1 Graphs and Subgraphs

1.1 GRAPHS AND SIMPLE GRAPHS

Many real-world situations can conveniently be described by means of a diagram consisting of a set of points together with lines joining certain pairs of these points. For example, the points could represent people, with lines joining pairs of friends; or the points might be communication centres, with lines representing communication links. Notice that in such diagrams one is mainly interested in whether or not two given points are joined by a line; the manner in which they are joined is immaterial. A mathematical abstraction of situations of this type gives rise to the concept of a graph.

A *graph* G is an ordered triple $(V(G), E(G), \psi_G)$ consisting of a nonempty set $V(G)$ of *vertices*, a set $E(G)$, disjoint from $V(G)$, of *edges*, and an *incidence function* ψ_G that associates with each edge of G an unordered pair of (not necessarily distinct) vertices of G. If e is an edge and u and v are vertices such that $\psi_G(e) = uv$, then e is said to *join* u and v; the vertices u and v are called the *ends* of e.

Two examples of graphs should serve to clarify the definition.

Example 1

$$G = (V(G), E(G), \psi_G)$$

where

$$V(G) = \{v_1, v_2, v_3, v_4, v_5\}$$
$$E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$$

and ψ_G is defined by

$$\psi_G(e_1) = v_1v_2,\ \psi_G(e_2) = v_2v_3,\ \psi_G(e_3) = v_3v_3,\ \psi_G(e_4) = v_3v_4$$
$$\psi_G(e_5) = v_2v_4,\ \psi_G(e_6) = v_4v_5,\ \psi_G(e_7) = v_2v_5,\ \psi_G(e_8) = v_2v_5$$

Example 2

$$H = (V(H), E(H), \psi_H)$$

where

$$V(H) = \{u, v, w, x, y\}$$
$$E(H) = \{a, b, c, d, e, f, g, h\}$$

and ψ_H is defined by

$$\psi_H(a) = uv, \quad \psi_H(b) = uu, \quad \psi_H(c) = vw, \quad \psi_H(d) = wx$$
$$\psi_H(e) = vx, \quad \psi_H(f) = wx, \quad \psi_H(g) = ux, \quad \psi_H(h) = xy$$

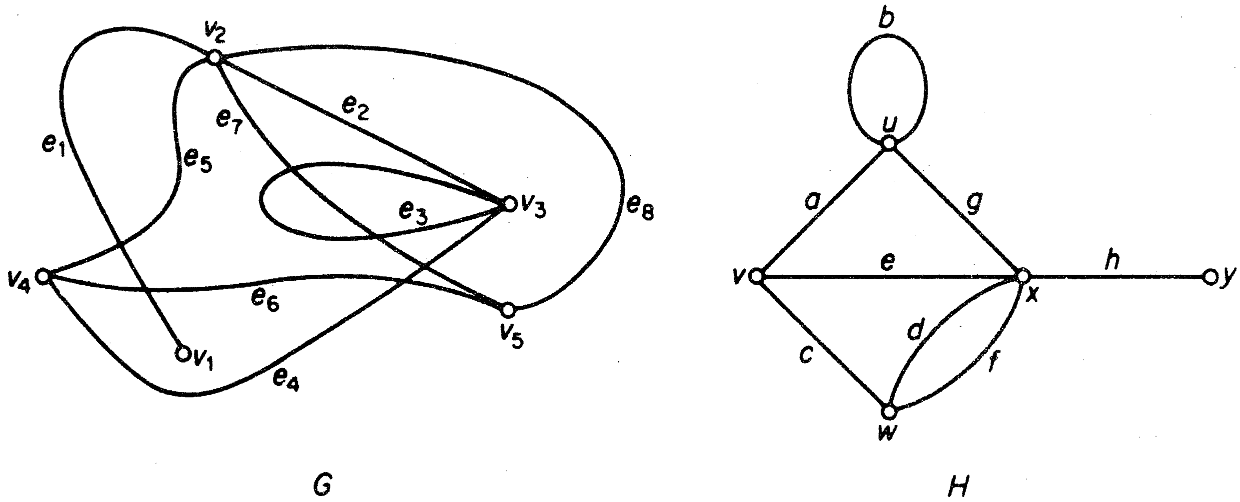


Figure 1.1. Diagrams of graphs G and H

Graphs are so named because they can be represented graphically, and it is this graphical representation which helps us understand many of their properties. Each vertex is indicated by a point, and each edge by a line joining the points which represent its ends.† Diagrams of G and H are shown in figure 1.1. (For clarity, vertices are depicted here as small circles.)

There is no unique way of drawing a graph; the relative positions of points representing vertices and lines representing edges have no significance. Another diagram of G, for example, is given in figure 1.2. A diagram of a graph merely depicts the incidence relation holding between its vertices and edges. We shall, however, often draw a diagram of a graph and refer to it as the graph itself; in the same spirit, we shall call its points 'vertices' and its lines 'edges'.

Note that two edges in a diagram of a graph may intersect at a point that

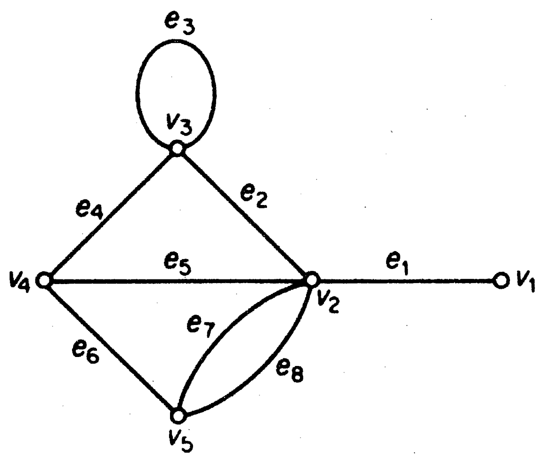


Figure 1.2. Another diagram of G

† In such a drawing it is understood that no line intersects itself or passes through a point representing a vertex which is not an end of the corresponding edge—this is clearly always possible.

is not a vertex (for example e_1 and e_6 of graph G in figure 1.1). Those graphs that have a diagram whose edges intersect only at their ends are called *planar*, since such graphs can be represented in the plane in a simple manner. The graph of figure 1.3*a* is planar, even though this is not immediately clear from the particular representation shown (see exercise 1.1.2). The graph of figure 1.3*b*, on the other hand, is nonplanar. (This will be proved in chapter 9.)

Most of the definitions and concepts in graph theory are suggested by the graphical representation. The ends of an edge are said to be *incident* with the edge, and vice versa. Two vertices which are incident with a common edge are *adjacent*, as are two edges which are incident with a common vertex. An edge with identical ends is called a *loop*, and an edge with distinct ends a *link*. For example, the edge e_3 of G (figure 1.2) is a loop; all other edges of G are links.

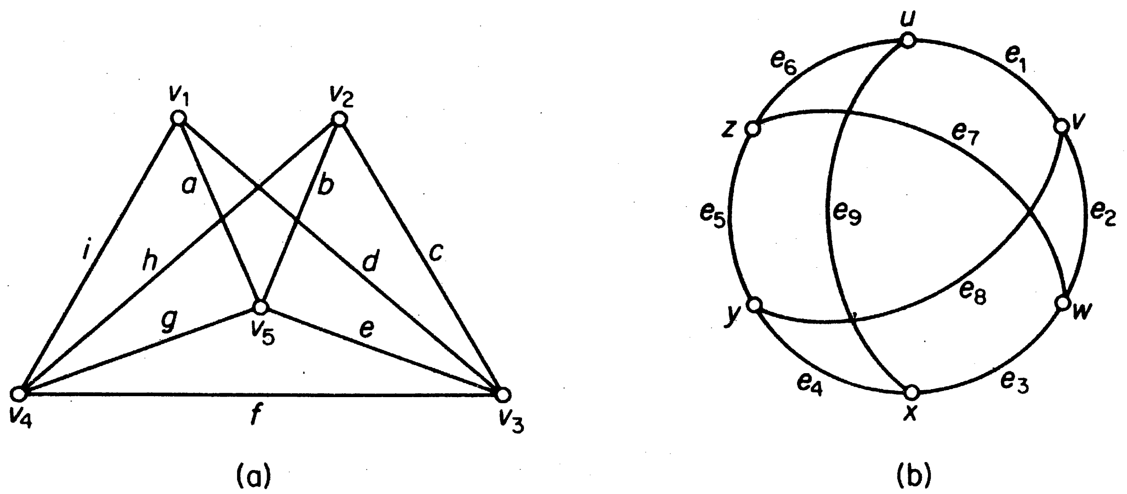


Figure 1.3. Planar and nonplanar graphs

A graph is *finite* if both its vertex set and edge set are finite. In this book we study only finite graphs, and so the term 'graph' always means 'finite graph'. We call a graph with just one vertex *trivial* and all other graphs *nontrivial.*

A graph is *simple* if it has no loops and no two of its links join the same pair of vertices. The graphs of figure 1.1 are not simple, whereas the graphs of figure 1.3 are. Much of graph theory is concerned with the study of simple graphs.

We use the symbols $\nu(G)$ and $\varepsilon(G)$ to denote the numbers of vertices and edges in graph G. Throughout the book the letter G denotes a graph. Moreover, when just one graph is under discussion, we usually denote this graph by G. We then omit the letter G from graph-theoretic symbols and write, for instance, V, E, ν and ε instead of $V(G)$, $E(G)$, $\nu(G)$ and $\varepsilon(G)$.

Exercises

1.1.1 List five situations from everyday life in which graphs arise naturally.

1.1.2 Draw a different diagram of the graph of figure 1.3*a* to show that it is indeed planar.

1.1.3 Show that if G is simple, then $\varepsilon \leq \binom{\nu}{2}$.

1.2 GRAPH ISOMORPHISM

Two graphs G and H are *identical* (written $G = H$) if $V(G) = V(H)$, $E(G) = E(H)$, and $\psi_G = \psi_H$. If two graphs are identical then they can clearly be represented by identical diagrams. However, it is also possible for graphs that are not identical to have essentially the same diagram. For example, the diagrams of G in figure 1.2 and H in figure 1.1 look exactly the same, with the exception that their vertices and edges have different labels. The graphs G and H are not identical, but isomorphic. In general, two graphs G and H are said to be *isomorphic* (written $G \cong H$) if there are bijections $\theta : V(G) \to V(H)$ and $\phi : E(G) \to E(H)$ such that $\psi_G(e) = uv$ if and only if $\psi_H(\phi(e)) = \theta(u)\theta(v)$; such a pair (θ, ϕ) of mappings is called an *isomorphism* between G and H.

To show that two graphs are isomorphic, one must indicate an isomorphism between them. The pair of mappings (θ, ϕ) defined by

$$\theta(v_1) = y, \quad \theta(v_2) = x, \quad \theta(v_3) = u, \quad \theta(v_4) = v, \quad \theta(v_5) = w$$

and

$$\phi(e_1) = h, \quad \phi(e_2) = g, \quad \phi(e_3) = b, \quad \phi(e_4) = a$$
$$\phi(e_5) = e, \quad \phi(e_6) = c, \quad \phi(e_7) = d, \quad \phi(e_8) = f$$

is an isomorphism between the graphs G and H of examples 1 and 2; G and H clearly have the same structure, and differ only in the names of vertices and edges. Since it is in structural properties that we shall primarily be interested, we shall often omit labels when drawing graphs; an unlabelled graph can be thought of as a representative of an equivalence class of isomorphic graphs. We assign labels to vertices and edges in a graph mainly for the purpose of referring to them. For instance, when dealing with simple graphs, it is often convenient to refer to the edge with ends u and v as 'the edge uv'. (This convention results in no ambiguity since, in a simple graph, at most one edge joins any pair of vertices.)

We conclude this section by introducing some special classes of graphs. A simple graph in which each pair of distinct vertices is joined by an edge is called a *complete graph*. Up to isomorphism, there is just one complete graph on n vertices; it is denoted by K_n. A drawing of K_5 is shown in figure 1.4*a*. An *empty graph*, on the other hand, is one with no edges. A *bipartite*

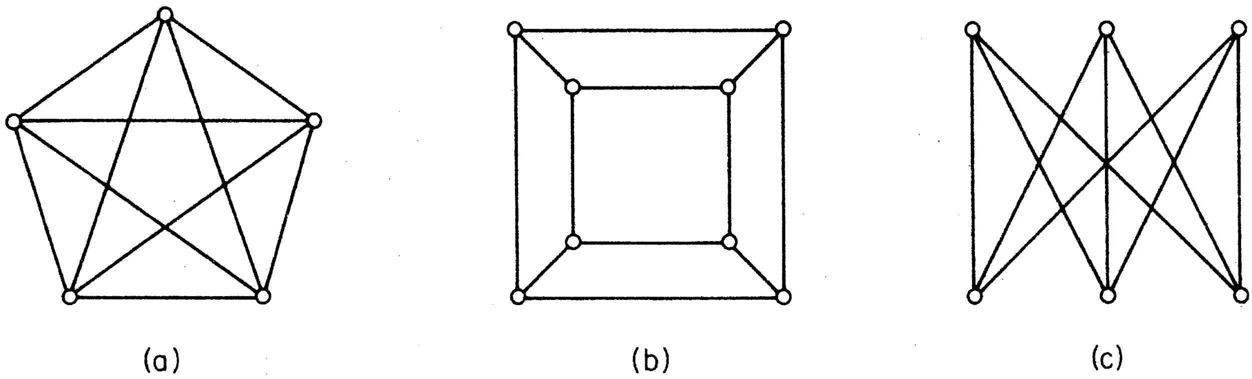


Figure 1.4. (*a*) K_5; (*b*) the cube; (*c*) $K_{3,3}$

graph is one whose vertex set can be partitioned into two subsets X and Y, so that each edge has one end in X and one end in Y; such a partition (X, Y) is called a *bipartition* of the graph. A *complete bipartite graph* is a simple bipartite graph with bipartition (X, Y) in which each vertex of X is joined to each vertex of Y; if $|X| = m$ and $|Y| = n$, such a graph is denoted by $K_{m,n}$. The graph defined by the vertices and edges of a cube (figure 1.4*b*) is bipartite; the graph in figure 1.4*c* is the complete bipartite graph $K_{3,3}$.

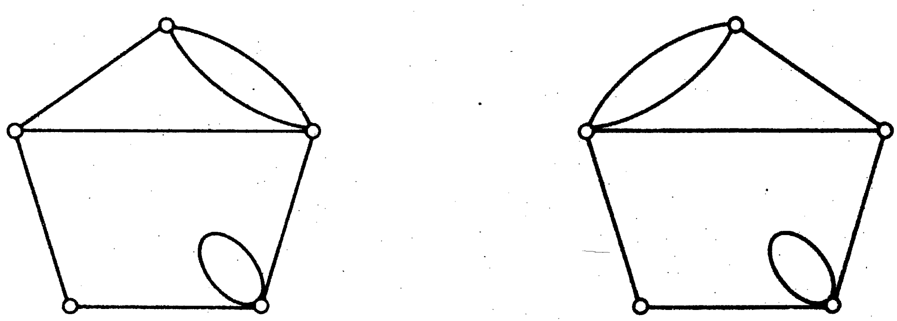
There are many other graphs whose structures are of special interest. Appendix III includes a selection of such graphs.

Exercises

1.2.1 Find an isomorphism between the graphs G and H of examples 1 and 2 different from the one given.

1.2.2 (*a*) Show that if $G \cong H$, then $\nu(G) = \nu(H)$ and $\varepsilon(G) = \varepsilon(H)$.
(*b*) Give an example to show that the converse is false.

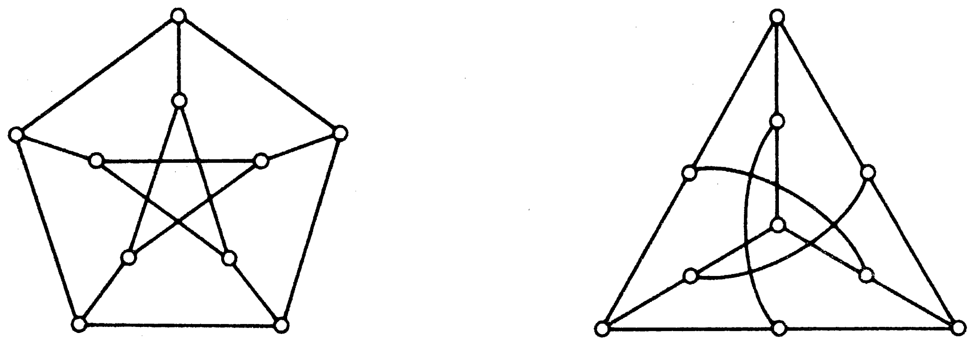
1.2.3 Show that the following graphs are not isomorphic:



1.2.4 Show that there are eleven nonisomorphic simple graphs on four vertices.

1.2.5 Show that two simple graphs G and H are isomorphic if and only if there is a bijection $\theta : V(G) \rightarrow V(H)$ such that $uv \in E(G)$ if and only if $\theta(u)\theta(v) \in E(H)$.

1.2.6 Show that the following graphs are isomorphic:



1.2.7 Let G be simple. Show that $\varepsilon = \binom{\nu}{2}$ if and only if G is complete.

1.2.8 Show that

(*a*) $\varepsilon(K_{m,n}) = mn$;

(*b*) if G is simple and bipartite, then $\varepsilon \leq \nu^2/4$.

1.2.9 A k-*partite graph* is one whose vertex set can be partitioned into k subsets so that no edge has both ends in any one subset; a *complete* k-*partite graph* is one that is simple and in which each vertex is joined to every vertex that is not in the same subset. The complete m-partite graph on n vertices in which each part has either $[n/m]$ or $\{n/m\}$ vertices is denoted by $T_{m,n}$. Show that

(*a*) $\varepsilon(T_{m,n}) = \binom{n-k}{2} + (m-1)\binom{k+1}{2}$, where $k = [n/m]$;

(*b*)* if G is a complete m-partite graph on n vertices, then $\varepsilon(G) \leq \varepsilon(T_{m,n})$, with equality only if $G \cong T_{m,n}$.

1.2.10 The k-*cube* is the graph whose vertices are the ordered k-tuples of 0's and 1's, two vertices being joined if and only if they differ in exactly one coordinate. (The graph shown in figure 1.4*b* is just the 3-cube.) Show that the k-cube has 2^k vertices, $k2^{k-1}$ edges and is bipartite.

1.2.11 (*a*) The *complement* G^c of a simple graph G is the simple graph with vertex set V, two vertices being adjacent in G^c if and only if they are not adjacent in G. Describe the graphs K_n^c and $K_{m,n}^c$.

(*b*) A simple graph G is *self-complementary* if $G \cong G^c$. Show that if G is self-complementary, then $\nu \equiv 0, 1 \pmod 4$.

1.2.12 An *automorphism* of a graph is an isomorphism of the graph onto itself.

(*a*) Show, using exercise 1.2.5, that an automorphism of a simple graph G can be regarded as a permutation on V which preserves adjacency, and that the set of such permutations form a group $\Gamma(G)$ (the *automorphism group* of G) under the usual operation of composition.

(*b*) Find $\Gamma(K_n)$ and $\Gamma(K_{m,n})$.

(*c*) Find a nontrivial simple graph whose automorphism group is the identity.

(*d*) Show that for any simple graph G, $\Gamma(G)=\Gamma(G^c)$.

(*e*) Consider the permutation group Λ with elements $(1)(2)(3)$, $(1,2,3)$ and $(1,3,2)$. Show that there is no simple graph G with vertex set $\{1,2,3\}$ such that $\Gamma(G)=\Lambda$.

(*f*) Find a simple graph G such that $\Gamma(G)\cong\Lambda$. (Frucht, 1939 has shown that every abstract group is isomorphic to the automorphism group of some graph.)

1.2.13 A simple graph G is *vertex-transitive* if, for any two vertices u and v, there is an element g in $\Gamma(G)$ such that $g(u)=g(v)$; G is *edge-transitive* if, for any two edges u_1v_1 and u_2v_2, there is an element h in $\Gamma(G)$ such that $h(\{u_1, v_1\})=\{u_2, v_2\}$. Find

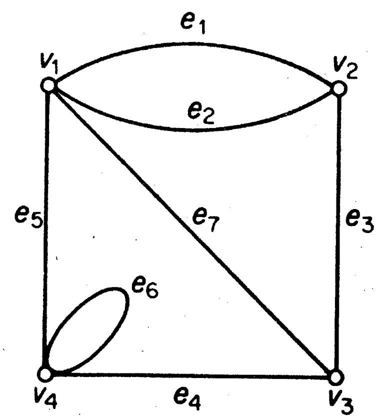
(*a*) a graph which is vertex-transitive but not edge-transitive;

(*b*) a graph which is edge-transitive but not vertex-transitive.

1.3 THE INCIDENCE AND ADJACENCY MATRICES

To any graph G there corresponds a $\nu\times\varepsilon$ matrix called the incidence matrix of G. Let us denote the vertices of G by $v_1, v_2, \ldots, v_\nu$ and the edges by $e_1, e_2, \ldots, e_\varepsilon$. Then the *incidence matrix* of G is the matrix $\mathbf{M}(G)=[m_{ij}]$, where m_{ij} is the number of times (0, 1 or 2) that v_i and e_j are incident. The incidence matrix of a graph is just a different way of specifying the graph.

Another matrix associated with G is the *adjacency matrix*; this is the $\nu\times\nu$ matrix $\mathbf{A}(G)=[a_{ij}]$, in which a_{ij} is the number of edges joining v_i and v_j. A graph, its incidence matrix, and its adjacency matrix are shown in figure 1.5.



G

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1	1	1	0	0	1	0	1
v_2	1	1	1	0	0	0	0
v_3	0	0	1	1	0	0	1
v_4	0	0	0	1	1	2	0

$\mathbf{M}(G)$

	v_1	v_2	v_3	v_4
v_1	0	2	1	1
v_2	2	0	1	0
v_3	1	1	0	1
v_4	1	0	1	1

$\mathbf{A}(G)$

Figure 1.5

The adjacency matrix of a graph is generally considerably smaller than its incidence matrix, and it is in this form that graphs are commonly stored in computers.

Exercises

1.3.1 Let **M** be the incidence matrix and **A** the adjacency matrix of a graph G.

(*a*) Show that every column sum of **M** is 2.

(*b*) What are the column sums of **A**?

1.3.2 Let G be bipartite. Show that the vertices of G can be enumerated so that the adjacency matrix of G has the form

$$\left[\begin{array}{c|c} \mathbf{0} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{0} \end{array}\right]$$

where $\mathbf{A}_{21}$ is the transpose of $\mathbf{A}_{12}$.

1.3.3* Show that if G is simple and the eigenvalues of **A** are distinct, then the automorphism group of G is abelian

1.4 SUBGRAPHS

A graph H is a *subgraph* of G (written $H \subseteq G$) if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, and ψ_H is the restriction of ψ_G to $E(H)$. When $H \subseteq G$ but $H \neq G$, we write $H \subset G$ and call H a *proper subgraph* of G. If H is a subgraph of G, G is a *supergraph* of H. A *spanning subgraph* (or *spanning supergraph*) of G is a subgraph (or supergraph) H with $V(H) = V(G)$.

By deleting from G all loops and, for every pair of adjacent vertices, all but one link joining them, we obtain a simple spanning subgraph of G, called the *underlying simple graph* of G. Figure 1.6 shows a graph and its underlying simple graph.

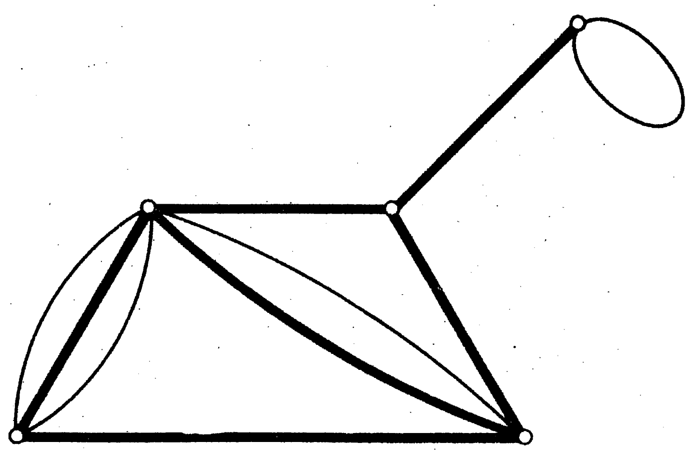


Figure 1.6. A graph and its underlying simple graph

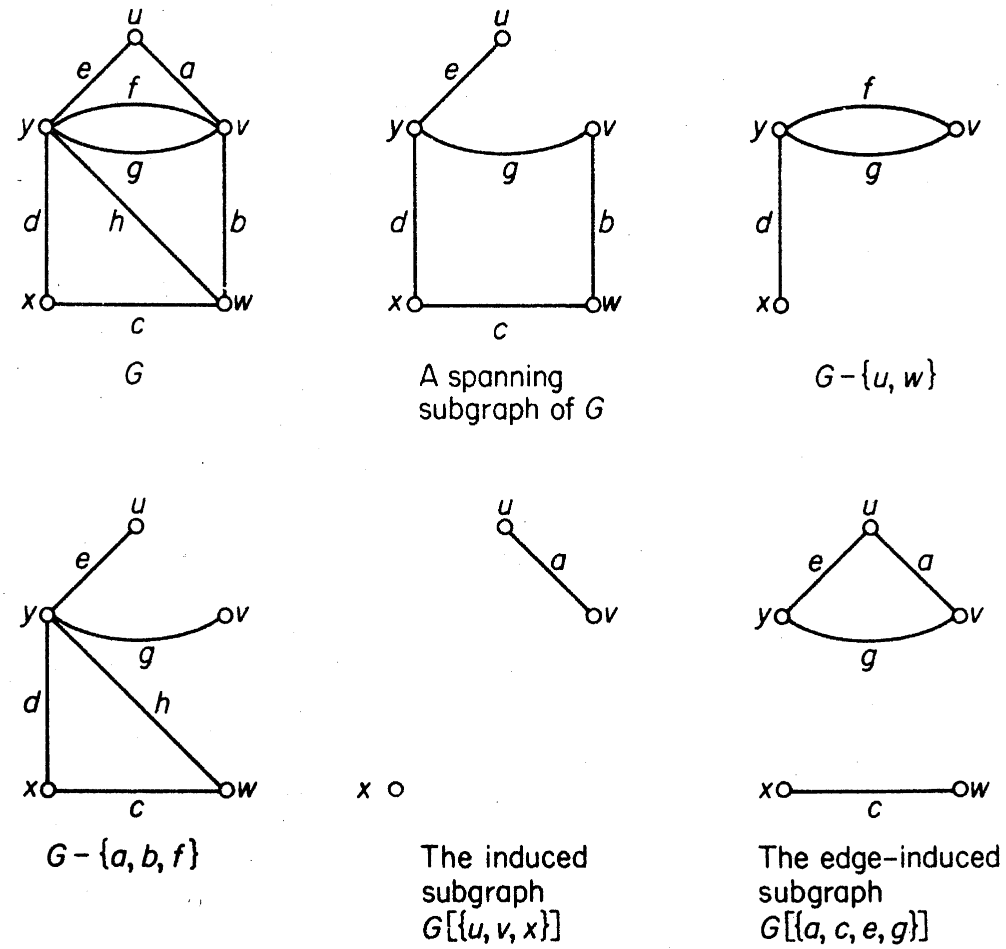


Figure 1.7

Suppose that V' is a nonempty subset of V. The subgraph of G whose vertex set is V' and whose edge set is the set of those edges of G that have both ends in V' is called the subgraph of G *induced* by V' and is denoted by $G[V']$; we say that $G[V']$ is an *induced subgraph* of G. The induced subgraph $G[V\backslash V']$ is denoted by $G-V'$; it is the subgraph obtained from G by deleting the vertices in V' together with their incident edges. If $V'=\{v\}$ we write $G-v$ for $G-\{v\}$.

Now suppose that E' is a nonempty subset of E. The subgraph of G whose vertex set is the set of ends of edges in E' and whose edge set is E' is called the subgraph of G *induced* by E' and is denoted by $G[E']$; $G[E']$ is an *edge-induced subgraph* of G. The spanning subgraph of G with edge set $E\backslash E'$ is written simply as $G-E'$; it is the subgraph obtained from G by deleting the edges in E'. Similarly, the graph obtained from G by adding a set of edges E' is denoted by $G+E'$. If $E'=\{e\}$ we write $G-e$ and $G+e$ instead of $G-\{e\}$ and $G+\{e\}$.

Subgraphs of these various types are depicted in figure 1.7.

Let G_1 and G_2 be subgraphs of G. We say that G_1 and G_2 are *disjoint* if they have no vertex in common, and *edge-disjoint* if they have no edge in common. The *union* $G_1 \cup G_2$ of G_1 and G_2 is the subgraph with vertex set

$V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$; if G_1 and G_2 are disjoint, we sometimes denote their union by $G_1 + G_2$. The *intersection* $G_1 \cap G_2$ of G_1 and G_2 is defined similarly, but in this case G_1 and G_2 must have at least one vertex in common.

Exercises

1.4.1 Show that every simple graph on n vertices is isomorphic to a subgraph of K_n.

1.4.2 Show that

(*a*) every induced subgraph of a complete graph is complete;
(*b*) every subgraph of a bipartite graph is bipartite.

1.4.3 Describe how $\mathbf{M}(G-E')$ and $\mathbf{M}(G-V')$ can be obtained from $\mathbf{M}(G)$, and how $\mathbf{A}(G-V')$ can be obtained from $\mathbf{A}(G)$.

1.4.4 Find a bipartite graph that is not isomorphic to a subgraph of any k-cube.

1.4.5* Let G be simple and let n be an integer with $1 < n < \nu - 1$. Show that if $\nu \geq 4$ and all induced subgraphs of G on n vertices have the same number of edges, then either $G \cong K_\nu$ or $G \cong K_\nu^c$.

1.5 VERTEX DEGREES

The *degree* $d_G(v)$ of a vertex v in G is the number of edges of G incident with v, each loop counting as two edges. We denote by $\delta(G)$ and $\Delta(G)$ the minimum and maximum degrees, respectively, of vertices of G.

Theorem 1.1

$$\sum_{v \in V} d(v) = 2\varepsilon$$

Proof Consider the incidence matrix $\mathbf{M}$. The sum of the entries in the row corresponding to vertex v is precisely $d(v)$, and therefore $\sum_{v \in V} d(v)$ is just the sum of all entries in $\mathbf{M}$. But this sum is also 2ε, since (exercise 1.3.1*a*) each of the ε column sums of $\mathbf{M}$ is 2 □

Corollary 1.1 In any graph, the number of vertices of odd degree is even.

Proof Let V_1 and V_2 be the sets of vertices of odd and even degree in G, respectively. Then

$$\sum_{v \in V_1} d(v) + \sum_{v \in V_2} d(v) = \sum_{v \in V} d(v)$$

is even, by theorem 1.1. Since $\sum_{v \in V_2} d(v)$ is also even, it follows that $\sum_{v \in V_1} d(v)$ is even. Thus $|V_1|$ is even □

A graph G is *k-regular* if $d(v)=k$ for all $v \in V$; a *regular graph* is one that is k-regular for some k. Complete graphs and complete bipartite graphs $K_{n,n}$ are regular; so, also, are the k-cubes.

Exercises

1.5.1 Show that $\delta \leq 2\varepsilon/\nu \leq \Delta$.

1.5.2 Show that if G is simple, the entries on the diagonals of both $\mathbf{MM}'$ and $\mathbf{A}^2$ are the degrees of the vertices of G.

1.5.3 Show that if a k-regular bipartite graph with $k>0$ has bipartition (X, Y), then $|X|=|Y|$.

1.5.4 Show that, in any group of two or more people, there are always two with exactly the same number of friends inside the group.

1.5.5 If G has vertices $v_1, v_2, \ldots, v_\nu$, the sequence $(d(v_1), d(v_2), \ldots, d(v_\nu))$ is called a *degree sequence* of G. Show that a sequence $(d_1, d_2, \ldots, d_n)$ of non-negative integers is a degree sequence of some graph if and only if $\sum_{i=1}^{n} d_i$ is even.

1.5.6 A sequence $\mathbf{d}=(d_1, d_2, \ldots, d_n)$ is *graphic* if there is a simple graph with degree sequence $\mathbf{d}$. Show that

(*a*) the sequences (7, 6, 5, 4, 3, 3, 2) and (6, 6, 5, 4, 3, 3, 1) are not graphic;

(*b*) if $\mathbf{d}$ is graphic and $d_1 \geq d_2 \geq \ldots \geq d_n$, then $\sum_{i=1}^{n} d_i$ is even and

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min\{k, d_i\} \quad \text{for} \quad 1 \leq k \leq n$$

(Erdös and Gallai, 1960 have shown that this necessary condition is also sufficient for $\mathbf{d}$ to be graphic.)

1.5.7 Let $\mathbf{d}=(d_1, d_2, \ldots, d_n)$ be a nonincreasing sequence of non-negative integers, and denote the sequence $(d_2-1, d_3-1, \ldots, d_{d_1+1}-1, d_{d_1+2}, \ldots, d_n)$ by $\mathbf{d}'$.

(*a*)* Show that $\mathbf{d}$ is graphic if and only if $\mathbf{d}'$ is graphic.

(*b*) Using (*a*), describe an algorithm for constructing a simple graph with degree sequence $\mathbf{d}$, if such a graph exists.

(V. Havel, S. Hakimi)

1.5.8* Show that a loopless graph G contains a bipartite spanning subgraph H such that $d_H(v) \geq \frac{1}{2} d_G(v)$ for all $v \in V$.

1.5.9* Let $S=\{x_1, x_2, \ldots, x_n\}$ be a set of points in the plane such that the distance between any two points is at least one. Show that there are at most $3n$ pairs of points at distance exactly one.

1.5.10 The *edge graph* of a graph G is the graph with vertex set $E(G)$ in which two vertices are joined if and only if they are adjacent edges in

G. Show that, if G is simple

(a) the edge graph of G has $\varepsilon(G)$ vertices and $\sum_{v\in V(G)}\binom{d_G(v)}{2}$ edges;

(b) the edge graph of K_5 is isomorphic to the complement of the graph featured in exercise 1.2.6.

1.6 PATHS AND CONNECTION

A *walk* in G is a finite non-null sequence $W = v_0e_1v_1e_2v_2 \dots e_kv_k$, whose terms are alternately vertices and edges, such that, for $1 \le i \le k$, the ends of e_i are v_{i-1} and v_i. We say that W is a walk *from* v_0 *to* v_k, or a (v_0, v_k)*-walk*. The vertices v_0 and v_k are called the *origin* and *terminus* of W, respectively, and $v_1, v_2, \dots, v_{k-1}$ its *internal vertices*. The integer k is the *length* of W.

If $W = v_0e_1v_1 \dots e_kv_k$ and $W' = v_ke_{k+1}v_{k+1} \dots e_lv_l$ are walks, the walk $v_ke_kv_{k-1} \dots e_1v_0$, obtained by reversing W, is denoted by W^{-1} and the walk $v_0e_1v_1 \dots e_lv_l$, obtained by concatenating W and W′ at v_k, is denoted by WW′. A *section* of a walk $W = v_0e_1v_1 \dots e_kv_k$ is a walk that is a subsequence $v_ie_{i+1}v_{i+1} \dots e_jv_j$ of consecutive terms of W; we refer to this subsequence as the (v_i, v_j)*-section* of W.

In a simple graph, a walk $v_0e_1v_1 \dots e_kv_k$ is determined by the sequence $v_0v_1 \dots v_k$ of its vertices; hence a walk in a simple graph can be specified simply by its vertex sequence. Moreover, even in graphs that are not simple, we shall sometimes refer to a sequence of vertices in which consecutive terms are adjacent as a 'walk'. In such cases it should be understood that the discussion is valid for every walk with that vertex sequence.

If the edges $e_1, e_2, \dots, e_k$ of a walk W are distinct, W is called a *trail*; in this case the length of W is just $\varepsilon(W)$. If, in addition, the vertices $v_0, v_1, \dots, v_k$ are distinct, W is called a *path*. Figure 1.8 illustrates a walk, a trail and a path in a graph. We shall also use the word 'path' to denote a graph or subgraph whose vertices and edges are the terms of a path.

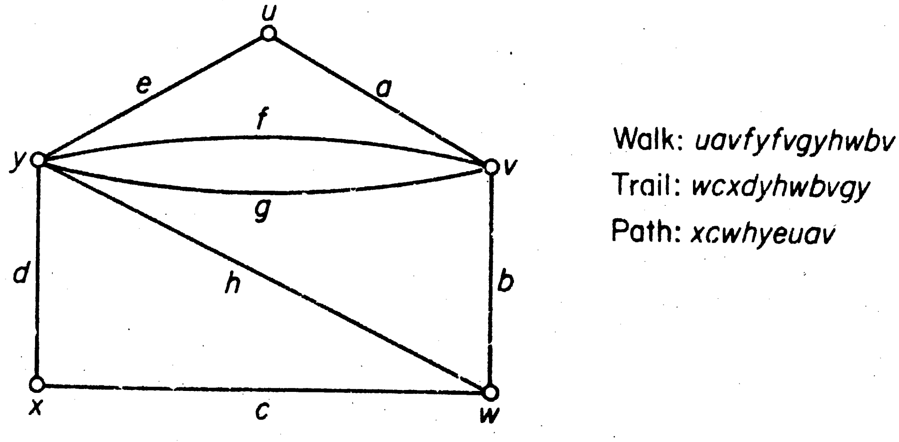


Figure 1.8

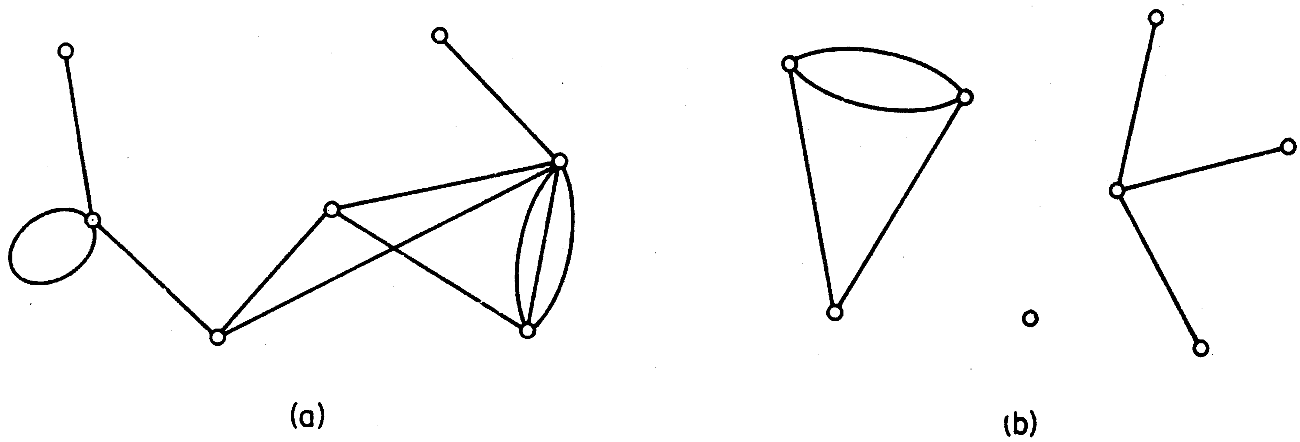


Figure 1.9. (*a*) A connected graph; (*b*) a disconnected graph with three components

Two vertices u and v of G are said to be *connected* if there is a (u, v)-path in G. Connection is an equivalence relation on the vertex set V. Thus there is a partition of V into nonempty subsets $V_1, V_2, \ldots, V_\omega$ such that two vertices u and v are connected if and only if both u and v belong to the same set V_i. The subgraphs $G[V_1], G[V_2], \ldots, G[V_\omega]$ are called the *components* of G. If G has exactly one component, G is *connected;* otherwise G is *disconnected.* We denote the number of components of G by $\omega(G)$. Connected and disconnected graphs are depicted in figure 1.9.

Exercises

1.6.1 Show that if there is a (u, v)-walk in G, then there is also a (u, v)-path in G.

1.6.2 Show that the number of (v_i, v_j)-walks of length k in G is the (i, j)th entry of $\mathbf{A}^k$.

1.6.3 Show that if G is simple and $\delta \geq k$, then G has a path of length k.

1.6.4 Show that G is connected if and only if, for every partition of V into two nonempty sets V_1 and V_2, there is an edge with one end in V_1 and one end in V_2.

1.6.5 (*a*) Show that if G is simple and $\varepsilon > \binom{\nu - 1}{2}$, then G is connected.
(*b*) For $\nu > 1$, find a disconnected simple graph G with $\varepsilon = \binom{\nu - 1}{2}$.

1.6.6 (*a*) Show that if G is simple and $\delta > [\nu/2] - 1$, then G is connected.
(*b*) Find a disconnected $([\nu/2] - 1)$-regular simple graph for ν even.

1.6.7 Show that if G is disconnected, then G^c is connected.

1.6.8 (*a*) Show that if $e \in E$, then $\omega(G) \leq \omega(G - e) \leq \omega(G) + 1$.
(*b*) Let $v \in V$. Show that $G - e$ cannot, in general, be replaced by $G - v$ in the above inequality.

1.6.9 Show that if G is connected and each degree in G is even, then, for any $v \in V$, $\omega(G - v) \leq \frac{1}{2}d(v)$.

1.6.10 Show that any two longest paths in a connected graph have a vertex in common.

1.6.11 If vertices u and v are connected in G, the *distance* between u and v in G, denoted by $d_G(u, v)$, is the length of a shortest (u, v)-path in G; if there is no path connecting u and v we define $d_G(u, v)$ to be infinite. Show that, for any three vertices u, v and w, $d(u, v)+d(v, w) \geq d(u, w)$.

1.6.12 The *diameter* of G is the maximum distance between two vertices of G. Show that if G has diameter greater than three, then G^c has diameter less than three.

1.6.13 Show that if G is simple with diameter two and $\Delta = \nu - 2$, then $\varepsilon \geq 2\nu - 4$.

1.6.14 Show that if G is simple and connected but not complete, then G has three vertices u, v and w such that $uv, vw \in E$ and $uw \notin E$.

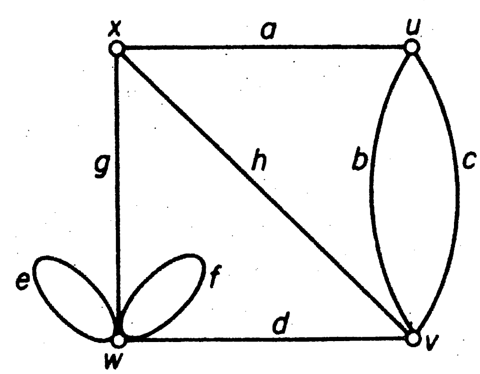
1.7 CYCLES

A walk is *closed* if it has positive length and its origin and terminus are the same. A closed trail whose origin and internal vertices are distinct is a *cycle*. Just as with paths we sometimes use the term 'cycle' to denote a graph corresponding to a cycle. A cycle of length k is called a *k-cycle*; a k-cycle is *odd* or *even* according as k is odd or even. A 3-cycle is often called a *triangle*. Examples of a closed trail and a cycle are given in figure 1.10.

Using the concept of a cycle, we can now present a characterisation of bipartite graphs.

Theorem 1.2 A graph is bipartite if and only if it contains no odd cycle.

Proof Suppose that G is bipartite with bipartition (X, Y), and let $C = v_0v_1 \ldots v_kv_0$ be a cycle of G. Without loss of generality we may assume that $v_0 \in X$. Then, since $v_0v_1 \in E$ and G is bipartite, $v_1 \in Y$. Similarly $v_2 \in X$ and, in general, $v_{2i} \in X$ and $v_{2i+1} \in Y$. Since $v_0 \in X$, $v_k \in Y$. Thus $k = 2i+1$, for some i, and it follows that C is even.



Closed trail: *ucvhxgwfwdvbu*
Cycle: *xaubvhx*

Figure 1.10

It clearly suffices to prove the converse for connected graphs. Let G be a connected graph that contains no odd cycles. We choose an arbitrary vertex u and define a partition (X, Y) of V by setting

$$X = \{x \in V \mid d(u, x) \quad \text{is even}\}$$
$$Y = \{y \in V \mid d(u, y) \quad \text{is odd}\}$$

We shall show that (X, Y) is a bipartition of G. Suppose that v and w are two vertices of X. Let P be a shortest (u, v)-path and Q be a shortest (u, w)-path. Denote by u_1 the last vertex common to P and Q. Since P and Q are shortest paths, the (u, u_1)-sections of both P and Q are shortest (u, u_1)-paths and, therefore, have the same length. Now, since the lengths of both P and Q are even, the lengths of the (u_1, v)-section P_1 of P and the (u_1, w)-section Q_1 of Q must have the same parity. It follows that the (v, w)-path $P_1^{-1}Q_1$ is of even length. If v were joined to w, $P_1^{-1}Q_1wv$ would be a cycle of odd length, contrary to the hypothesis. Therefore no two vertices in X are adjacent; similarly, no two vertices in Y are adjacent □

Exercises

1.7.1 Show that if an edge e is in a closed trail of G, then e is in a cycle of G.

1.7.2 Show that if $\delta \geq 2$, then G contains a cycle.

1.7.3* Show that if G is simple and $\delta \geq 2$, then G contains a cycle of length at least $\delta + 1$.

1.7.4 The *girth* of G is the length of a shortest cycle in G; if G has no cycles we define the girth of G to be infinite. Show that

(*a*) a k-regular graph of girth four has at least $2k$ vertices, and (up to isomorphism) there exists exactly one such graph on $2k$ vertices;

(*b*) a k-regular graph of girth five has at least $k^2 + 1$ vertices.

1.7.5 Show that a k-regular graph of girth five and diameter two has exactly $k^2 + 1$ vertices, and find such a graph for $k = 2, 3$. (Hoffman and Singleton, 1960 have shown that such a graph can exist only if $k = 2, 3, 7$ and, possibly, 57.)

1.7.6 Show that

(*a*) if $\varepsilon \geq \nu$, G contains a cycle;

(*b*)* if $\varepsilon \geq \nu + 4$, G contains two edge-disjoint cycles. (L. Pósa)

APPLICATIONS

1.8 THE SHORTEST PATH PROBLEM

With each edge e of G let there be associated a real number $w(e)$, called its *weight*. Then G, together with these weights on its edges, is called a *weighted*

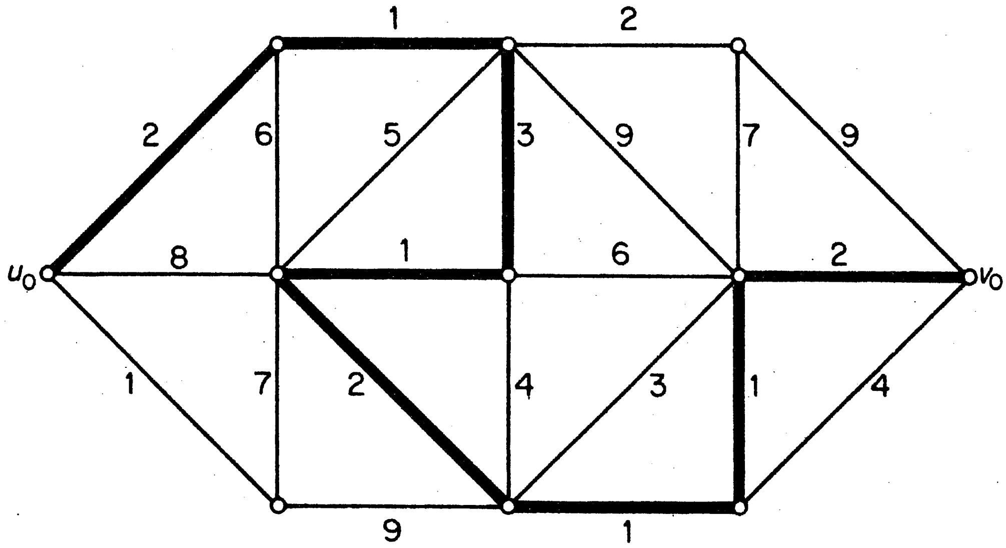


Figure 1.11. A (u_0, v_0)-path of minimum weight

graph. Weighted graphs occur frequently in applications of graph theory. In the friendship graph, for example, weights might indicate intensity of friendship; in the communications graph, they could represent the construction or maintenance costs of the various communication links.

If H is a subgraph of a weighted graph, the *weight* $w(H)$ of H is the sum of the weights $\sum_{e \in E(H)} w(e)$ on its edges. Many optimisation problems amount to finding, in a weighted graph, a subgraph of a certain type with minimum (or maximum) weight. One such is the *shortest path problem:* given a railway network connecting various towns, determine a shortest route between two specified towns in the network.

Here one must find, in a weighted graph, a path of minimum weight connecting two specified vertices u_0 and v_0; the weights represent distances by rail between directly-linked towns, and are therefore non-negative. The path indicated in the graph of figure 1.11 is a (u_0, v_0)-path of minimum weight (exercise 1.8.1).

We now present an algorithm for solving the shortest path problem. For clarity of exposition, we shall refer to the weight of a path in a weighted graph as its *length;* similarly the minimum weight of a (u, v)-path will be called the *distance* between u and v and denoted by $d(u, v)$. These definitions coincide with the usual notions of length and distance, as defined in section 1.6, when all the weights are equal to one.

It clearly suffices to deal with the shortest path problem for simple graphs; so we shall assume here that G is simple. We shall also assume that all the weights are positive. This, again, is not a serious restriction because, if the weight of an edge is zero, then its ends can be identified. We adopt the convention that $w(uv) = \infty$ if $uv \notin E$.

The algorithm to be described was discovered by Dijkstra (1959) and, independently, by Whiting and Hillier (1960). It finds not only a shortest (u_0, v_0)-path, but shortest paths from u_0 to all other vertices of G. The basic idea is as follows.

Suppose that S is a proper subset of V such that $u_0 \in S$, and let $\bar{S}$ denote $V \backslash S$. If $P = u_0 \ldots \bar{u}\bar{v}$ is a shortest path from u_0 to $\bar{S}$ then clearly $\bar{u} \in S$ and the $(u_0, \bar{u})$-section of P must be a shortest $(u_0, \bar{u})$-path. Therefore

$$d(u_0, \bar{v}) = d(u_0, \bar{u}) + w(\bar{u}\bar{v})$$

and the distance from u_0 to $\bar{S}$ is given by the formula

$$d(u_0, \bar{S}) = \min_{\substack{u \in S \\ v \in \bar{S}}} \{d(u_0, u) + w(uv)\} \tag{1.1}$$

This formula is the basis of Dijkstra's algorithm. Starting with the set $S_0 = \{u_0\}$, an increasing sequence $S_0, S_1, \ldots, S_{\nu-1}$ of subsets of V is constructed, in such a way that, at the end of stage i, shortest paths from u_0 to all vertices in S_i are known.

The first step is to determine a vertex nearest to u_0. This is achieved by computing $d(u_0, \bar{S}_0)$ and selecting a vertex $u_1 \in \bar{S}_0$ such that $d(u_0, u_1) = d(u_0, \bar{S}_0)$; by (1.1)

$$d(u_0, \bar{S}_0) = \min_{\substack{u \in S_0 \\ v \in \bar{S}_0}} \{d(u_0, u) + w(uv)\} = \min_{v \in \bar{S}_0} \{w(u_0 v)\}$$

and so $d(u_0, \bar{S}_0)$ is easily computed. We now set $S_1 = \{u_0, u_1\}$ and let P_1 denote the path u_0u_1; this is clearly a shortest (u_0, u_1)-path. In general, if the set $S_k = \{u_0, u_1, \ldots, u_k\}$ and corresponding shortest paths $P_1, P_2, \ldots, P_k$ have already been determined, we compute $d(u_0, \bar{S}_k)$ using (1.1) and select a vertex $u_{k+1} \in \bar{S}_k$ such that $d(u_0, u_{k+1}) = d(u_0, \bar{S}_k)$. By (1.1), $d(u_0, u_{k+1}) = d(u_0, u_j) + w(u_ju_{k+1})$ for some $j \leq k$; we get a shortest (u_0, u_{k+1})-path by adjoining the edge u_ju_{k+1} to the path P_j.

We illustrate this procedure by considering the weighted graph depicted in figure 1.12*a*. Shortest paths from u_0 to the remaining vertices are determined in seven stages. At each stage, the vertices to which shortest paths have been found are indicated by solid dots, and each is labelled by its distance from u_0; initially u_0 is labelled 0. The actual shortest paths are indicated by solid lines. Notice that, at each stage, these shortest paths together form a connected graph without cycles; such a graph is called a *tree*, and we can think of the algorithm as a 'tree-growing' procedure. The final tree, in figure 1.12*h*, has the property that, for each vertex v, the path connecting u_0 and v is a shortest (u_0, v)-path.

Dijkstra's algorithm is a refinement of the above procedure. This refinement is motivated by the consideration that, if the minimum in (1.1) were to be computed from scratch at each stage, many comparisons would be

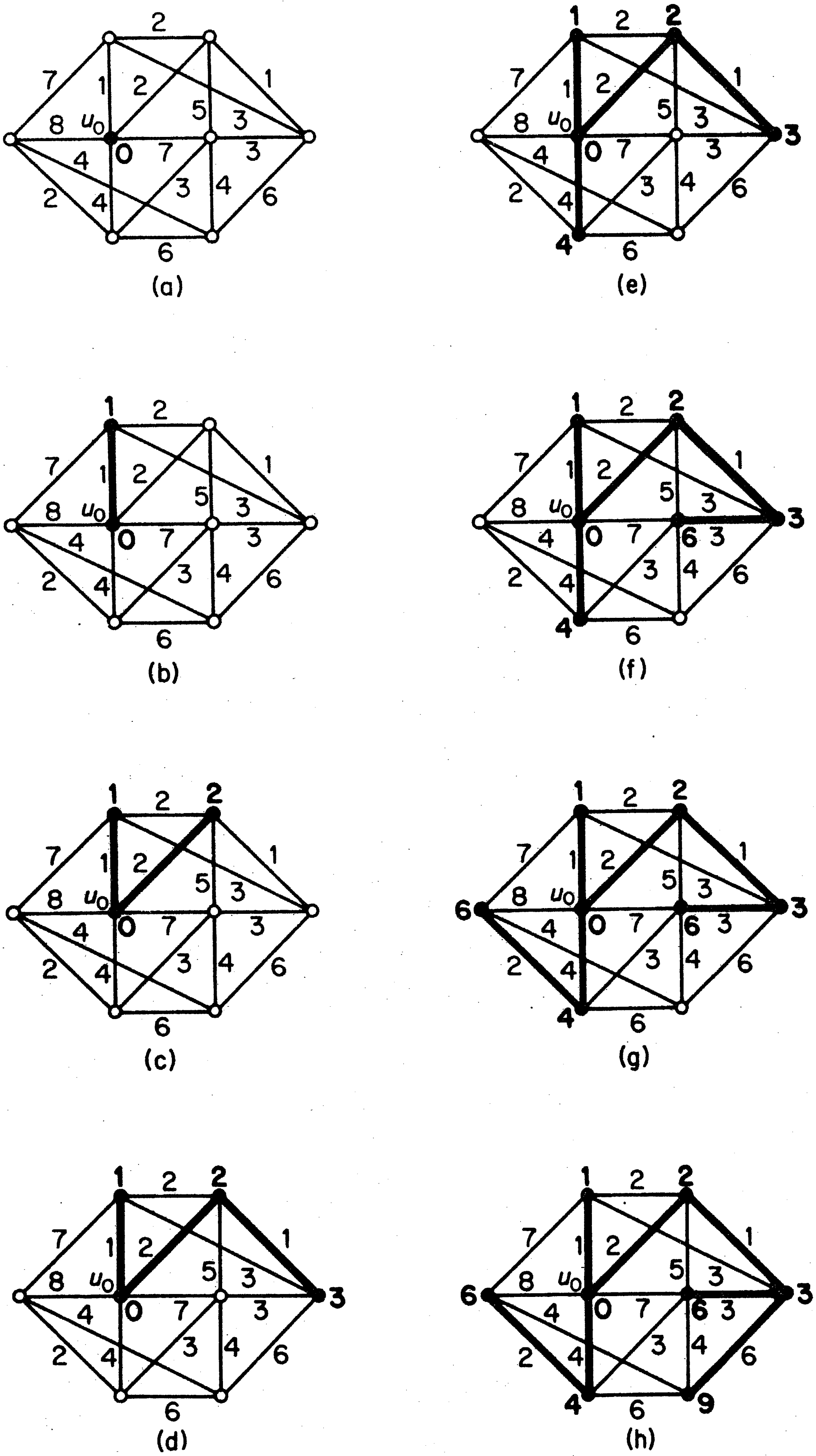


Figure 1.12. Shortest path algorithm

repeated unnecessarily. To avoid such repetitions, and to retain computational information from one stage to the next, we adopt the following labelling procedure. Throughout the algorithm, each vertex v carries a label $l(v)$ which is an upper bound on $d(u_0, v)$. Initially $l(u_0)=0$ and $l(v)=\infty$ for $v \neq u_0$. (In actual computations ∞ is replaced by any sufficiently large number.) As the algorithm proceeds, these labels are modified so that, at the end of stage i,

$$l(u) = d(u_0, u) \quad \text{for} \quad u \in S_i$$

and

$$l(v) = \min_{u \in S_{i-1}}\{d(u_0, u) + w(uv)\} \quad \text{for} \quad v \in \bar{S}_i$$

Dijkstra's Algorithm

1. Set $l(u_0)=0$, $l(v)=\infty$ for $v \neq u_0$, $S_0=\{u_0\}$ and $i=0$.
2. For each $v \in \bar{S}_i$, replace $l(v)$ by $\min\{l(v), l(u_i)+w(u_iv)\}$. Compute $\min_{v \in \bar{S}_i}\{l(v)\}$ and let u_{i+1} denote a vertex for which this minimum is attained. Set $S_{i+1} = S_i \cup \{u_{i+1}\}$.
3. If $i = \nu - 1$, stop. If $i < \nu - 1$, replace i by $i+1$ and go to step 2.

When the algorithm terminates, the distance from u_0 to v is given by the final value of the label $l(v)$. (If our interest is in determining the distance to one specific vertex v_0, we stop as soon as some u_j equals v_0.) A flow diagram summarising this algorithm is shown in figure 1.13.

As described above, Dijkstra's algorithm determines only the distances from u_0 to all the other vertices, and not the actual shortest paths. These shortest paths can, however, be easily determined by keeping track of the predecessors of vertices in the tree (exercise 1.8.2).

Dijkstra's algorithm is an example of what Edmonds (1965) calls a good algorithm. A graph-theoretic algorithm is *good* if the number of computational steps required for its implementation on any graph G is bounded above by a polynomial in ν and ε (such as $3\nu^2\varepsilon$). An algorithm whose implementation may require an exponential number of steps (such as 2^ν) might be very inefficient for some large graphs.

To see that Dijkstra's algorithm is good, note that the computations involved in boxes 2 and 3 of the flow diagram, totalled over all iterations, require $\nu(\nu-1)/2$ additions and $\nu(\nu-1)$ comparisons. One of the questions that is not elaborated upon in the flow diagram is the matter of deciding whether a vertex belongs to $\bar{S}$ or not (box 1). Dreyfus (1969) reports a technique for doing this that requires a total of $(\nu-1)^2$ comparisons. Hence, if we regard either a comparison or an addition as a basic computational unit, the total number of computations required for this algorithm is approximately $5\nu^2/2$, and thus of order ν^2. (A function $f(\nu, \varepsilon)$ is of *order*

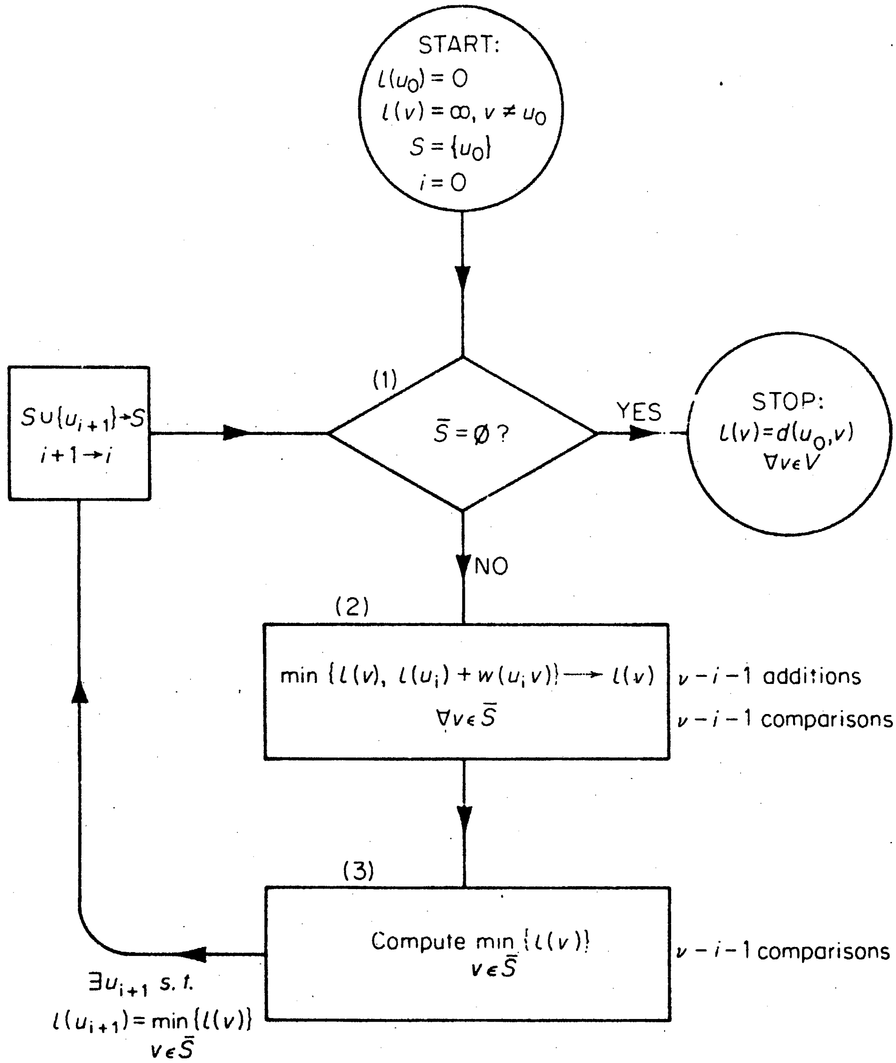


Figure 1.13. Dijkstra's algorithm

$g(\nu, \varepsilon)$ if there exists a positive constant c such that $f(\nu, \varepsilon)/g(\nu, \varepsilon) \leq c$ for all ν and ε.)

Although the shortest path problem can be solved by a good algorithm, there are many problems in graph theory for which no good algorithm is known. We refer the reader to Aho, Hopcroft and Ullman (1974) for further details.

Exercises

1.8.1 Find shortest paths from u_0 to all other vertices in the weighted graph of figure 1.11.

1.8.2 What additional instructions are needed in order that Dijkstra's algorithm determine shortest paths rather than merely distances?

1.8.3 A company has branches in each of six cities $C_1, C_2, \ldots, C_6$. The fare for a direct flight from C_i to C_j is given by the (i, j)th entry in the following matrix (∞ indicates that there is no direct flight):

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

The company is interested in computing a table of cheapest routes between pairs of cities. Prepare such a table.

1.8.4 A wolf, a goat and a cabbage are on one bank of a river. A ferryman wants to take them across, but, since his boat is small, he can take only one of them at a time. For obvious reasons, neither the wolf and the goat nor the goat and the cabbage can be left unguarded. How is the ferryman going to get them across the river?

1.8.5 Two men have a full eight-gallon jug of wine, and also two empty jugs of five and three gallons capacity, respectively. What is the simplest way for them to divide the wine equally?

1.8.6 Describe a good algorithm for determining

(a) the components of a graph;

(b) the girth of a graph.

How good are your algorithms?

1.9 SPERNER'S LEMMA

Every continuous mapping f of a closed n-disc to itself has a fixed point (that is, a point x such that $f(x)=x$). This powerful theorem, known as *Brouwer's fixed-point theorem*, has a wide range of applications in modern mathematics. Somewhat surprisingly, it is an easy consequence of a simple combinatorial lemma due to Sperner (1928). And, as we shall see in this section, Sperner's lemma is, in turn, an immediate consequence of corollary 1.1.

Sperner's lemma concerns the decomposition of a simplex (line segment, triangle, tetrahedron and so on) into smaller simplices. For the sake of simplicity we shall deal with the two-dimensional case.

Let T be a closed triangle in the plane. A subdivision of T into a finite number of smaller triangles is said to be *simplicial* if any two intersecting triangles have either a vertex or a whole side in common (see figure 1.14a).

Suppose that a simplicial subdivision of T is given. Then a labelling of the vertices of triangles in the subdivision in three symbols 0, 1 and 2 is said to be *proper* if

(i) the three vertices of T are labelled 0, 1 and 2 (in any order), and

(ii) for $0 \leq i < j \leq 2$, each vertex on the side of T joining vertices labelled i and j is labelled either i or j.

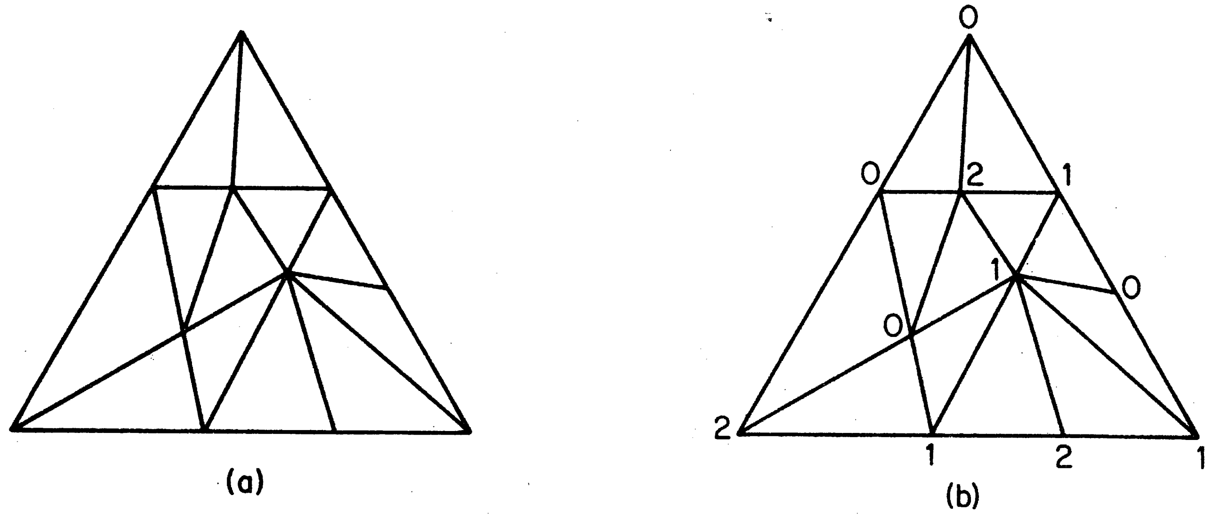


Figure 1.14. (*a*) A simplicial subdivision of a triangle; (*b*) a proper labelling of the subdivision

We call a triangle in the subdivision whose vertices receive all three labels a *distinguished* triangle. The proper labelling in figure 1.14*b* has three distinguished triangles.

Theorem 1.3 (*Sperner's lemma*) Every properly labelled simplicial subdivision of a triangle has an odd number of distinguished triangles.

Proof Let T_0 denote the region outside T, and let $T_1, T_2, \ldots, T_n$ be the triangles of the subdivision. Construct a graph on the vertex set $\{v_0, v_1, \ldots, v_n\}$ by joining v_i and v_j whenever the common boundary of T_i and T_j is an edge with labels 0 and 1 (see figure 1.15).

In this graph, v_0 is clearly of odd degree (exercise 1.9.1). It follows from corollary 1.1 that an odd number of the vertices $v_1, v_2, \ldots, v_n$ are of odd degree. Now it is easily seen that none of these vertices can have degree

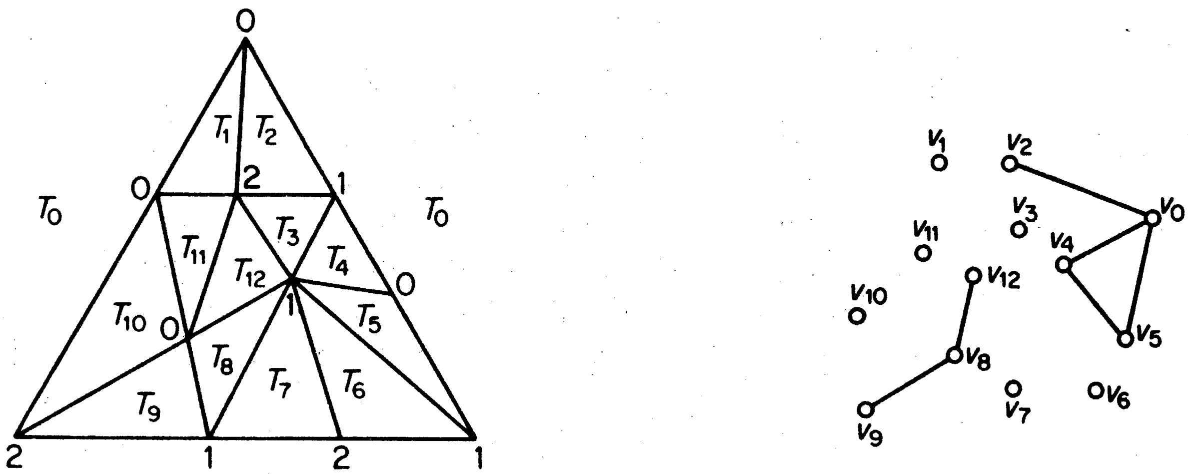


Figure 1.15

three, and so those with odd degree must have degree one. But a vertex v_i is of degree one if and only if the triangle T_i is distinguished □

We shall now briefly indicate how Sperner's lemma can be used to deduce Brouwer's fixed-point theorem. Again, for simplicity, we shall only deal with the two-dimensional case. Since a closed 2-disc is homeomorphic to a closed triangle, it suffices to prove that a continuous mapping of a closed triangle to itself has a fixed point.

Let T be a given closed triangle with vertices x_0, x_1 and x_2. Then each point x of T can be written uniquely as $x = a_0x_0 + a_1x_1 + a_2x_2$, where each $a_i \geq 0$ and $\Sigma\, a_i = 1$, and we can represent x by the vector (a_0, a_1, a_2); the real numbers a_0, a_1 and a_2 are called the *barycentric coordinates* of x.

Now let f be any continuous mapping of T to itself, and suppose that

$$f(a_0, a_1, a_2) = (a_0', a_1', a_2')$$

Define S_i as the set of points (a_0, a_1, a_2) in T for which $a_i' \leq a_i$. To show that f has a fixed point, it is enough to show that $S_0 \cap S_1 \cap S_2 \neq \emptyset$. For suppose that $(a_0, a_1, a_2) \in S_0 \cap S_1 \cap S_2$. Then, by the definition of S_i, we have that $a_i' \leq a_i$ for each i, and this, coupled with the fact that $\Sigma\, a_i' = \Sigma\, a_i$, yields

$$(a_0', a_1', a_2') = (a_0, a_1, a_2)$$

In other words, (a_0, a_1, a_2) is a fixed point of f.

So consider an arbitrary subdivision of T and a proper labelling such that each vertex labelled i belongs to S_i; the existence of such a labelling is easily seen (exercise 1.9.2*a*). It follows from Sperner's lemma that there is a triangle in the subdivision whose three vertices belong to S_0, S_1 and S_2. Now this holds for any subdivision of T and, since it is possible to choose subdivisions in which each of the smaller triangles are of arbitrarily small diameter, we conclude that there exist three points of S_0, S_1 and S_2 which are arbitrarily close to one another. Because the sets S_i are closed (exercise 1.9.2*b*), one may deduce that $S_0 \cap S_1 \cap S_2 \neq \emptyset$.

For details of the above proof and other applications of Sperner's lemma, the reader is referred to Tompkins (1964).

Exercises

1.9.1 In the proof of Sperner's lemma, show that the vertex v_0 is of odd degree.

1.9.2 In the proof of Brouwer's fixed-point theorem, show that

(*a*) there exists a proper labelling such that each vertex labelled i belongs to S_i;

(*b*) the sets S_i are closed.

1.9.3 State and prove Sperner's lemma for higher dimensional simplices.

REFERENCES

Aho, A. V., Hopcroft, J. E. and Ullman, J. D. (1974). *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer. Math.*, **1,** 269–71

Dreyfus, S. E. (1969). An appraisal of some shortest-path algorithms. *Operations Res.*, **17,** 395–412

Edmonds, J. (1965). Paths, trees and flowers. *Canad. J. Math.*, **17,** 449–67

Erdös, P. and Gallai, T. (1960). Graphs with prescribed degrees of vertices (Hungarian). *Mat. Lapok*, **11,** 264–74

Frucht, R. (1939). Herstellung von Graphen mit vorgegebener abstrakter Gruppe. *Compositio Math.*, **6,** 239–50

Hoffman, A. J. and Singleton, R. R. (1960). On Moore graphs with diameters 2 and 3. *IBM J. Res. Develop.*, **4,** 497–504

Sperner, E. (1928). Neuer Beweis für die Invarianz der Dimensionszahl und des Gebietes. *Hamburger Abhand.*, **6,** 265–72

Tompkins, C. B. (1964). Sperner's lemma and some extensions, in *Applied Combinatorial Mathematics*, ch. 15 (ed. E. F. Beckenbach), Wiley, New York, pp. 416–55

Whiting, P. D. and Hillier, J. A. (1960). A method for finding the shortest route through a road network. *Operational Res. Quart.*, **11,** 37–40