!

1.3.4 Find all nonisomorphic complete bipartite graphs with at most 7 vertices.

1.4 Vertex Degrees

An edge e of a graph G is said to be **incident** with the vertex v if v is an end vertex of e. In this case we also say that v is **incident** with e. Two edges e and f which are incident with a common vertex v are said to be **adjacent**.

Note that an edge is incident with either 1 or 2 vertices (1 if it is a loop), whereas a vertex may be incident with any finite number, including 0, of edges.

> Let v be a vertex of the graph G. The **degree** $d(v)$ (or $d_G(v)$ if we want to emphasize G) of v is the number of edges of G incident with v, counting each loop twice, i.e., it is the number of times v is an end vertex of an edge.

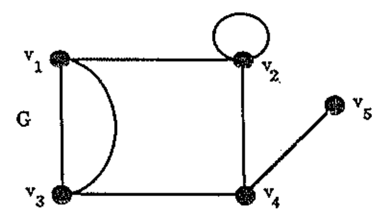


Figure 1.18

In the graph of Figure 1.18 we have $d(v_1) = 3$, $d(v_2) = 4$, $d(v_3) = 3$, $d(v_4) = 3$, $d(v_5) = 1$. Note that in this example

$$d(v_1) + d(v_2) + d(v_3) + d(v_4) + d(v_5) = 14 = 2 \times (\text{number of edges in } G).$$

This is no coincidence because of:

Theorem 1.1 (The First Theorem of Graph Theory) *For any graph G with e edges and n vertices, $v_1, \ldots, v_n$,*

$$\sum_{i=1}^{n} d(v_i) = 2e.$$

Proof Each edge, since it has two end vertices, contributes precisely 2 to the sum of the degrees, i.e., when the degrees of the vertices are summed each edge is counted twice. (Note that even a loop contributes 2 although the 2 ends are identical.) □

> A vertex of a graph is called **odd** or **even** depending on whether its degree is odd or even.

In the graph G of Figure 1.18 there is an even number of odd vertices (these vertices being v_1, v_3, v_4, v_5). Again this is a consequence of a general result:

Corollary 1.2 *In any graph G there is an even number of odd vertices.*

Proof Let W be the set of odd vertices of G and let U be the set of even vertices of G. Then, for each $u \in U$, $d(u)$ is even and so $\sum_{u \in U} d(u)$, being a sum of even numbers, is even. However

$$\sum_{u \in U} d(u) + \sum_{w \in W} d(w) = \sum_{v \in V} d(v) = 2e,$$

by Theorem 1.1, where V is the vertex set of G and e is the number of its edges. Thus

$$\sum_{w \in W} d(w) = 2e - \sum_{u \in U} d(u)$$

is even (being the difference of two even numbers). As all the terms in $\sum_{w \in W} d(w)$ are odd and their sum is even there must be an even number of them (because the sum of an odd number of odd numbers is odd). □

Note that it is not true in general that a graph must have an *odd* number of *even* vertices, e.g., the graph of Figure 1.19 has four even vertices (and we leave the reader to construct a graph which has exactly n even vertices for *any* given n).

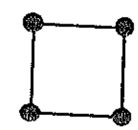


Figure 1.19

> If for some positive integer k, $d(v) = k$ for *every* vertex v of the graph G, then G is called **k-regular**.
> A **regular** graph is one that is k-regular for some k.

The graph drawn above is 2-regular. The complete graph K_n is $(n-1)$-regular. The complete bipartite graph $K_{n,n}$ on $2n$ vertices is n-regular. The graph G_2 of Figure 1.12 is 3-regular, as is the graph of Figure 1.11 (c).

Exercises for Section 1.4

1.4.1 List the degrees of each of the vertices of the graph G of Figure 1.20. How many even vertices does G have? How may odd vertices does G have?

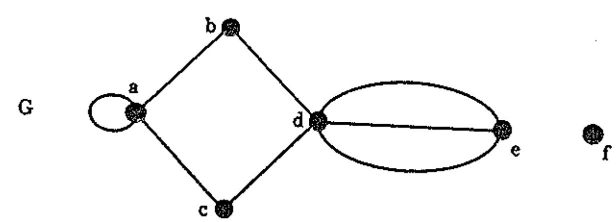


Figure 1.20: Find the degree of each vertex of this graph.

1.4.2 Let G be a graph in which there is no pair of adjacent edges. What can you say about the degrees of the vertices in G?

1.4.3 Let G be a graph with n vertices and e edges and let m be the smallest positive integer such that $m \geq 2e/n$. Prove that G has a vertex of degree at least m.

1.4.4 Prove that it is impossible to have a group of nine people at a party such that each one knows exactly five of the others in the group.

1.4.5 Let G be a graph with n vertices, t of which have degree k and the others have degree $k+1$. Prove that $t = (k+1)n - 2e$, where e is the number of edges in G.

1.4.6 Let G be a k-regular graph, where k is an odd number. Prove that the number of edges in G is a multiple of k.

1.4.7 Let k be some positive integer, greater than 1. The **k-cube**, Q_k, is the graph whose vertices are the ordered k-tuples of 0's and 1's, two vertices being joined by an edge if and only if they differ in exactly one position. Thus, for example, for $k = 3$ the vertices are $(0,0,0)$, $(0,0,1)$, $(0,1,0)$, $(1,0,0)$, $(1,0,1)$, $(1,1,0)$, $(0,1,1)$, $(1,1,1)$ and, for example, $(0,0,0)$ is joined to $(0,0,1)$, $(0,1,0)$ and $(1,0,0)$ but not to any other vertex. The 3-cube Q_3 is shown in Figure 1.21.

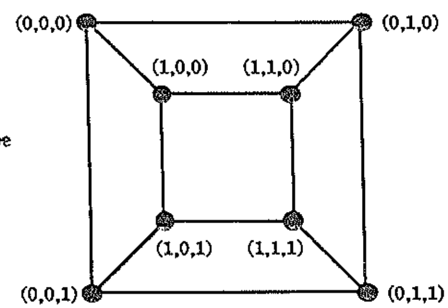


Figure 1.21: The 3-cube Q_3.

(a) Show that the k-cube has 2^k vertices, $k2^{k-1}$ edges and is bipartite.

(b) Using the bipartite property, draw a picture of the 4-cube Q_4.

1.4.8 Let G be a graph with n vertices and exactly $n-1$ edges. Prove that G has either a vertex of degree 1 or an isolated vertex.

1.4.9 What is the smallest integer n such that the complete graph K_n has at least 500 edges?

1.4.10 Prove that there is no simple graph with six vertices, one of which has degree 2, two have degree 3, three have degree 4 and the remaining vertex has degree 5.

1.4.11 Prove that there is no simple graph on four vertices, three of which have degree 3 and the remaining vertex has degree 1.

1.4.12 Let G be a simple regular graph with n vertices and 24 edges. Find all possible values of n and give examples of G in each case.

1.5 Subgraphs

It is often the case that a graph under study is contained within some larger graph also being investigated.

Let H be a graph with vertex set $V(H)$ and edge set $E(H)$ and, similarly, let G be a graph with vertex set $V(G)$ and edge set $E(G)$. Then we say that H is a **subgraph** of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. In such a case, we also say that G is a **supergraph** of H.

For example, in Figure 1.22, G_1 is a subgraph of both G_2 and G_3 but G_3 is not a subgraph of G_2.

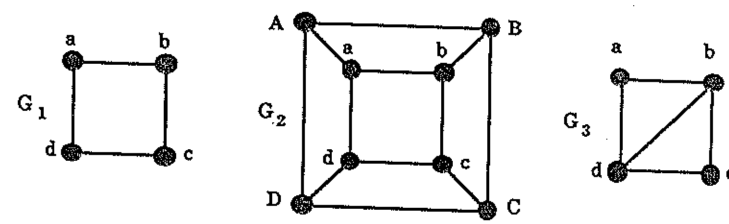


Figure 1.22: $G_1 \subseteq G_2$, $G_1 \subseteq G_3$ but $G_3 \not\subseteq G_2$.

Any graph *isomorphic* to a subgraph of G is also referred to as a **subgraph** of G.

If H is a subgraph of G then we write $H \subseteq G$. When $H \subseteq G$ but $H \neq G$, i.e., $V(H) \neq V(G)$ or $E(H) \neq E(G)$, then H is called a **proper subgraph** of G.

A **spanning subgraph** (or **spanning supergraph**) of G is a subgraph (or supergraph) H with $V(H) = V(G)$, i.e., H and G have exactly the same vertex set.

It follows easily from the definitions that any *simple* graph on n vertices is a subgraph of the complete graph K_n.

In Figure 1.22, G_1 is a proper spanning subgraph of G_3.

The simplest types of subgraph of a graph G are those obtained by the **deletion** of a vertex or an edge and we now define these.

If $G = (V, E)$ and V has at least two elements (i.e., G has at least two vertices), then for any vertex v of G, $G - v$ denotes the subgraph of G with vertex set $V - \{v\}$ and whose edges are all those of G which are not incident with v, i.e., $G - v$ is obtained from G by removing v and all the edges of G which have v as an end. $G - v$ is referred to as a **vertex deleted subgraph**.

If $G = (V, E)$ and e is an edge of G then $G - e$ denotes the subgraph of G having V as its vertex set and $E - \{e\}$ as its edge set, i.e., $G - e$ is obtained from G by removing the edge e, (but not its endpoint(s)). $G - e$ is referred to as an **edge deleted subgraph**.

We extend the above definition to cater for the deletion of several vertices or edges.

If $G = (V, E)$ and U is a proper subset of V then $G - U$ denotes the subgraph of G with vertex set $V - U$ and whose edges are all those of G which are not incident with any vertex in U.

If F is a subset of the edge set E then $G - F$ denotes the subgraph of G with vertex set V and edge set $E - F$, i.e., obtained by deleting all the edges in F, but not their endpoints.

$G - U$ and $G - F$ are also referred to as **vertex deleted** and **edge deleted subgraphs** (respectively).

Figure 1.24 gives some examples.

By deleting from a graph G all loops and in each collection of parallel edges all edges but one in the collection we obtain a simple spanning subgraph of G, called the **underlying simple graph** of G.

Figure 1.23 shows a graph and its underlying simple graph obtained by deleting the loop e_6, two of the parallel edges e_2, e_3, e_4, and one of the pair of parallel edges e_7, e_8.

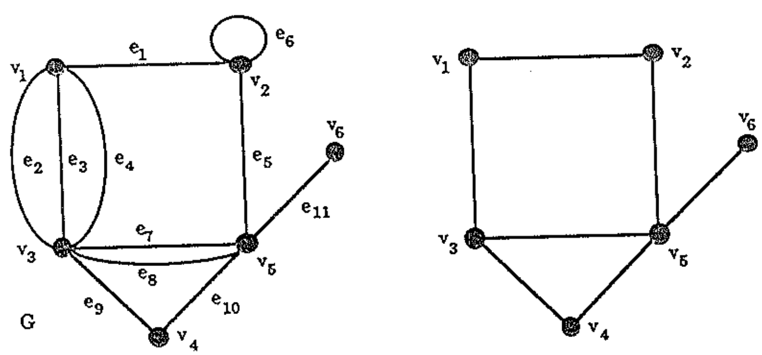


Figure 1.23: A graph and its underlying simple graph.

Some of the more important subgraphs we shall encounter are the *induced subgraphs* and we now define these.

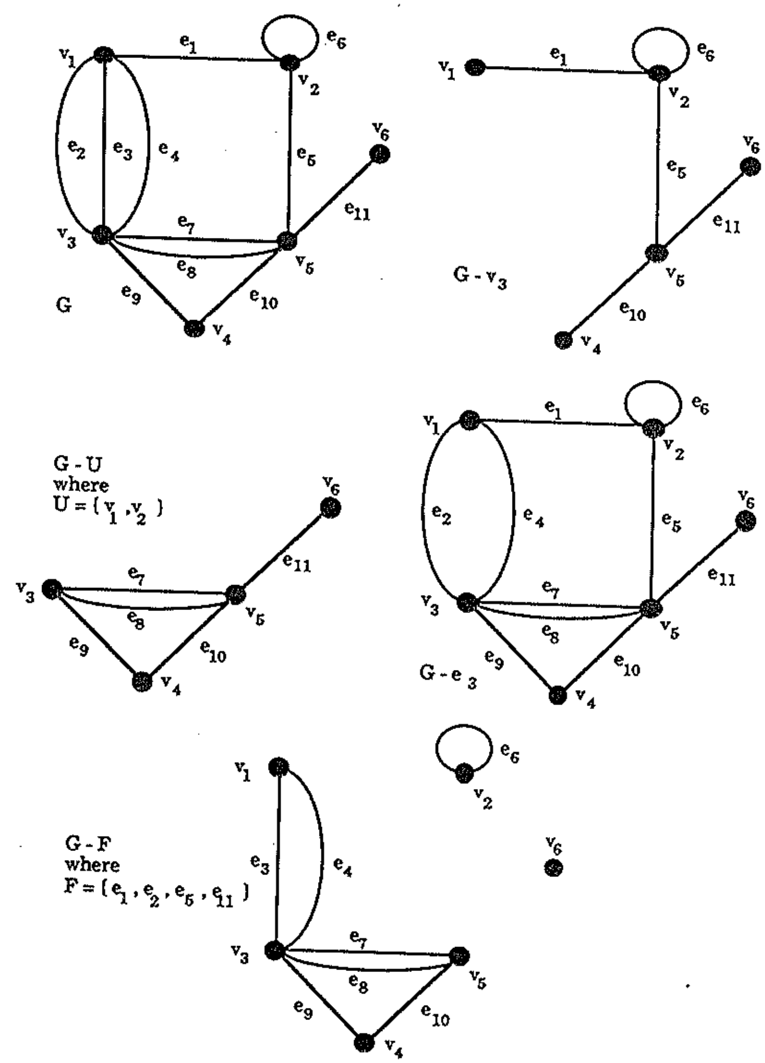


Figure 1.24: A graph and some vertex deleted and edge deleted subgraphs.

If U is a nonempty subset of the vertex set V of the graph G then the subgraph $G[U]$ of G **induced** by U is defined to be the graph having vertex set U and edge set consisting of those edges of G that have both ends in U.

Similarly if F is a nonempty subset of the edge set E of G then the subgraph $G[F]$ of G **induced** by F is the graph whose vertex set is the set of ends of edges in F and whose edge set is F.

For the graph G of Figure 1.24, taking $U = \{v_2, v_3, v_5\}$ and $F = \{e_1, e_3, e_5, e_7, e_9\}$, we get $G[U]$ and $G[F]$ as in Figure 1.25.

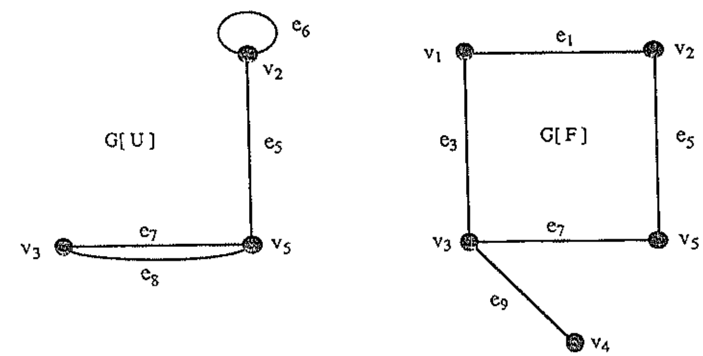


Figure 1.25: $G[U]$ and $G[F]$ for $U = \{v_2, v_3, v_5\}$ and $F = \{e_1, e_3, e_5, e_7, e_9\}$.

Two subgraphs G_1 and G_2 of a graph G are said to be **disjoint** if they have no vertex in common, and **edge disjoint** if they have no edge in common.

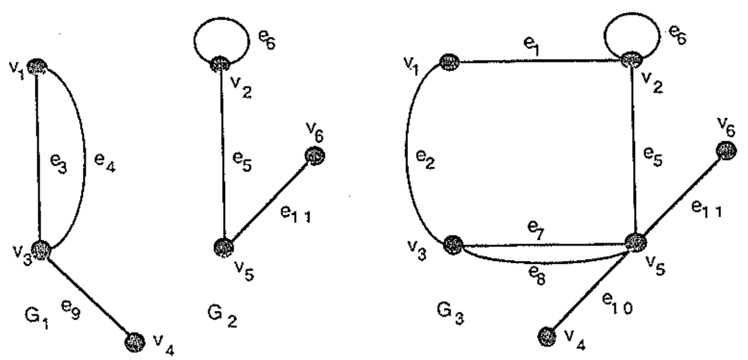


Figure 1.26: G_1 and G_2 are disjoint and G_1 and G_3 are edge disjoint.

For example, Figure 1.26 shows three subgraphs G_1, G_2 and G_3 of the graph G of Figure 1.24. Of these, G_1 and G_2 are disjoint and G_1 and G_3 are edge disjoint.

If two subgraphs G_1 and G_2 are disjoint then they must also be edge disjoint — if not then there would be an edge e of G in both G_1 and G_2 and then the end(s) of e would also be in both G_1 and G_2.

Given two subgraphs G_1 and G_2 of G, the **union** $G_1 \cup G_2$ is the subgraph of G with vertex set consisting of all those vertices which are in either G_1 or G_2 (or both) and with edge set consisting of all those edges which are in either G_1 or G_2 (or both); symbolically

$$V(G_1 \cup G_2) = V(G_1) \cup V(G_2),$$

$$E(G_1 \cup G_2) = E(G_1) \cup E(G_2).$$

For example, Figure 1.27 shows $G[U] \cup G[F]$ for the subgraphs $G[U]$ and $G[F]$ of Figure 1.25, while, for G_1 and G_3 of Figure 1.26, we have $G_1 \cup G_3 = G$.

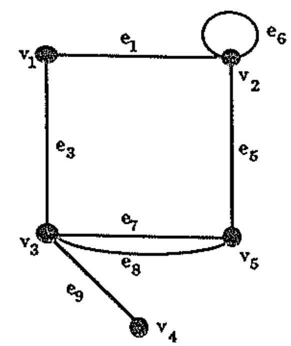


Figure 1.27: $G[U] \cup G[F]$.

If G_1 and G_2 are two subgraphs of G with at least one vertex in common then the **intersection** $G_1 \cap G_2$ is given by

$$V(G_1 \cap G_2) = V(G_1) \cap V(G_2),$$

$$E(G_1 \cap G_2) = E(G_1) \cap E(G_2).$$

Notice the requirement that the two subgraphs G_1 and G_2 must have at least one vertex in common before we can form their intersection — if there is no vertex belonging to both G_1 and G_2 we get $V(G_1) \cap V(G_2) = \emptyset$, denying us the possibility of any reasonable definition of an intersection, since any graph must have at least one vertex.

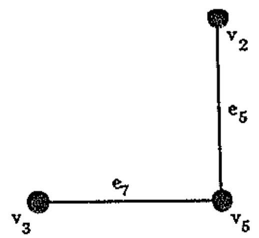


Figure 1.28: $G[U] \cap G[F]$.

Figure 1.28 shows $G[U] \cap G[F]$ for the subgraphs $G[U]$ and $G[F]$ of Figure 1.25, while, for G_1 and G_3 of Figure 1.26, we have $G_1 \cap G_3$ consists of three isolated vertices, namely v_1, v_3 and v_4.

Exercises for Section 1.5

1.5.1 Let G be the graph of Figure 1.29.

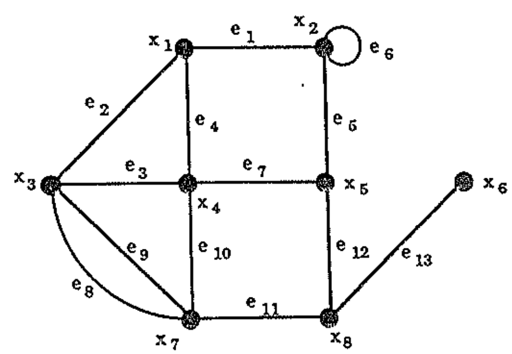


Figure 1.29

(a) Find $G - U$ where $U = \{x_1, x_3, x_5, x_7\}$.
(b) Find $G - F$ where $F = \{e_2, e_4, e_6, e_8, e_{10}, e_{12}\}$.
(c) Find $G[U]$ where $U = \{x_2, x_3, x_4, x_7\}$.
(d) Find $G[F]$ where $F = \{e_1, e_2, e_8, e_{11}\}$.
(e) Find a subgraph H of G isomorphic to K_3.
(f) Is there a subgraph of G isomorphic to K_4?

(g) What is the underlying simple graph of G? In how many ways can this be obtained?
(h) What is the intersection of the two subgraphs you found in (a) and (b)?
(i) What is the union of the two subgraphs you found in (c) and (d)?

1.5.2 Let G be a simple graph with n vertices. The **complement** $\overline{G}$ of G is defined to be the simple graph with the same vertex set as G and where two vertices u and v are adjacent precisely when they are *not* adjacent in G. Roughly speaking then, the complement of G can be obtained from the complete graph K_n by "rubbing out" all the edges of G. Figure 1.30 shows a graph G on 6 vertices and its complement $\overline{G}$.

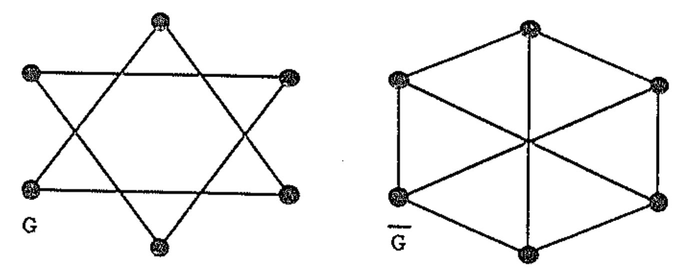


Figure 1.30: A graph and its complement.

Find the complements of the graphs in Figure 1.31.

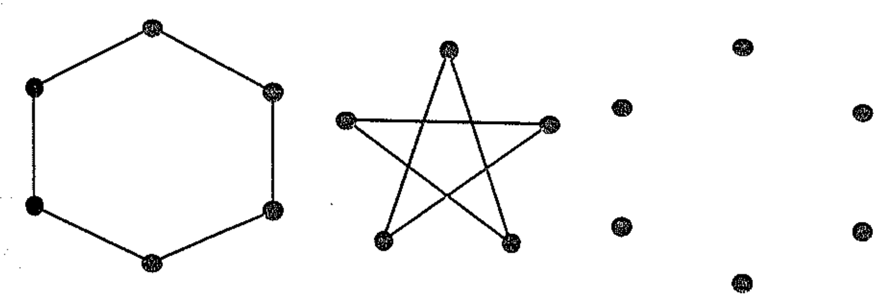


Figure 1.31

1.5.3 A simple graph is called **self-complementary** if it is isomorphic to its own complement.

(a) Find which of the graphs of Figure 1.31 are self-complementary.
(b) Prove that if G is a self-complementary graph with n vertices then n is either $4t$ or $4t+1$ for some integer t. (Hint: consider the number of edges in K_n.)

(c) List all the self-complementary graphs with 4 or 5 vertices.

1.5.4 Let G be a simple graph with n vertices and let $\overline{G}$ be its complement.

(a) Prove that, for each vertex v in G, $d_G(v) + d_{\overline{G}}(v) = n - 1$.

(b) Suppose that G has exactly one even vertex. How many odd vertices does $\overline{G}$ have?

1.5.5 Let G_1 and G_2 be two graphs with no vertex in common. We define the **join** of G_1 and G_2, denoted by $G_1 + G_2$, to be the graph with vertex set and edge set given as follows:

$$V(G_1 + G_2) = V(G_1) \cup V(G_2),$$

$$E(G_1 + G_2) = E(G_1) \cup E(G_2) \cup J$$

where $J = \{x_1x_2 : x_1 \in V(G_1), x_2 \in V(G_2)\}$. Thus J consists of edges which join every vertex of G_1 to every vertex of G_2. We illustrate this in Figure 1.32.

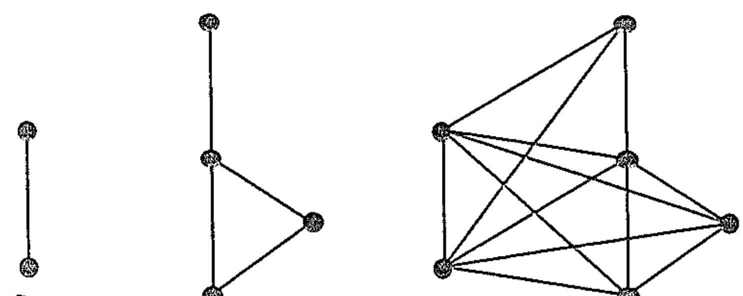


Figure 1.32: $G_1 + G_2$ is the join of G_1 and G_2.

(a) Prove that the join of two vertex disjoint complete graphs is a complete graph.

(b) Prove that the complete bipartite graph $K_{m,n}$ is the join of the complements of K_m and K_n.

(c) Let G_1, G_2 and G_3 be three graphs with no vertex common to any pair. Prove that $(G_1+G_2)+G_3 = G_1+(G_2+G_3)$. (This means that the expression $G_1 + G_2 + G_3$ is unambiguous.)

(d) Prove that if G_1 and G_2 are disjoint simple graphs then the complement of their join is the union of their complements.

1.6 Paths and Cycles

A **walk** in a graph G is a finite sequence

$$W = v_0e_1v_1e_2v_2 \ldots v_{k-1}e_kv_k$$

whose terms are alternately vertices and edges such that, for $1 \le i \le k$, the edge e_i has ends v_{i-1} and v_i.

Thus each edge e_i is immediately preceded and succeeded by the two vertices with which it is incident.

We say that the above walk W is a $v_0 - v_k$ **walk** or a **walk from** v_0 **to** v_k. The vertex v_0 is called the **origin** of the walk W, while v_k is called the **terminus** of W. Note that v_0 and v_k need not be distinct.

The vertices $v_1, \ldots, v_{k-1}$ in the above walk W are called its **internal vertices**. The integer k, the number of edges in the walk, is called the **length** of W.

Note that in a walk there may be repetition of vertices and edges.

In a *simple* graph, a walk $v_0e_1v_1e_2v_2 \ldots v_{k-1}e_kv_k$ is determined by the sequence $v_0v_1 \ldots v_k$ of its vertices, since for each pair $v_{i-1}v_i$ there is only one possible edge with ends determined by the pair. In fact, even in graphs that are not simple, a walk is often simply denoted by a sequence of vertices

$$v_0v_1v_2 \ldots v_{k-1}v_k$$

where consecutive vertices are adjacent. When this is done, it is to be understood that the discussion is valid for every walk with that vertex sequence.

A **trivial walk** is one containing no edges.

Thus, for any vertex v of G, $W = v$ gives a trivial walk. It has length 0.

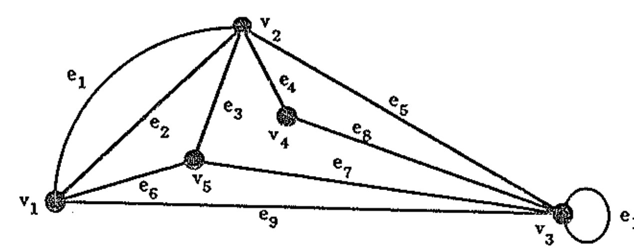


Figure 1.33

In Figure 1.33, $W_1 = v_1\ e_1\ v_2\ e_5\ v_3\ e_{10}\ v_3\ e_5\ v_2\ e_3\ v_5$ and $W_2 = v_1\ e_1\ v_2\ e_1\ v_1\ e_1\ v_2$ are both walks, of length 5 and 3 respectively, from v_1 to v_5 and from v_1 to v_2 respectively.

Given two vertices u and v of a graph G, a $u-v$ walk is called **closed** or **open** depending on whether $u = v$ or $u \neq v$.

The two examples W_1 and W_2 above are both open while $W_3 = v_1\ v_5\ v_2\ v_4\ v_3\ v_1$ is closed.

If the edges $e_1, e_2, \ldots, e_k$ of the walk $W = v_0e_1v_1e_2v_2\ldots e_kv_k$ are distinct then W is called a **trail**.

In other words, a trail is a walk in which no edge is repeated. The examples W_1 and W_2 above are not trails, since, for example, e_5 is repeated in W_1, while e_1 is repeated in W_2. However, W_3 is a trail.

If the vertices $v_0, v_1, \ldots, v_k$ of the walk $W = v_0e_1v_1e_2v_2\ldots e_kv_k$ are distinct then W is called a **path**.
Clearly any two paths with the same number of vertices are isomorphic.
A path with n vertices will sometimes be denoted by P_n. Note that P_n has length $n-1$.

In other words, a path is a walk in which no vertex is repeated. Thus in a path no edge can be repeated either, so every path is a trail. Not every trail is a path, though. For example, W_3 above is not since v_1 is repeated. However $W_4 = v_2\ v_4\ v_3\ v_5v_1$ *is a* path in the graph G (of Figure 1.33).
By definition, every path is a walk. Although the converse of this statement is not true in general, we do have the following.

Theorem 1.3 *Given any two vertices u and v of a graph G, every $u-v$ walk contains a $u-v$ path, i.e., given any walk*

$$W = ue_1v_1\ldots v_{k-1}e_kv$$

then, after some deletion of vertices and edges if necessary, we can find a subsequence P of W which is a $u-v$ path.

Proof If $u = v$, i.e., if W is closed, then the trivial path $P = u$ will do.
Now suppose $u \neq v$, i.e., W is open and let the vertices of W be given, in order, by

$$u = u_0, u_1, u_2, \ldots, u_{k-1}, u_k = v.$$

If none of the vertices of G occurs in W more than once then W is already a $u-v$ path and so we are finished by taking $P = W$.
So now suppose that there are vertices of G that occur in W twice or more. Then there are distinct i, j, with $i < j$, say, such that $u_i = u_j$. If the terms $u_i, u_{i+1}, \ldots, u_{j-1}$ (and the preceding edges) are deleted from W then we obtain a $u-v$ walk W_1 having fewer vertices than W. If there is no repetition of vertices in W_1, then W_1 is a $u-v$ path and setting $P = W_1$ finishes the proof.
If this is not the case, then we repeat the above deletion procedure until finally arriving at a $u-v$ walk that is a path, as required. □

To illustrate the above proof let us look at the walk

$$W = v_1\ e_1\ v_2\ e_5\ v_3\ e_{10}\ v_3\ e_5\ v_2\ e_3\ v_5$$

in our example G of Figure 1.33. Here, (see Figure 1.34),

$$u = u_1 = v_1,\ u_2 = v_2,\ u_3 = v_3,\ u_4 = v_3,\ u_5 = v_2,\ u_6 = v_5 = v.$$

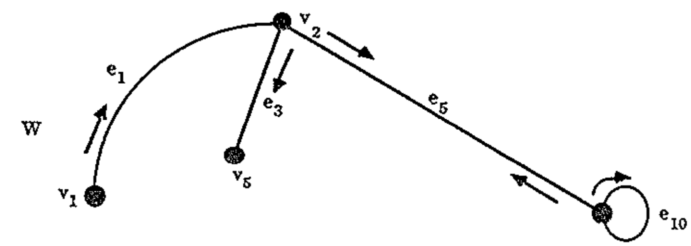


Figure 1.34: The walk W.

Since $u_3 = u_4$ the deletion procedure deletes u_3 (and the edge e_{10}) to give the walk

$$W_1 = v_1\ e_1\ v_2\ e_5\ v_3\ e_5\ v_2\ e_3\ v_5.$$

shown in Figure 1.35. The next stage is to delete $v_2e_5v_3e_5$ to give

$$W_2 = v_1\ e_1\ v_2\ e_3\ v_5,$$

also shown in Figure 1.35, and this is a path, as required.

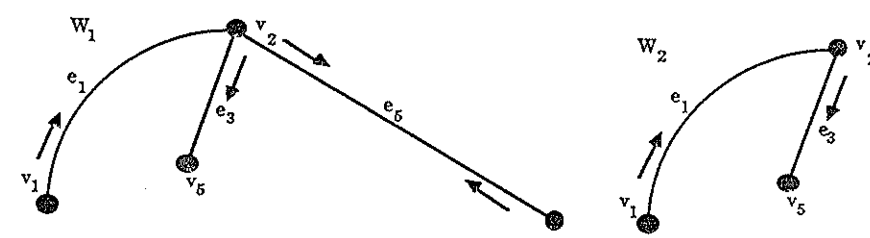


Figure 1.35: The deletion procedure applied to the walk W of Figure 1.34.

A vertex u is said to be **connected** to a vertex v in a graph G if there is a path in G from u to v.

Clearly if u is connected to v then v is connected to u — just reverse the path. Also any vertex u is connected to itself by the trivial path $P = u$.
Moreover, if u is connected to v and v is connected to w, then u is connected to w. To see this, suppose that $W_1 = ue_1\ldots e_kv$ is a $u-v$ path and $W_2 = vf_1\ldots f_tw$ is a

$v - w$ path. Then, joining the two paths together in the obvious way, we get a $u - w$ walk

$$W = ue_1 \dots e_k v f_1 \dots f_t w$$

and, by Theorem 1.3, this contains the required $u - w$ path.

The above process of joining two walks which have a common "end point" to form a longer walk is called **concatenation** — meaning "stringing together".

A graph G is called **connected** if every two of its vertices are connected. A graph that is not connected is called **disconnected**.

Given any vertex u of a graph G, let $C(u)$ denote the set of all vertices in G that are connected to u. Then the subgraph of G induced by $C(u)$ is called the **connected component containing** u, or simply the **component containing** u.

If u and v are two connected vertices in the graph G, i.e., if there is a path from u to v, then, by the remarks above, $C(u) = C(v)$ and so u and v have the same connected component. Conversely, if u and v have the same component then v is in $C(u)$ so v and u must be connected.

Another way of describing a component is as follows: it is a connected subgraph C of the graph G which is not properly contained in any other connected subgraph of G. For if there was a connected subgraph D of G such that $C \subseteq D$ then, since D is connected, every vertex w of D is connected to every vertex of C, showing, from above, that C and D have the same vertex set, and by the definition of component, C has got all the edges of G that have both ends in C and so D can not have any more edges than C, i.e., $C = D$. We also express this by saying that a component of G is a subgraph that is maximal with respect to the property of being connected.

The graph G of Figure 1.36 has 6 components.

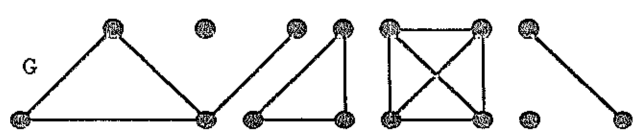


Figure 1.36: A graph with six connected components.

The number of components of a graph G is denoted by $\omega(G)$.

Thus $\omega(G) = 6$ for the graph G of Figure 1.36. Of course a graph G is connected if and only if $\omega(G) = 1$.

It is sometimes not obvious what $\omega(G)$ is or what the components of G look like. For example, in Figure 1.37, $\omega(G) = 2$ but this is not seen immediately.

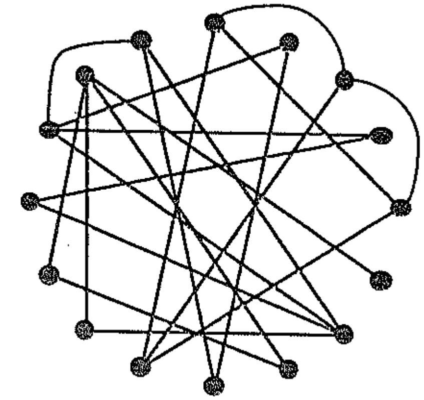


Figure 1.37: $\omega(G) = 2$ — can you see why?

A nontrivial closed trail in a graph G is called a **cycle** if its origin and internal vertices are distinct. In detail, the closed trail $C = v_1 v_2 \dots v_n v_1$ is a cycle if

(i) C has at least 1 edge and

$v_1, v_2, \dots, v_n$ are n distinct vertices.

A cycle of length k, i.e., with k edges, is called a **k-cycle**. A k-cycle is called **odd** or **even** depending on whether k is odd or even.

A 3-cycle is often called a **triangle**.

Clearly any two cycles of the same length are isomorphic. An n-cycle, i.e., a cycle with n vertices, will sometimes be denoted by C_n.

For example in Figure 1.38, $C = v_1\ v_2\ v_3\ v_4\ v_1$ is a 4-cycle, $T = v_1\ v_2\ v_5\ v_3\ v_4\ v_5\ v_1$ is a nontrivial closed trail which is not a cycle (because v_5 occurs twice as an internal vertex), and $C' = v_1\ v_2\ v_5\ v_1$ is a triangle.

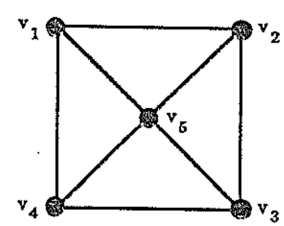


Figure 1.38

Note that a loop is just a 1-cycle. Also, given a pair of parallel edges e_1 and e_2 with distinct end vertices v_1 and v_2, we can form the cycle $v_1\ e_1\ v_2\ e_2\ v_1$ of length 2. Conversely, the two edges of any cycle of length 2 are a pair of parallel edges.

We now characterize bipartite graphs using cycles.

Theorem 1.4 *Let G be a nonempty graph with at least two vertices. Then G is bipartite if and only if it has no odd cycles.*[1]

Proof Suppose that G is bipartite with vertex set V and bipartition $V = X \cup Y$. Let $C = v_0v_1 \dots v_kv_0$ be a cycle of G. For the sake of argument, assume that v_0 is in X. Then, because G is bipartite, v_1 must be in Y. Similarly v_2 must be in X, v_3 in Y, etc. In fact, in general, the odd-indexed vertices v_{2i+1} must be in Y while the even-indexed vertices v_{2i} must be in X. Now, since v_0 is in X, we must have (at the "other end" of the cycle) v_k in Y. Hence k must be an odd number. Thus the cycle $C = v_0v_1 \dots v_kv_0$ is even. Since C was any cycle in the graph G, G has no odd cycles.[2]

Now, to prove the converse, we assume that G is a nonempty graph which has no odd cycles. We wish to show that G is bipartite. Now G will be bipartite if each of its nonempty connected components is bipartite, since, if these components are $C_1, \dots, C_n$ and their vertex sets $V_1, \dots, V_n$ have bipartitions $V_1 = X_1 \cup Y_1, \dots, V_n = X_n \cup Y_n$, then the vertex set V of G has bipartition $V = X \cup Y$ where

$$X = X_0 \cup X_1 \cup X_2 \cup \dots \cup X_n \text{ and } Y = Y_1 \cup Y_2 \cup \dots \cup Y_n,$$

where X_0 is the set of isolated vertices in G. (The details of this argument are asked for in Exercise 1.6.15.) As a result of this it is enough to show that if G is a nonempty *connected* graph with no odd cycles then G is bipartite.

With this assumption, let u be a fixed vertex of G. We define two subsets of the vertex set V of G as follows:

X is the set of all vertices v of G with the property that any shortest $u - v$ path of G has even length,

Y is the set of all vertices w of G with the property that any shortest $u - w$ path of G has odd length,

i.e., X consists of those vertices of G an "even distance" from u, while Y consists of those vertices of G an "odd distance" from u.

Note that u itself is in X. Then, clearly, $V = X \cup Y$ and X and Y have no element in common. We show that $V = X \cup Y$ is a bipartition of G by showing that any edge of G must have one end in X and the other in Y.

Let v and w be two vertices both in X and assume they are adjacent. Let P and Q be a shortest $u - v$ path and a shortest $u - w$ path respectively, say

$$P = u_1, u_2, \dots, u_{2n+1} \text{ and } Q = w_1, w_2, \dots, w_{2m+1},$$

(so that $u = u_1 = w_1$, $v = u_{2n+1}$ and $w = w_{2m+1}$.) Suppose that w' is a vertex that the two paths have in common, and further that w' is the last such vertex. (If $v = w$

[1] Some readers may not be familiar with the phrase "if and only if". It involves two implications. Here for example it says:
(1) *if* a nonempty graph is bipartite *then* it has no odd cycles, *and also the converse statement, namely:*
(2) *if* a nonempty graph has no odd cycles *then* it is bipartite.
Another way of describing this is to say that the following two statements are equivalent:
(i) the nonempty graph G is bipartite ;
(ii) the nonempty graph G has no odd cycles.

[2] We are not finished the proof — in fact only half-finished; we still have to prove "the other way".

then of course $w' = v = w$. Moreover, there is always such a vertex w' since the vertex u is common to both paths.) Then that part of P from u to w' is a shortest path from u to w' and that part of Q from u to w' is a shortest path from u to w' also. In other words, we have two shortest paths from u to w'. It follows that, since these two paths have the same length, there exists an i such that $w' = u_i = w_i$. However this produces an odd cycle in G:

$$C = \underbrace{u_iu_{i+1} \dots u_{2n+1}}_{*}\underbrace{w_{2m+1}w_{2m} \dots w_i}_{**}$$

since if i is odd then the above parts $*$ and $**$ are both of even length while if i is even then they are both of odd length, giving the total length of C as odd + 1 + odd or even + 1 + even, in either case odd. Since G has no odd cycles, something is wrong (!), namely the assumption that v and w are adjacent.

Hence there are no edges in G joining vertices of X. A similar argument shows that there are no edges of G joining vertices of Y. Hence G is bipartite, as required. □

Exercises for Section 1.6

1.6.1 Let G be the graph of Figure 1.39.

(a) Find a closed walk of length 6. Is your walk a trail?
(b) Find an open walk of length 12. Is your walk a path?
(c) Find a closed trail of length 6. Is your trail a cycle?
(d) What is the length of the longest cycle in G?
(e) What is the length of a longest path in G? How many paths in G are there of this length?

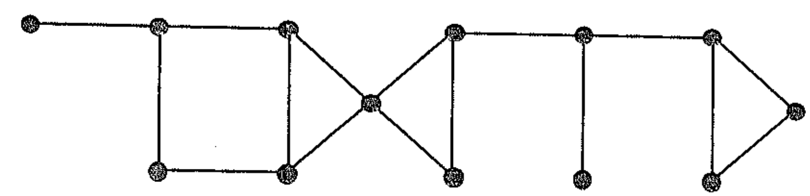


Figure 1.39

1.6.2 Let G be a graph with 15 vertices and 4 connected components. Prove that G has at least one component with at least 4 vertices. What is the largest number of vertices that a component of G can have?

1.6.3 Let v be a vertex in the graph G and let T be a closed trail with origin (and terminus) v. Modify the proof of Theorem 1.3 to show that T contains a cycle with origin v.

1.6.4 For each $n \geq 4$, the **wheel** graph, W_n, with n vertices, is defined to be the join $K_1 + C_{n-1}$, of an isolated vertex with a cycle of length $n-1$. (See Exercise 1.5.5 for the definition of *join.*)

(a) Draw the graphs W_4, W_5, W_6 and W_7 in such a way that you can see why the name *wheel* is given to this family of graphs.

(b) Show that W_n contains a cycle of length k for each k, $4 \leq k \leq n$.

1.6.5 Prove that if u is an odd vertex in a graph G then there must be a path in G from u to another odd vertex v of G.

1.6.6 Give an example of a graph in which the length of the longest cycle is 9 and the length of the shortest cycle is 4.

1.6.7 The graph G of Figure 1.40 is known as the **Petersen graph**. Find in G

(a) a trail of length 5,

(b) a path of length 9,

(c) cycles of lengths 5 , 6 , 8 and 9.

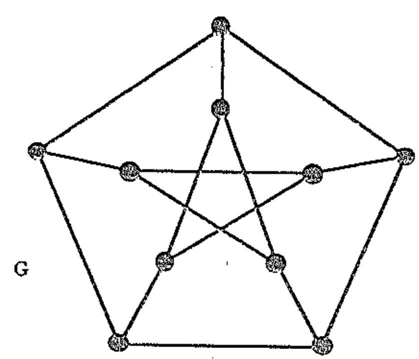


Figure 1.40: The Petersen graph.

1.6.8 For any two vertices u and v connected by a path in a graph G, we define the **distance** between u and v, denoted by $d(u,v)$, to be the length of a shortest u-v path. If there is no path connecting u and v we define $d(u,v)$ to be infinite. (This idea of distance was used in the proof of Theorem 1.4.)

(a) Prove that for any vertices u, v, w in G we have

$$d(u,w) \leq d(u,v) + d(v,w).$$

(b) Prove that if $d(u,v) \geq 2$, then there is a vertex z in G such that

$$d(u,v) = d(u,z) + d(z,v).$$

(c) Prove that in the Petersen graph of Exercise 1.6.7 above, $d(u,v) \leq 2$ for any pair of vertices u, v. (By using the symmetry of the graph you need not look at *every* pair of vertices.)

1.6.9 Let G be a connected graph with vertex set V. For each $v \in V$, the **eccentricity** of v, denoted by $e(v)$, is defined by

$$e(v) = \max\{d(u,v) : u \in V, u \neq v\}.$$

The **radius** of G, denoted by rad G , is defined by

$$\text{rad } G = \min\{e(v) : v \in V\},$$

while the **diameter** of G, denoted by diam G, is defined by

$$\text{diam } G = \max\{e(v) : v \in V\}.$$

Thus the diameter of G is given by $\max\{d(u,v) : u, v \in V\}$.

(a) Find the radius and the diameter of the graph of Figure 1.39 and the Petersen graph (Figure 1.40).

(b) Prove that for any connected graph G,

$$\text{rad } G \leq \text{diam } G \leq 2 \text{ rad } G.$$

(c) Find the radius and the diameter of the wheel graphs W_n of Exercise 1.6.4.

(d) Which simple graphs have diameter 1?

1.6.10 Let G be a simple connected graph. The **square** of G, denoted by G^2, is defined to be the graph with the same vertex set as G and in which two vertices u and v are joined by an edge if and only if in G we have $1 \leq d(u,v) \leq 2$. An example of a graph G and its square is shown in Figure 1.41.

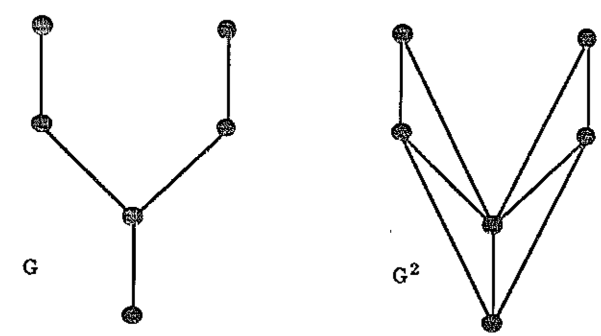


Figure 1.41: A graph and its square.

(a) Show that the square of $K_{1,3}$ is K_4. Can you find two more graphs whose square is K_4?

(b) Draw the squares of the paths P_4, P_5, P_6, the cycles C_5, C_6 and the wheels W_5 and W_6.

1.6.11 Let G be a simple graph with n vertices, where $n \geq 2$. Prove that G has two vertices u and v with $d(u) = d(v)$. (Hint: if G is nonempty, consider $G - e$ where e is an edge of G, and use induction on the number of edges of G.)

1.6.12 Let G be a simple graph. Show that if G is not connected then its complement $\overline{G}$ is connected.

1.6.13 A **complete tripartite graph** G is a simple graph G in which the vertex set V is the union of three nonempty subsets V_1, V_2 and V_3 where $V_i \cap V_j = \emptyset$ for $i \neq j$ and an edge joins two vertices u, v of G if and only if u and v do not belong to the same V_i. If V_1, V_2 and V_3 have r,s and t elements respectively where $r \leq s \leq t$ we denote G by $K_{r,s,t}$.

(a) Draw $K_{1,2,2}$, $K_{2,2,2}$, $K_{2,3,3}$.

(b) How many edges are there in $K_{r,s,t}$?

(c) Formulate a definition of a **complete n-partite graph** for any $n \geq 3$.

1.6.14 Which of the graphs in Figure 1.42 are bipartite? Justify your answer using Theorem 1.4 and redraw those that are bipartite showing the bipartite property more clearly.

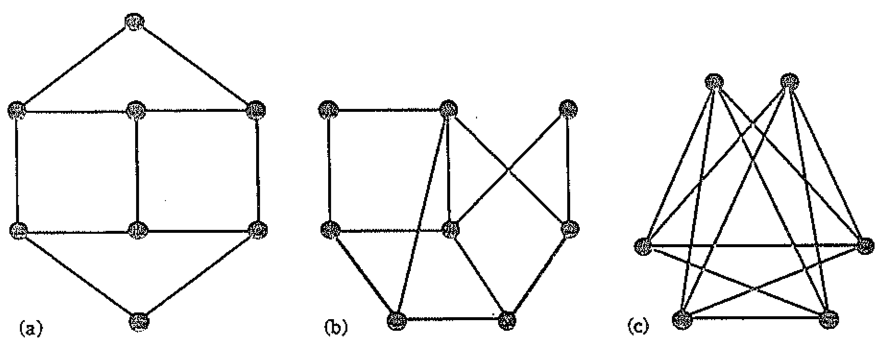


Figure 1.42: Which of these graphs are bipartite?

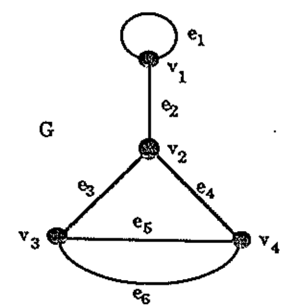
1.6.15 Let G be a graph each of whose nonempty connected components is a bipartite graph. Assuming that G has at least one nonempty component, prove that G is bipartite. (This is used in the proof of Theorem 1.4 — see there on how to get started.)

1.7 The Matrix Representation of Graphs

There are essentially two different ways of representing a graph inside a computer, namely by using the adjacency matrix or the incidence matrix of a graph.

Let G be a graph with n vertices, listed as $v_1, \ldots, v_n$. The **adjacency matrix** of G, with respect to this particular listing of the n vertices of G, is the $n \times n$ matrix $A(G) = (a_{ij})$ where the (i,j)th entry a_{ij} is the number of edges joining the vertex v_i to the vertex v_j.

Figure 1.43 shows a graph G with vertices listed as $v_1, \ldots, v_4$ and its adjacency matrix $A(G)$ with respect to this listing.



$A(G)$: a 4×4 matrix.
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Figure 1.43: A graph and its adjacency matrix.

Note that in $A(G)$ we have $a_{ij} = a_{ji}$ for each i and j. A matrix with this property is called **symmetric**. Note also that if G has no loops then all the entries of the main diagonal of $A(G)$ are 0, while if G has no parallel edges then the entries of $A(G)$ are either 0 or 1.

Given an $n \times n$ symmetric matrix $A = (a_{ij})$ in which all the entries are non-negative integers, we can associate with it a graph G whose adjacency matrix is A, simply by letting G have n vertices, labelled 1 to n, say, and joining vertex i to vertex j by a_{ij} edges. Figure 1.44 shows such a symmetric matrix A and a graph produced from it.

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \text{gives G:}$$

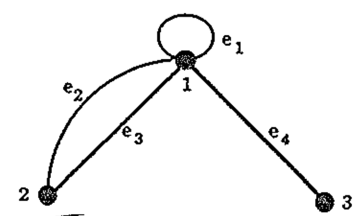


Figure 1.44: A symmetric matrix A of non-negative integers and a graph G with $A(G) = A$.

In this example the matrix multiplication $A \times A = A^2$ gives

$$A^2 = \begin{bmatrix} 6 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = (b_{ij}), \text{ say,}$$

and in fact b_{ij} gives the number of walks of length 2 from vertex i to vertex j. For example, $b_{11} = 6$ and the 6 walks of length 2 from vertex 1 to itself are

$$1e_1 1e_1 1,\ 1e_2 2e_2 1,\ 1e_3 2e_3 1,\ 1e_2 2e_3 1,\ 1e_3 2e_2 1,\ 1e_4 3e_4 1.$$

As a further example, $b_{23} = b_{32} = 2$ and the 2 walks of length 2 from vertex 2 to vertex 3 are

$$2e_2 1e_4 3,\ 2e_3 1e_4 3.$$

This is a particular case of the following result:

Theorem 1.5 *Let G be a graph with n vertices $v_1, \ldots, v_n$ and let A denote the adjacency matrix of G with respect to this listing of the vertices. Let k be any positive integer and let A^k denote the matrix multiplication of k copies of A. Then the (i,j)th entry of A^k is the number of different $v_i - v_j$ walks in G of length k.*

Proof The proof is by mathematical induction on k. For $k = 1$ the theorem says that the (i,j)th entry of A is the number of different $v_i - v_j$ walks in G of length 1, which is true by the definition of the adjacency matrix since a length 1 walk from v_i to v_j is just an edge from v_i to v_j.

Now suppose that the result is true for A^{k-1}, where k is some integer greater than 1. We wish to prove it true for A^k. Thus, setting $A^{k-1} = (b_{ij})$, we are assuming that b_{ij} is the number of different walks of length $k-1$ from v_i to v_j and we want to prove that if $A^k = (c_{ij})$ then c_{ij} is the number of different walks of length k from v_i to v_j. Set $A = (a_{ij})$. Since $A^k = A^{k-1} \times A$, from the definition of matrix multiplication we get

$$c_{ij} = \sum_{t=1}^{n} ((i,t)\text{th element of } A^{k-1}) \times ((t,j)\text{th element of } A) = \sum_{t=1}^{n} b_{it}\, a_{tj}.$$

Now every $v_i - v_j$ walk of length k consists of a $v_i - v_t$ walk of length $k-1$, where v_t is adjacent to v_j, followed by an edge $v_t v_j$. Since there are b_{it} such walks of length $k-1$ and a_{tj} such edges for each vertex v_t, the total number of all $v_i - v_j$ walks is

$$\sum_{t=1}^{n} b_{it}\, a_{tj}.$$

Since this, by the above, is just c_{ij} we have established the result for A^k. Thus assuming the result true for $k-1$ we have proved it true for k. Hence by induction the proof is complete. □

We can use the above result to determine whether or not a graph is connected:

Theorem 1.6 *Let G be a graph with n vertices $v_1, \ldots, v_n$ and let A denote the adjacency matrix of G with respect to this listing of the vertices. Let $B = (b_{ij})$ be the matrix*

$$B = A + A^2 + \cdots + A^{n-1}.$$

Then G is a connected graph if and only if for every pair of distinct indices i, j we have $b_{ij} \neq 0$, i.e., if and only if B has no zero entries off the main diagonal.

Proof Let $a_{ij}^{(k)}$ denote the (i,j)th entry of the matrix A^k for each $k = 1, \ldots, n-1$. Then

$$b_{ij} = a_{ij}^{(1)} + a_{ij}^{(2)} + \cdots + a_{ij}^{(n-1)}.$$

However, by Theorem 1.5, $a_{ij}^{(k)}$ denotes the number of distinct walks of length k from v_i to v_j and so

$$\begin{aligned} b_{ij} &= (\text{number of different } v_i - v_j \text{ walks of length } 1) \\ &+ (\text{number of different } v_i - v_j \text{ walks of length } 2) \\ &\ \ \vdots \\ &+ (\text{number of different } v_i - v_j \text{ walks of length } (n-1)), \end{aligned}$$

i.e., b_{ij} is the number of different $v_i - v_j$ walks of length less than n.

Now suppose that G is connected. Then for every pair of distinct indices i, j there is a path from v_i to v_j. Since G has only n vertices this path goes through at most n vertices and so it has length less than n, i.e., there is at least 1 path from v_i to v_j of length less than n. Hence $b_{ij} \neq 0$ for each i, j with $i \neq j$, as required.

Conversely, suppose that for each distinct pair i, j we have $b_{ij} \neq 0$. Then, from above, there is at least 1 walk (of length less than n) from v_i to v_j. In particular, v_i is connected to v_j. Thus G is a connected graph, as required, since i and j were an arbitrary pair of distinct vertices. □

As an illustration, we use Theorem 1.6 to find whether a particular graph G is connected or not using its adjacency matrix $A(G)$. Suppose that $A(G)$ is the following matrix A:

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

Here $n = 5$ so the theorem tells us to look at $B = A + A^2 + A^3 + A^4$. Now

$$A^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 1 & 1 & 0 & 0 & 2 \end{bmatrix} \text{ so } A + A^2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 2 & 0 & 2 & 1 & 1 \\ 0 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{bmatrix},$$

$$A^3 = \begin{bmatrix} 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 2 & 1 & 0 & 0 & 3 \\ 1 & 2 & 0 & 0 & 3 \\ 0 & 0 & 3 & 3 & 0 \end{bmatrix} \text{ so } A + A^2 + A^3 = \begin{bmatrix} 1 & 0 & 3 & 1 & 1 \\ 0 & 1 & 1 & 3 & 1 \\ 3 & 1 & 2 & 1 & 4 \\ 1 & 3 & 1 & 2 & 4 \\ 1 & 1 & 4 & 4 & 2 \end{bmatrix}, \text{ and}$$

$$A^4 = \begin{bmatrix} 2 & 1 & 0 & 0 & 3 \\ 1 & 2 & 0 & 0 & 3 \\ 0 & 0 & 5 & 4 & 0 \\ 0 & 0 & 4 & 5 & 0 \\ 3 & 3 & 0 & 0 & 6 \end{bmatrix} \text{ so } B = A + A^2 + A^3 + A^4 = \begin{bmatrix} 3 & 1 & 3 & 1 & 4 \\ 1 & 3 & 1 & 3 & 4 \\ 3 & 1 & 7 & 5 & 4 \\ 1 & 3 & 5 & 7 & 4 \\ 4 & 4 & 4 & 4 & 8 \end{bmatrix}.$$

Since this last matrix, B, has no nonzero entries off the main diagonal we conclude that the graph is connected. Figure 1.45 gives a drawing of the graph G.

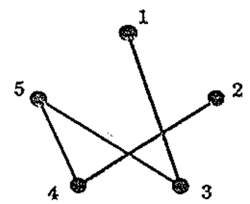


Figure 1.45

Often we do not have to go up to A^{n-1}. For example, for the graph with adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \text{ we have } A^2 = \begin{bmatrix} 5 & 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 1 & 2 & 1 & 1 \\ 2 & 1 & 3 & 1 & 2 & 1 \\ 2 & 2 & 1 & 3 & 1 & 2 \\ 2 & 1 & 2 & 1 & 3 & 1 \\ 1 & 1 & 1 & 2 & 1 & 1 \end{bmatrix}$$

and here, by Theorem 1.5, since every entry is nonzero there is at least one walk of length 2 from every vertex in the graph to every other vertex and so G is connected. Notice however that in our worked example above, we *did* have to go up to A^{n-1}.

For the graph G with adjacency matrix

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

we have $n = 5$ and (check) $B = A + A^2 + A^3 + A^4$ is

$$\begin{bmatrix} 3 & 3 & 0 & 0 & 3 \\ 3 & 6 & 0 & 0 & 3 \\ 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 & 0 \\ 3 & 3 & 0 & 0 & 3 \end{bmatrix}.$$

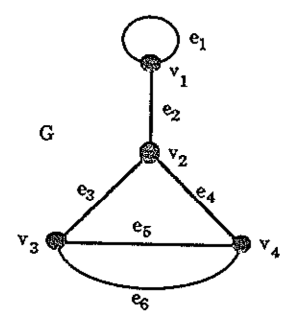
So, since B has zeros off the diagonal, G is not connected. In fact, looking at B we can identify two components, one with vertices v_1, v_2, v_5, the other with v_3, v_4.

Another matrix associated with a graph G is given as follows.

Suppose that G has n vertices, listed as $v_1, \dots, v_n$, and t edges, listed as $e_1, \dots, e_t$. The **incidence matrix** of G, with respect to these particular listings of the vertices and edges of G, is the $n \times t$ matrix $M(G) = (m_{ij})$ where m_{ij} is the number of times that the vertex v_i is incident with the edge e_j, i.e.,

$$m_{ij} = \begin{cases} 0 & \text{if } v_i \text{ is not an end of } e_j \\ 1 & \text{if } v_i \text{ is an end of the non-loop } e_j \\ 2 & \text{if } v_i \text{ is an end of the loop } e_j. \end{cases}$$

Figure 1.46 shows a graph G, with four vertices $v_1, \dots, v_4$ and six edges $e_1, \dots, e_6$, and its incidence matrix $M(G)$ with respect to these listings of the vertices and edges.



$M(G)$: a 4×6 matrix.

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Figure 1.46: A graph and its incidence matrix.

Notice that the sum of the elements in the ith row of $M(G)$ gives us the degree of the vertex v_i, while the sum of the elements in each column is 2 (corresponding to the 2 ends of the edge).

Exercises for Section 1.7

1.7.1 Draw the graphs having the following matrices as their adjacency matrices.

(a) $\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 2 \\ 0 & 2 & 1 & 2 \\ 0 & 2 & 2 & 1 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}$.

1.7.2 Write down the adjacency matrix and the incidence matrix for each of the graphs in Figure 1.47 using the ordering of the vertices and edges given.

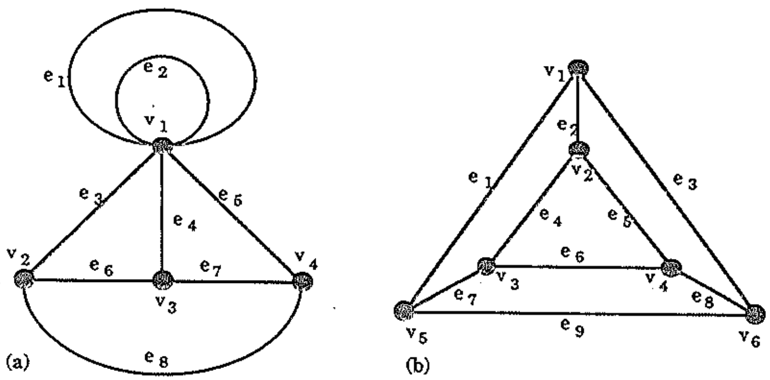


Figure 1.47

1.7.3 Let G be a simple graph and A be its adjacency matrix. Prove that the entries on the main diagonal of A^2 give the degrees of the vertices of G. Does this remain true if we drop the "simple" condition?

1.7.4 Let G be a bipartite graph. Show that we can list the vertices of G so that the corresponding adjacency matrix of G has the form

$$A[G] = \left[\begin{array}{c|c} O & C \\ \hline D & O \end{array}\right]$$

where the O's, C and D are submatrices, the submatrices O have entries all zero, and C is the "mirror-image" of D. (Formally, C is the matrix transpose of D.)

1.7.5 Use the powers of the adjacency matrix to determine if the graphs of Exercise 1.7.1 are connected or not.

1.8 Fusion

Let u and v be distinct vertices of a graph G. We can construct a new graph G_1 by **fusing** (or **identifying**) the two vertices, namely by replacing them by a single new vertex x such that every edge that was incident with either u or v in G is now incident with x, i.e., the end u and the end v become end x.

Thus the new graph G_1 has one less vertex than G but the same number of edges as G and the degree of the new vertex x is the sum of the degrees of u and v.

We illustrate the process in Figure 1.48.

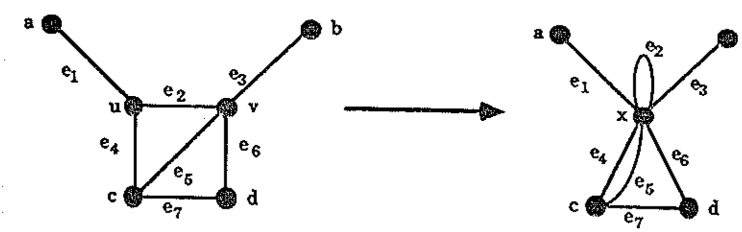


Figure 1.48: The fusion of vertices u and v.

The fusion process when applied to adjacent vertices u and v does not alter the number of connected components of the graph — the component containing u (and v) in G gets changed to the component containing the fused vertex x in G_1, while all other components of G remain unaltered.

If v_i is fused to its neighbour v_j to form the new vertex w then, since each edge of the form v_iv_k or of the form v_jv_k gets changed to one of the form wv_k, it follows that in the adjacency matrix of the new graph G_1 the entries in the row (and column) corresponding to w are just the sum of the corresponding entries given by v_i and v_j in the adjacency matrix for G.

One can more precisely describe this as the following two step process:

The adjacency matrix after fusion of two adjacent vertices u and v.

Step 1. Change u's row to the sum of u's row with v's row and (symmetrically) change u's column to the sum of u's column with v's column.

Step 2. Delete the row and column corresponding to v. The resulting matrix is the adjacency matrix of the new graph G_1.

As an example, we consider the fusion of u and v given in Figure 1.48. Listing the vertices of G as a, u, b, v, c, d, the corresponding adjacency matrix is

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Since u and v are the second and fourth vertex in the list, Step 1 of the process gives the matrix

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 2 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Step 2 now deletes the fourth row and column (corresponding to v) of this matrix to give

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

and this is the adjacency matrix of G_1 (as the reader can check from Figure 1.48).

We now use the above remarks and techniques to give an alternative way of finding whether or not a graph G is connected. The problem with the $B = A + A^2 + \cdots + A^{n-1}$ method described earlier is that it involves much matrix multiplication and this can be very time-consuming when dealing with a large number of vertices (i.e., when n is large). The following method is more efficient and also tells us how many connected components the graph has.

The fusion algorithm for connectedness.

Step 1. Replace G by its underlying simple graph — it is easy to see that the underlying simple graph has exactly the same number of connected components as G and it has the advantage of having all entries in its adjacency matrix as either 0 or 1. To get the adjacency matrix of the new graph just replace all nonzero entries off the diagonal by 1 and make all entries on the diagonal 0. We denote the underlying simple graph of G also by G.

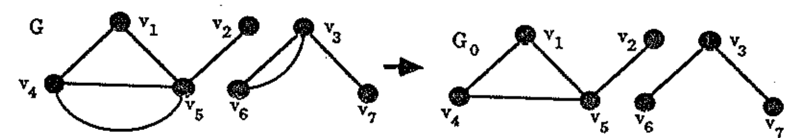
Step 2. Fuse vertex v_1 to the first of the vertices $v_2, \ldots, v_n$ with which it is adjacent to give a new graph, also denoted by G, in which the new vertex is also denoted by v_1. (The above two step process gives the adjacency matrix $A(G)$.)

Step 3. Carry out step 1 on the new graph G.

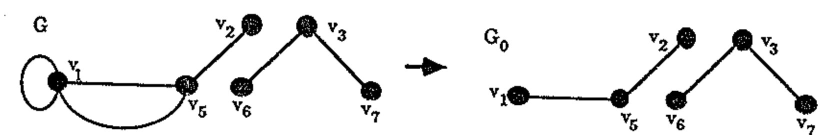
Step 4. Carry out steps 2 and 3 repeatedly with v_1 and the vertices of the new graphs until v_1 is not adjacent to any of the other vertices.

Step 5. Carry out steps 2–4 on the vertex v_2 (instead of v_1) of the latest graph and then on all the remaining vertices of the resulting graphs in turn. The final graph is empty and the number of its (isolated) vertices is the number of connected components of the initial graph G.

We illustrate the algorithm in Figures 1.49 and 1.50 starting with a graph G with seven vertices listed as $v_1, \ldots, v_7$. The resulting graphs are shown in pairs G and G_0, where G_0 is the underlying simple graph of G, with their adjacency matrices shown immediately below. Following through the algorithm, after the initial simplification, vertex v_1 is first fused with v_4 (and the new vertex labelled v_1), as shown in Figure 1.49. Then (see Figure 1.50) v_1 is fused with v_5 and then with v_2. Next v_3 is fused with v_6 and v_7 in turn — the new vertices are labelled v_3 in both cases.

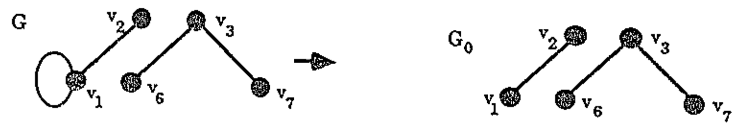


$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 1.49: v_1 is fused with v_4.



$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Figure 1.50: v_1 is fused with v_5 and then with v_2. Next v_3 is fused with v_6 and then with v_7.

Since the final adjacency matrix $A(G)$ in Figure 1.50 is a 2×2 matrix we can conclude that the original graph G has 2 connected components. Of course this is obvious from our drawing of the original G in Figure 1.49. However usually a computer has no such drawing to help it but only a knowledge of the graph in matrix form.

Exercises for Section 1.8

1.8.1 Use the fusion process to determine whether the graphs of Exercise 1.7.1, specified by their adjacency matrix, are connected or not. At each stage of the process, give both the corresponding graph and its adjacency matrix.

1.8.2 Give a formal proof of the remark made in the main text that if two adjacent vertices u and v of the graph G are fused to produce the graph G_1 then $\omega(G) = \omega(G_1)$.