2 Trees

2.1 TREES

An *acyclic* graph is one that contains no cycles. A *tree* is a connected acyclic graph. The trees on six vertices are shown in figure 2.1.

Theorem 2.1 In a tree, any two vertices are connected by a unique path.

Proof By contradiction. Let G be a tree, and assume that there are two distinct (u, v)-paths P_1 and P_2 in G. Since $P_1 \neq P_2$, there is an edge $e = xy$ of P_1 that is not an edge of P_2. Clearly the graph $(P_1 \cup P_2) - e$ is connected. It therefore contains an (x, y)-path P. But then $P + e$ is a cycle in the acyclic graph G, a contradiction □

The converse of this theorem holds for graphs without loops (exercise 2.1.1).

Observe that all the trees on six vertices (figure 2.1) have five edges. In general we have:

Theorem 2.2 If G is a tree, then $\varepsilon = \nu - 1$.

Proof By induction on ν. When $\nu = 1$, $G \cong K_1$ and $\varepsilon = 0 = \nu - 1$.

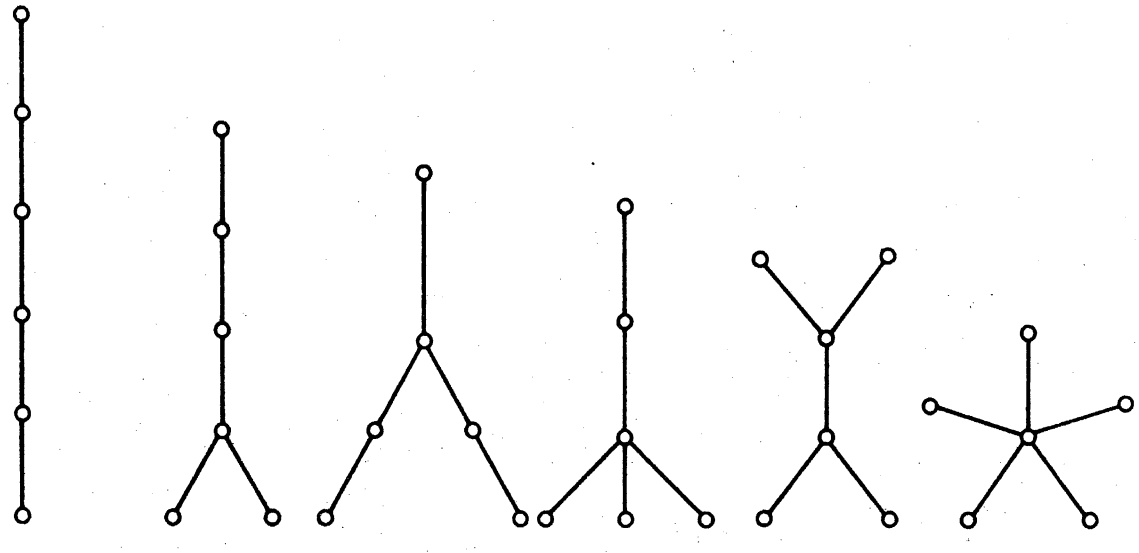


Figure 2.1. The trees on six vertices

Suppose the theorem true for all trees on fewer than ν vertices, and let G be a tree on $\nu \geq 2$ vertices. Let $uv \in E$. Then $G - uv$ contains no (u, v)-path, since uv is the unique (u, v)-path in G. Thus $G - uv$ is disconnected and so (exercise 1.6.8*a*) $\omega(G - uv) = 2$. The components G_1 and G_2 of $G - uv$, being acyclic, are trees. Moreover, each has fewer than ν vertices. Therefore, by the induction hypothesis

$$\varepsilon(G_i) = \nu(G_i) - 1 \quad \text{for} \quad i = 1, 2$$

Thus

$$\varepsilon(G) = \varepsilon(G_1) + \varepsilon(G_2) + 1 = \nu(G_1) + \nu(G_2) - 1 = \nu(G) - 1 \quad \square$$

Corollary 2.2 Every nontrivial tree has at least two vertices of degree one.

Proof Let G be a nontrivial tree. Then

$$d(v) \geq 1 \quad \text{for all} \quad v \in V$$

Also, by theorems 1.1 and 2.2, we have

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

It now follows that $d(v) = 1$ for at least two vertices v $\square$

Another, perhaps more illuminating, way of proving corollary 2.2 is to show that the origin and terminus of a longest path in a nontrivial tree both have degree one (see exercise 2.1.2).

Exercises

2.1.1 Show that if any two vertices of a loopless graph G are connected by a unique path, then G is a tree.

2.1.2 Prove corollary 2.2 by showing that the origin and terminus of a longest path in a nontrivial tree both have degree one.

2.1.3 Prove corollary 2.2 by using exercise 1.7.2.

2.1.4 Show that every tree with exactly two vertices of degree one is a path.

2.1.5 Let G be a graph with $\nu - 1$ edges. Show that the following three statements are equivalent:

(*a*) G is connected;
(*b*) G is acyclic;
(*c*) G is a tree.

2.1.6 Show that if G is a tree with $\Delta \geq k$, then G has at least k vertices of degree one.

2.1.7 An acyclic graph is also called a *forest*. Show that

(*a*) each component of a forest is a tree;
(*b*) G is a forest if and only if $\varepsilon = \nu - \omega$.

2.1.8 A *centre* of G is a vertex u such that $\max_{v \in V} d(u, v)$ is as small as possible. Show that a tree has either exactly one centre or two, adjacent, centres.

2.1.9 Show that if G is a forest with exactly $2k$ vertices of odd degree, then there are k edge-disjoint paths $P_1, P_2, \ldots, P_k$ in G such that $E(G) = E(P_1) \cup E(P_2) \cup \ldots \cup E(P_k)$.

2.1.10* Show that a sequence $(d_1, d_2, \ldots, d_\nu)$ of positive integers is a degree sequence of a tree if and only if $\sum_{i=1}^{\nu} d_i = 2(\nu - 1)$.

2.1.11 Let T be an arbitrary tree on $k+1$ vertices. Show that if G is simple and $\delta \geq k$ then G has a subgraph isomorphic to T.

2.1.12 A saturated hydrocarbon is a molecule C_mH_n in which every carbon atom has four bonds, every hydrogen atom has one bond, and no sequence of bonds forms a cycle. Show that, for every positive integer m, C_mH_n can exist only if $n = 2m + 2$.

2.2 CUT EDGES AND BONDS

A *cut edge* of G is an edge e such that $\omega(G - e) > \omega(G)$. The graph of figure 2.2 has the three cut edges indicated.

Theorem 2.3 An edge e of G is a cut edge of G if and only if e is contained in no cycle of G.

Proof Let e be a cut edge of G. Since $\omega(G - e) > \omega(G)$, there exist vertices u and v of G that are connected in G but not in $G - e$. There is therefore some (u, v)-path P in G which, necessarily, traverses e. Suppose that x and y are the ends of e, and that x precedes y on P. In $G - e$, u is connected to x by a section of P and y is connected to v by a section of P. If e were in a cycle C, x and y would be connected in $G - e$ by the path $C - e$. Thus, u and v would be connected in $G - e$, a contradiction.

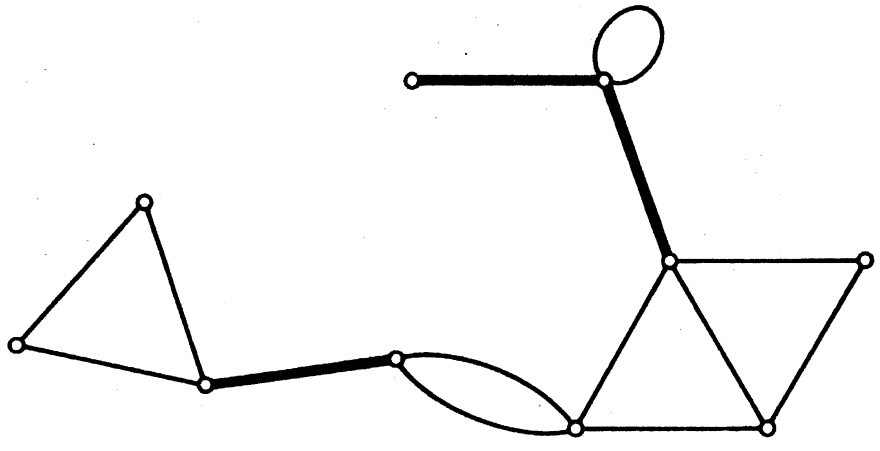


Figure 2.2. The cut edges of a graph

Conversely, suppose that $e = xy$ is not a cut edge of G; thus, $\omega(G-e) = \omega(G)$. Since there is an (x, y)-path (namely xy) in G, x and y are in the same component of G. It follows that x and y are in the same component of $G-e$, and hence that there is an (x, y)-path P in $G-e$. But then e is in the cycle $P+e$ of G □

Theorem 2.4 A connected graph is a tree if and only if every edge is a cut edge.

Proof Let G be a tree and let e be an edge of G. Since G is acyclic, e is contained in no cycle of G and is therefore, by theorem 2.3, a cut edge of G.

Conversely, suppose that G is connected but is not a tree. Then G contains a cycle C. By theorem 2.3, no edge of C can be a cut edge of G □

A *spanning tree* of G is a spanning subgraph of G that is a tree.

Corollary 2.4.1 Every connected graph contains a spanning tree.

Proof Let G be connected and let T be a minimal connected spanning subgraph of G. By definition $\omega(T) = 1$ and $\omega(T-e) > 1$ for each edge e of T. It follows that each edge of T is a cut edge and therefore, by theorem 2.4, that T, being connected, is a tree □

Figure 2.3 depicts a connected graph and one of its spanning trees.

Corollary 2.4.2 If G is connected, then $\varepsilon \geq \nu - 1$.

Proof Let G be connected. By corollary 2.4.1, G contains a spanning tree T. Therefore

$$\varepsilon(G) \geq \varepsilon(T) = \nu(T) - 1 = \nu(G) - 1 \quad \square$$

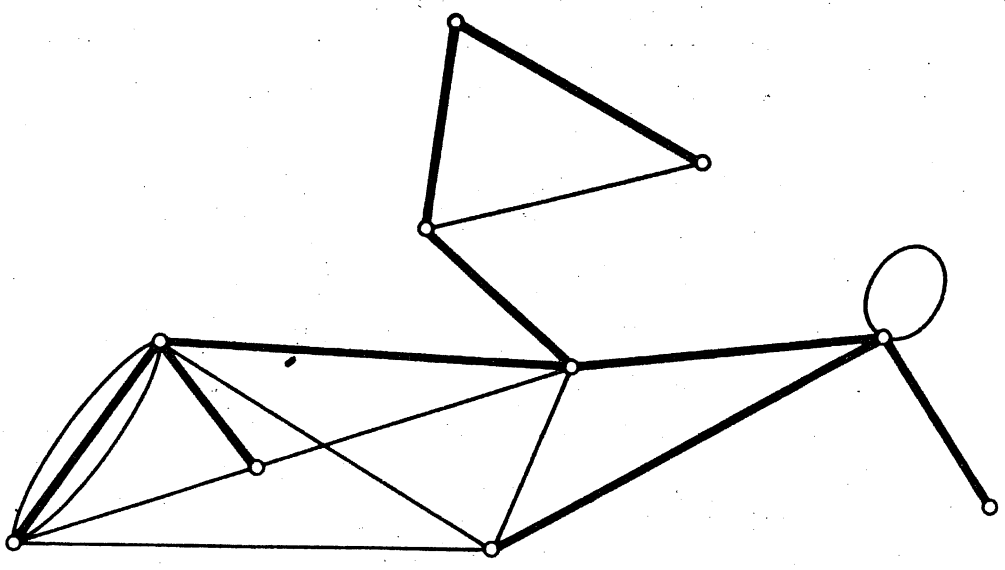


Figure 2.3. A spanning tree in a connected graph

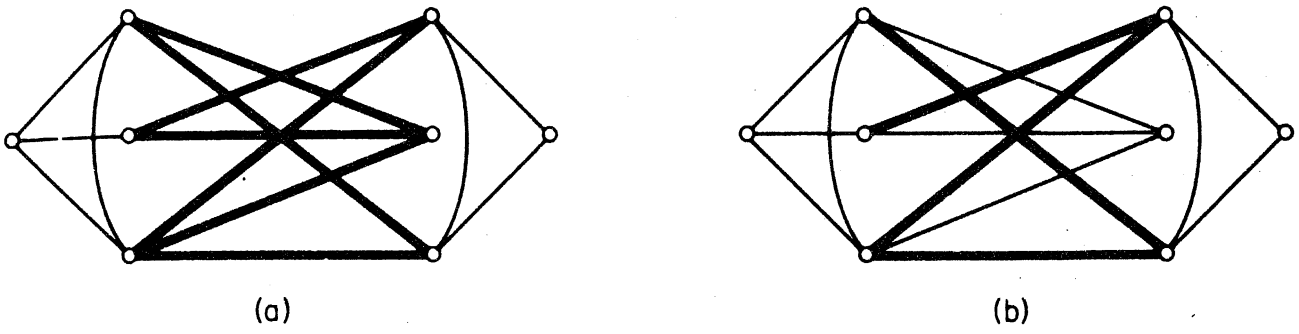


Figure 2.4. (*a*) An edge cut; (*b*) a bond

Theorem 2.5 Let T be a spanning tree of a connected graph G and let e be an edge of G not in T. Then $T+e$ contains a unique cycle.

Proof Since T is acyclic, each cycle of $T+e$ contains e. Moreover, C is a cycle of $T+e$ if and only if $C-e$ is a path in T connecting the ends of e. By theorem 2.1, T has a unique such path; therefore $T+e$ contains a unique cycle □

For subsets S and S' of V, we denote by $[S, S']$ the set of edges with one end in S and the other in S'. An *edge cut* of G is a subset of E of the form $[S, \bar{S}]$, where S is a nonempty proper subset of V and $\bar{S} = V \backslash S$. A minimal nonempty edge cut of G is called a *bond*; each cut edge e, for instance, gives rise to a bond $\{e\}$. If G is connected, then a bond B of G is a minimal subset of E such that $G-B$ is disconnected. Figure 2.4 indicates an edge cut and a bond in a graph.

If H is a subgraph of G, the *complement of H in G*, denoted by $\bar{H}(G)$, is the subgraph $G-E(H)$. If G is connected, a subgraph of the form $\bar{T}$, where T is a spanning tree, is called a *cotree* of G.

Theorem 2.6 Let T be a spanning tree of a connected graph G, and let e be any edge of T. Then

(i) the cotree $\bar{T}$ contains no bond of G;
(ii) $\bar{T}+e$ contains a unique bond of G.

Proof (i) Let B be a bond of G. Then $G-B$ is disconnected, and so cannot contain the spanning tree T. Therefore B is not contained in $\bar{T}$. (ii) Denote by S the vertex set of one of the two components of $T-e$. The edge cut $B=[S, \bar{S}]$ is clearly a bond of G, and is contained in $\bar{T}+e$. Now, for any $b \in B$, $T-e+b$ is a spanning tree of G. Therefore every bond of G contained in $\bar{T}+e$ must include every such element b. It follows that B is the only bond of G contained in $\bar{T}+e$ □

The relationship between bonds and cotrees is analogous to that between cycles and spanning trees. Statement (i) of theorem 2.6 is the analogue for

bonds of the simple fact that a spanning tree is acyclic, and (ii) is the analogue of theorem 2.5. This 'duality' between cycles and bonds will be further explored in chapter 12 (see also exercise 2.2.10).

Exercises

2.2.1 Show that G is a forest if and only if every edge of G is a cut edge.

2.2.2 Let G be connected and let $e \in E$. Show that

(*a*) e is in every spanning tree of G if and only if e is a cut edge of G;

(*b*) e is in no spanning tree of G if and only if e is a loop of G.

2.2.3 Show that if G is loopless and has exactly one spanning tree T, then $G = T$.

2.2.4 Let F be a maximal forest of G. Show that

(*a*) for every component H of G, $F \cap H$ is a spanning tree of H;

(*b*) $\varepsilon(F) = \nu(G) - \omega(G)$.

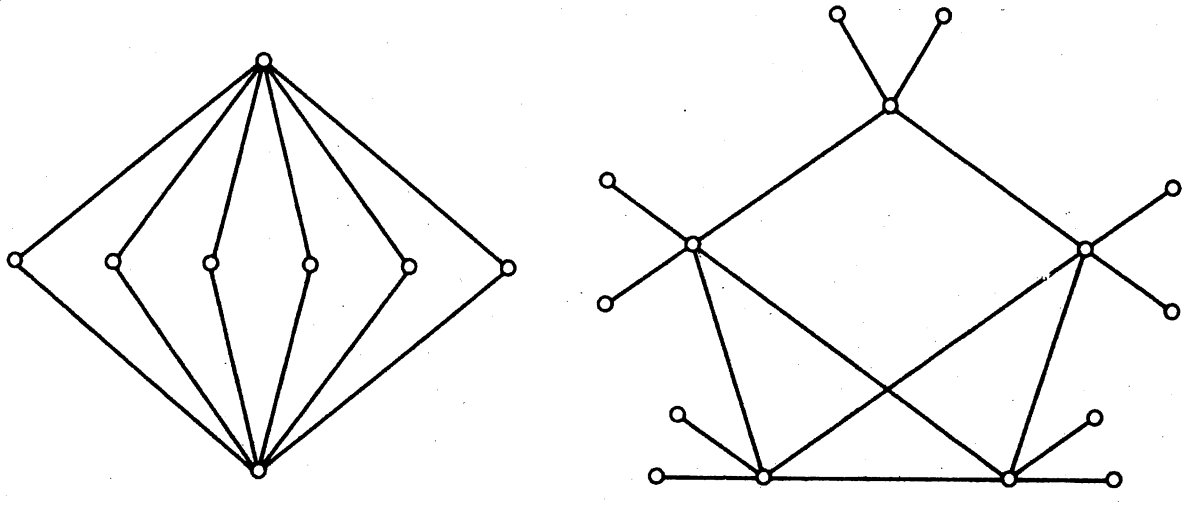
2.2.5 Show that G contains at least $\varepsilon - \nu + \omega$ distinct cycles.

2.2.6 Show that

(*a*) if each degree in G is even, then G has no cut edge;

(*b*) if G is a k-regular bipartite graph with $k \geq 2$, then G has no cut edge.

2.2.7 Find the number of nonisomorphic spanning trees in the following graphs:



2.2.8 Let G be connected and let S be a nonempty proper subset of V. Show that the edge cut $[S, \bar{S}]$ is a bond of G if and only if both $G[S]$ and $G[\bar{S}]$ are connected.

2.2.9 Show that every edge cut is a disjoint union of bonds.

2.2.10 Let B_1 and B_2 be bonds and let C_1 and C_2 be cycles (regarded as

sets of edges) in a graph. Show that

(*a*) $B_1 \Delta B_2$ is a disjoint union of bonds;
(*b*) $C_1 \Delta C_2$ is a disjoint union of cycles,

where Δ denotes symmetric difference;

(*c*) for any edge e, $(B_1 \cup B_2)\backslash\{e\}$ contains a bond;
(*d*) for any edge e, $(C_1 \cup C_2)\backslash\{e\}$ contains a cycle.

2.2.11 Show that if a graph G contains k edge-disjoint spanning trees then, for each partition $(V_1, V_2, \ldots, V_n)$ of V, the number of edges which have ends in different parts of the partition is at least $k(n-1)$.

(Tutte, 1961 and Nash-Williams, 1961 have shown that this necessary condition for G to contain k edge-disjoint spanning trees is also sufficient.)

2.2.12* Let S be an n-element set, and let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be a family of n distinct subsets of S. Show that there is an element $x \in S$ such that the sets $A_1 \cup \{x\}, A_2 \cup \{x\}, \ldots, A_n \cup \{x\}$ are all distinct.

2.3 CUT VERTICES

A vertex v of G is a *cut vertex* if E can be partitioned into two nonempty subsets E_1 and E_2 such that $G[E_1]$ and $G[E_2]$ have just the vertex v in common. If G is loopless and nontrivial, then v is a cut vertex of G if and only if $\omega(G-v) > \omega(G)$. The graph of figure 2.5 has the five cut vertices indicated.

Theorem 2.7 A vertex v of a tree G is a cut vertex of G if and only if $d(v) > 1$.

Proof If $d(v) = 0$, $G \cong K_1$ and, clearly, v is not a cut vertex.

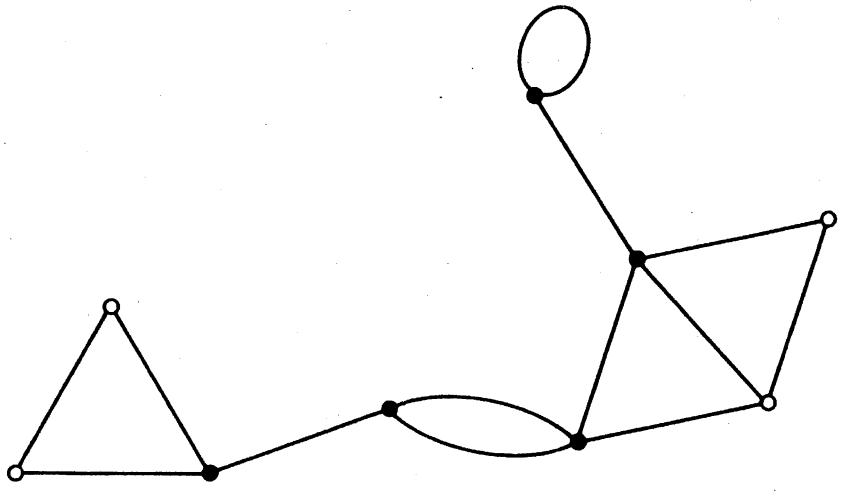


Figure 2.5. The cut vertices of a graph

If $d(v)=1$, $G-v$ is an acyclic graph with $\nu(G-v)-1$ edges, and thus (exercise 2.1.5) a tree. Hence $\omega(G-v)=1=\omega(G)$, and v is not a cut vertex of G.

If $d(v)>1$, there are distinct vertices u and w adjacent to v. The path uvw is a (u, w)-path in G. By theorem 2.1 uvw is the unique (u, w)-path in G. It follows that there is no (u, w)-path in $G-v$, and therefore that $\omega(G-v)>1=\omega(G)$. Thus v is a cut vertex of G □

Corollary 2.7 Every nontrivial loopless connected graph has at least two vertices that are not cut vertices.

Proof Let G be a nontrivial loopless connected graph. By corollary 2.4.1, G contains a spanning tree T. By corollary 2.2 and theorem 2.7, T has at least two vertices that are not cut vertices. Let v be any such vertex. Then

$$\omega(T-v)=1$$

Since T is a spanning subgraph of G, $T-v$ is a spanning subgraph of $G-v$ and therefore

$$\omega(G-v)\leq\omega(T-v)$$

It follows that $\omega(G-v)=1$, and hence that v is not a cut vertex of G. Since there are at least two such vertices v, the proof is complete □

Exercises

2.3.1 Let G be connected with $\nu\geq 3$. Show that

(*a*) if G has a cut edge, then G has a vertex v such that $\omega(G-v)>\omega(G)$;

(*b*) the converse of (*a*) is not necessarily true.

2.3.2 Show that a simple connected graph that has exactly two vertices which are not cut vertices is a path.

2.4 CAYLEY'S FORMULA

There is a simple and elegant recursive formula for the number of spanning trees in a graph. It involves the operation of contraction of an edge, which we now introduce. An edge e of G is said to be *contracted* if it is deleted and its ends are identified; the resulting graph is denoted by $G\cdot e$. Figure 2.6 illustrates the effect of contracting an edge.

It is clear that if e is a link of G, then

$$\nu(G\cdot e)=\nu(G)-1 \qquad \varepsilon(G\cdot e)=\varepsilon(G)-1 \quad \text{and} \quad \omega(G\cdot e)=\omega(G)$$

Therefore, if T is a tree, so too is $T\cdot e$.

We denote the number of spanning trees of G by $\tau(G)$.

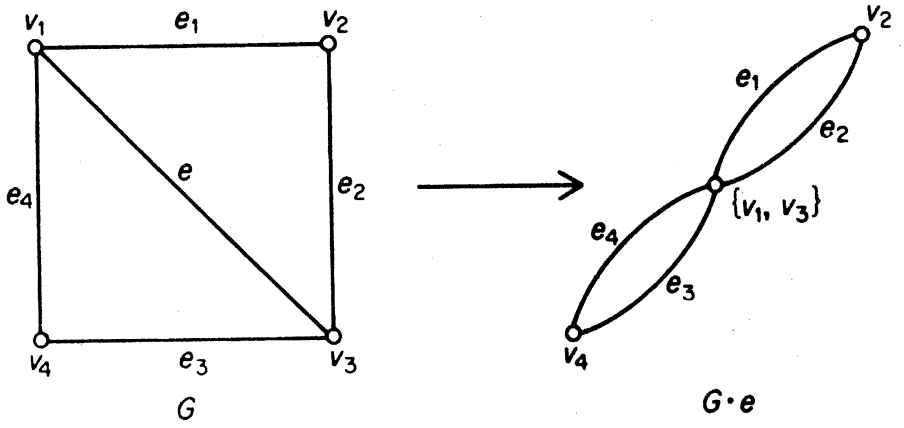


Figure 2.6. Contraction of an edge

Theorem 2.8 If e is a link of G, then $\tau(G) = \tau(G-e) + \tau(G \cdot e)$.

Proof Since every spanning tree of G that does not contain e is also a spanning tree of $G-e$, and conversely, $\tau(G-e)$ is the number of spanning trees of G that do not contain e.

Now to each spanning tree T of G that contains e, there corresponds a spanning tree $T \cdot e$ of $G \cdot e$. This correspondence is clearly a bijection (see figure 2.7). Therefore $\tau(G \cdot e)$ is precisely the number of spanning trees of G that contain e. It follows that $\tau(G) = \tau(G-e) + \tau(G \cdot e)$ □

Figure 2.8 illustrates the recursive calculation of $\tau(G)$ by means of theorem 2.8; the number of spanning trees in a graph is represented symbolically by the graph itself.

Although theorem 2.8 provides a method of calculating the number of spanning trees in a graph, this method is not suitable for large graphs. Fortunately, and rather surprisingly, there is a closed formula for $\tau(G)$ which expresses $\tau(G)$ as a determinant; we shall present this result in chapter 12. In the special case when G is complete, a simple formula for $\tau(G)$ was discovered by Cayley (1889). The proof we give is due to Prüfer (1918).

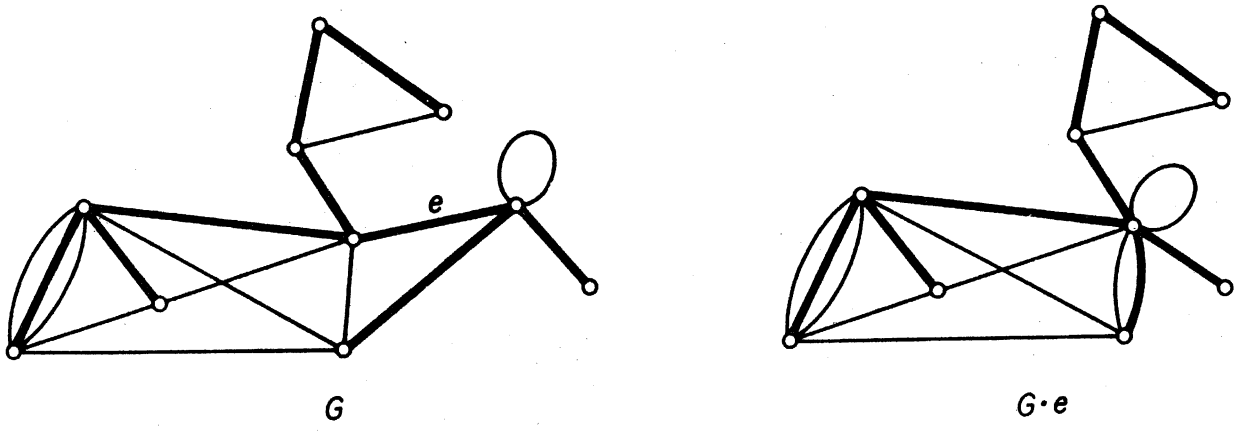


Figure 2.7

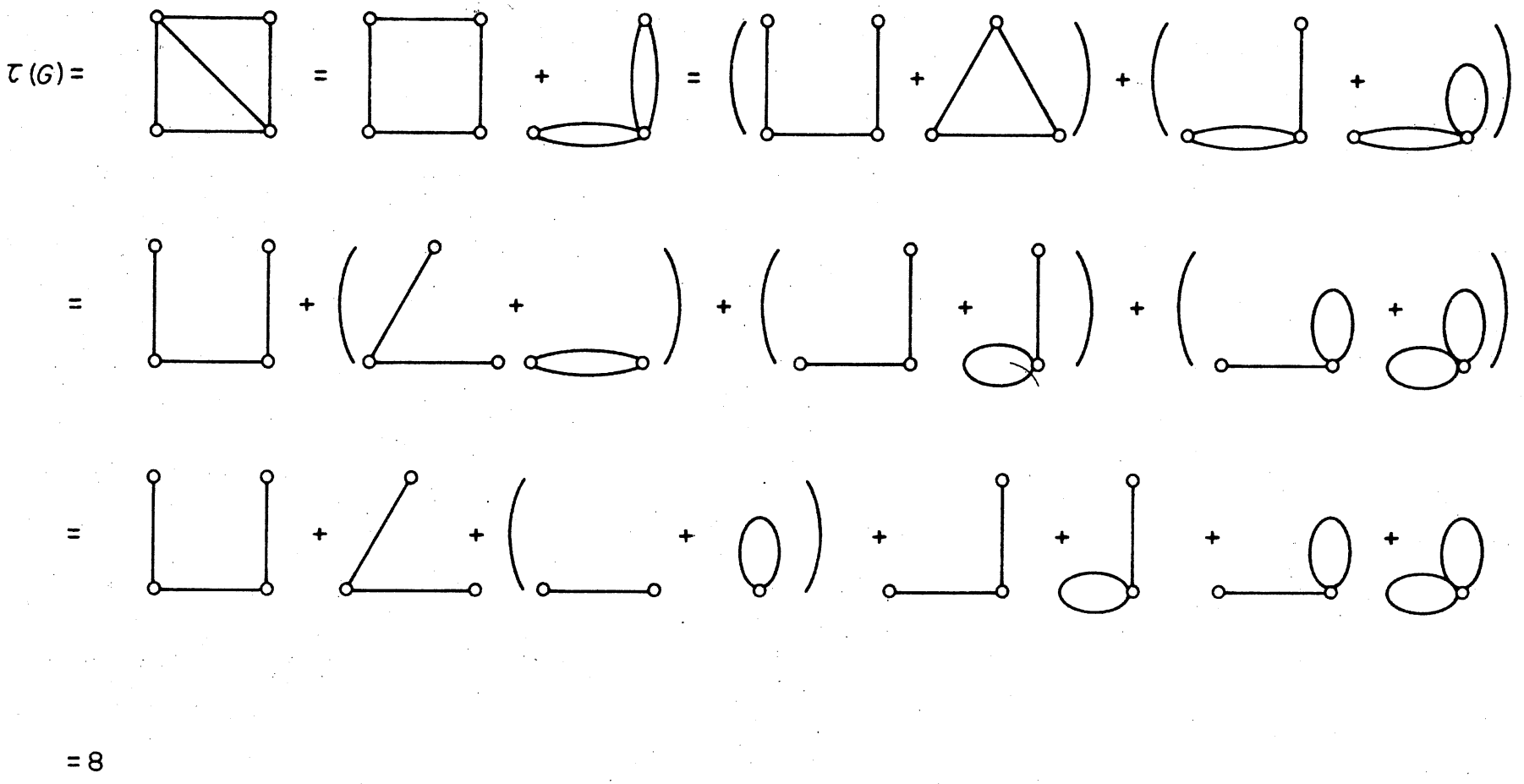


Figure 2.8. Recursive calculation of $\tau(G)$

Theorem 2.9 $\tau(K_n) = n^{n-2}$.

Proof Let the vertex set of K_n be $N = \{1, 2, \ldots, n\}$. We note that n^{n-2} is the number of sequences of length $n-2$ that can be formed from N. Thus, to prove the theorem, it suffices to establish a one–one correspondence between the set of spanning trees of K_n and the set of such sequences.

With each spanning tree T of K_n, we associate a unique sequence $(t_1, t_2, \ldots, t_{n-2})$ as follows. Regarding N as an ordered set, let s_1 be the first vertex of degree one in T; the vertex adjacent to s_1 is taken as t_1. We now delete s_1 from T, denote by s_2 the first vertex of degree one in $T - s_1$, and take the vertex adjacent to s_2 as t_2. This operation is repeated until t_{n-2} has been defined and a tree with just two vertices remains; the tree in figure 2.9, for instance, gives rise to the sequence (4, 3, 5, 3, 4, 5). It can be seen that different spanning trees of K_n determine difference sequences.

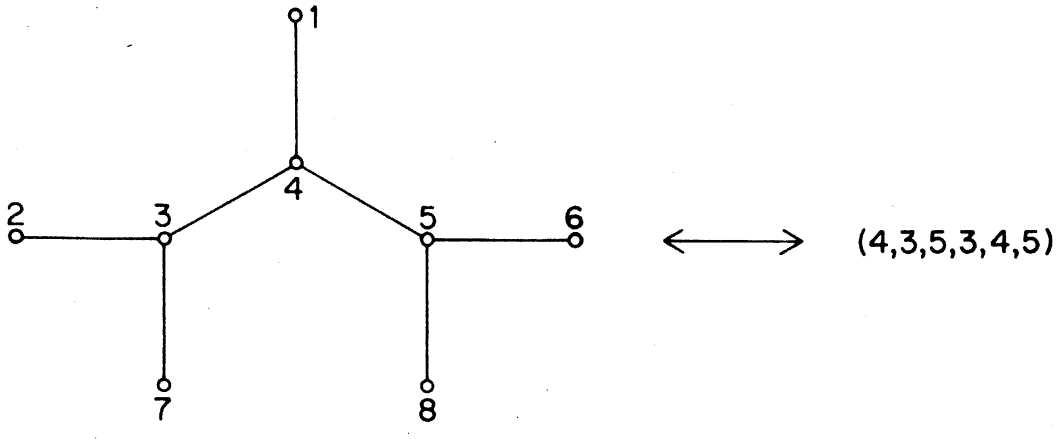


Figure 2.9

The reverse procedure is equally straightforward. Observe, first, that any vertex v of T occurs $d_T(v) - 1$ times in $(t_1, t_2, \ldots, t_{n-2})$. Thus the vertices of degree one in T are precisely those that do not appear in this sequence. To reconstruct T from $(t_1, t_2, \ldots, t_{n-2})$, we therefore proceed as follows. Let s_1 be the first vertex of N not in $(t_1, t_2, \ldots, t_{n-2})$; join s_1 to t_1. Next, let s_2 be the first vertex of $N \backslash \{s_1\}$ not in $(t_2, \ldots, t_{n-2})$, and join s_2 to t_2. Continue in this way until the $n-2$ edges $s_1t_1, s_2t_2, \ldots, s_{n-2}t_{n-2}$ have been determined. T is now obtained by adding the edge joining the two remaining vertices of $N \backslash \{s_1, s_2, \ldots, s_{n-2}\}$. It is easily verified that different sequences give rise to different spanning trees of K_n. We have thus established the desired one–one correspondence □

Note that n^{n-2} is not the number of nonisomorphic spanning trees of K_n, but the number of distinct spanning trees of K_n; there are just six nonisomorphic spanning trees of K_6 (see figure 2.1), whereas there are $6^4 = 1296$ distinct spanning trees of K_6.

Exercises

2.4.1 Using the recursion formula of theorem 2.8, evaluate the number of spanning trees in $K_{3,3}$.

2.4.2* A *wheel* is a graph obtained from a cycle by adding a new vertex and edges joining it to all the vertices of the cycle; the new edges are called the *spokes* of the wheel. Obtain an expression for the number of spanning trees in a wheel with n spokes.

2.4.3 Draw all sixteen spanning trees of K_4.

2.4.4 Show that if e is an edge of K_n, then $\tau(K_n - e) = (n-2)n^{n-3}$.

2.4.5 (*a*) Let H be a graph in which every two adjacent vertices are joined by k edges and let G be the underlying simple graph of H. Show that $\tau(H) = k^{\nu-1}\tau(G)$.

(*b*) Let H be the graph obtained from a graph G when each edge of G is replaced by a path of length k. Show that $\tau(H) = k^{\epsilon-\nu+1}\tau(G)$.

(*c*) Deduce from (*b*) that $\tau(K_{2,n}) = n2^{n-1}$.

APPLICATIONS

2.5 THE CONNECTOR PROBLEM

A railway network connecting a number of towns is to be set up. Given the cost c_{ij} of constructing a direct link between towns v_i and v_j, design such a network to minimise the total cost of construction. This is known as the *connector problem.*

By regarding each town as a vertex in a weighted graph with weights $w(v_iv_j) = c_{ij}$, it is clear that this problem is just that of finding, in a weighted graph G, a connected spanning subgraph of minimum weight. Moreover, since the weights represent costs, they are certainly non-negative, and we may therefore assume that such a minimum-weight spanning subgraph is a spanning tree T of G. A minimum-weight spanning tree of a weighted graph will be called an *optimal tree*; the spanning tree indicated in the weighted graph of figure 2.10 is an optimal tree (exercise 2.5.1).

We shall now present a good algorithm for finding an optimal tree in a nontrivial weighted connected graph, thereby solving the connector problem.

Consider, first, the case when each weight $w(e) = 1$. An optimal tree is then a spanning tree with as few edges as possible. Since each spanning tree of a graph has the same number of edges (theorem 2.2), in this special case we merely need to construct some spanning tree of the graph. A simple

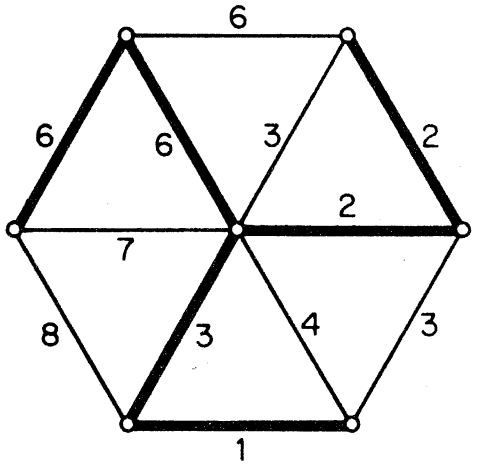


Figure 2.10. An optimal tree in a weighted graph

inductive algorithm for finding such a tree is the following:

1. Choose a link e_1.
2. If edges $e_1, e_2, \ldots, e_i$ have been chosen, then choose e_{i+1} from $E\backslash\{e_1, e_2, \ldots, e_i\}$ in such a way that $G[\{e_1, e_2, \ldots, e_{i+1}\}]$ is acyclic.
3. Stop when step 2 cannot be implemented further.

This algorithm works because a maximal acyclic subgraph of a connected graph is necessarily a spanning tree. It was extended by Kruskal (1956) to solve the general problem; his algorithm is valid for arbitrary real weights.

Kruskal's Algorithm

1. Choose a link e_1 such that $w(e_1)$ is as small as possible.
2. If edges $e_1, e_2, \ldots, e_i$ have been chosen, then choose an edge e_{i+1} from $E\backslash\{e_1, e_2, \ldots, e_i\}$ in such a way that
   (i) $G[\{e_1, e_2, \ldots, e_{i+1}\}]$ is acyclic;
   (ii) $w(e_{i+1})$ is as small as possible subject to (i).
3. Stop when step 2 cannot be implemented further.

As an example, consider the table of airline distances in miles between six of the largest cities in the world, London, Mexico City, New York, Paris, Peking and Tokyo:

	L	MC	NY	Pa	Pe	T
L	—	5558	3469	214	5074	5959
MC	5558	—	2090	5725	7753	7035
NY	3469	2090	—	3636	6844	6757
Pa	214	5725	3636	—	5120	6053
Pe	5074	7753	6844	5120	—	1307
T	5959	7035	6757	6053	1307	—

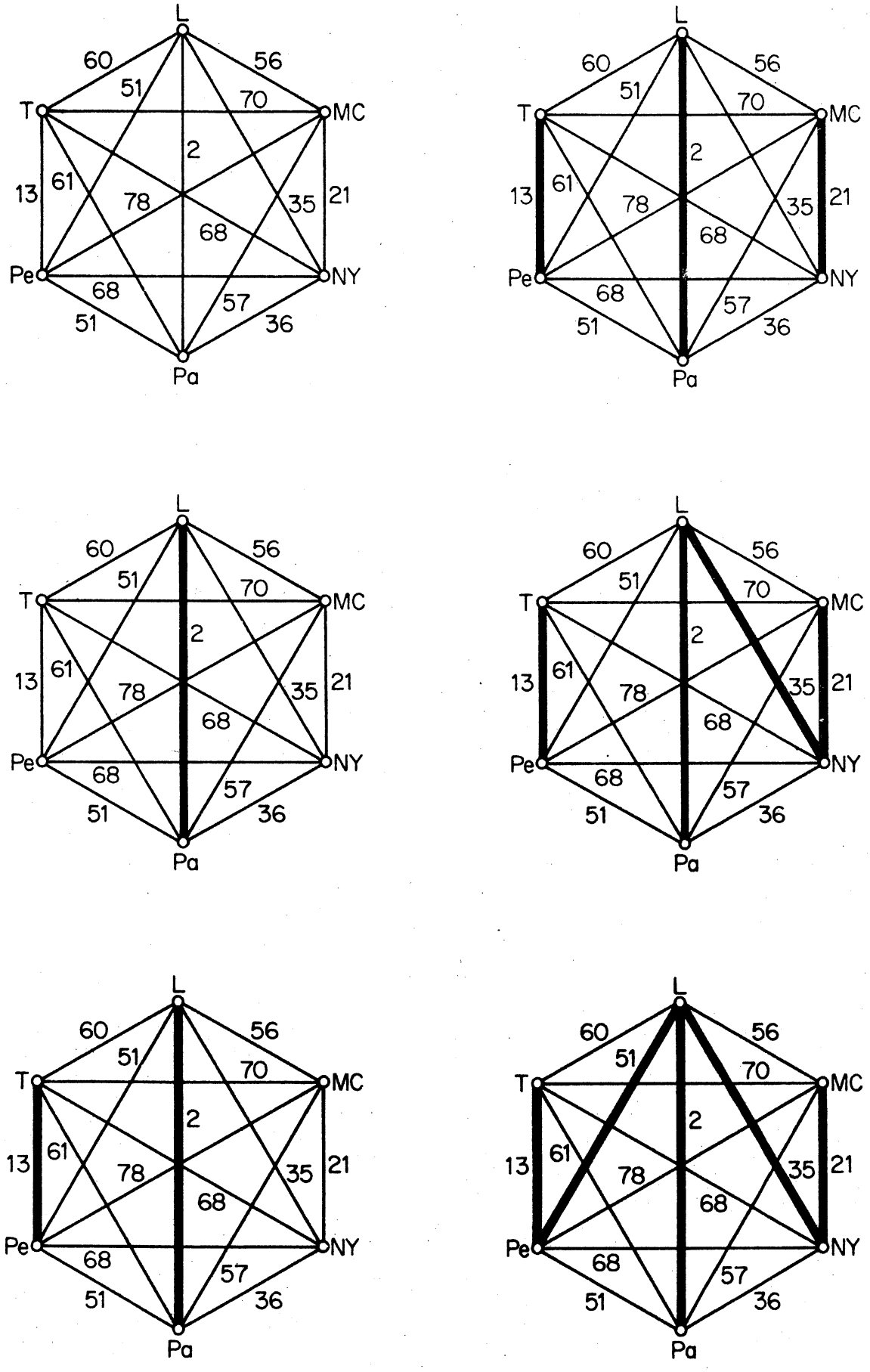


Figure 2.11

This table determines a weighted complete graph with vertices L, MC, NY, Pa, Pe and T. The construction of an optimal tree in this graph is shown in figure 2.11 (where, for convenience, distances are given in hundreds of miles).

Kruskal's algorithm clearly produces a spanning tree (for the same reason that the simpler algorithm above does). The following theorem ensures that such a tree will always be optimal.

Theorem 2.10 Any spanning tree $T^* = G[\{e_1, e_2, \ldots, e_{\nu-1}\}]$ constructed by Kruskal's algorithm is an optimal tree.

Proof By contradiction. For any spanning tree T of G other than T^*, denote by $f(T)$ the smallest value of i such that e_i is not in T. Now assume that T^* is not an optimal tree, and let T be an optimal tree such that $f(T)$ is as large as possible.

Suppose that $f(T) = k$; this means that $e_1, e_2, \ldots, e_{k-1}$ are in both T and T^*, but that e_k is not in T. By theorem 2.5, $T + e_k$ contains a unique cycle C. Let e'_k be an edge of C that is in T but not in T^*. By theorem 2.3, e'_k is not a cut edge of $T + e_k$. Hence $T' = (T + e_k) - e'_k$ is a connected graph with $\nu - 1$ edges, and therefore (exercise 2.1.5) is another spanning tree of G. Clearly

$$w(T') = w(T) + w(e_k) - w(e'_k) \tag{2.1}$$

Now, in Kruskal's algorithm, e_k was chosen as an edge with the smallest weight such that $G[\{e_1, e_2, \ldots, e_k\}]$ was acyclic. Since $G[\{e_1, e_2, \ldots, e_{k-1}, e'_k\}]$ is a subgraph of T, it is also acyclic. We conclude that

$$w(e'_k) \geq w(e_k) \tag{2.2}$$

Combining (2.1) and (2.2) we have

$$w(T') \leq w(T)$$

and so T', too, is an optimal tree. However

$$f(T') > k = f(T)$$

contradicting the choice of T. Therefore $T = T^*$, and T^* is indeed an optimal tree □

A flow diagram for Kruskal's algorithm is shown in figure 2.12. The edges are first sorted in order of increasing weight (box 1); this takes about $\varepsilon \log \varepsilon$ computations (see Knuth, 1973). Box 2 just checks to see how many edges have been chosen. (S is the set of edges already chosen and i is their number.) When $i = \nu - 1$, $S = \{e_1, e_2, \ldots, e_{\nu-1}\}$ is the edge set of an optimal tree T^* of G. In box 3, to check if $G[S \cup \{a_j\}]$ is acyclic, one must ascertain whether the ends of a_j are in different components of the forest $G[S]$ or not. This can be achieved in the following way. The vertices are labelled so that, at any stage, two vertices belong to the same component of $G[S]$ if and only

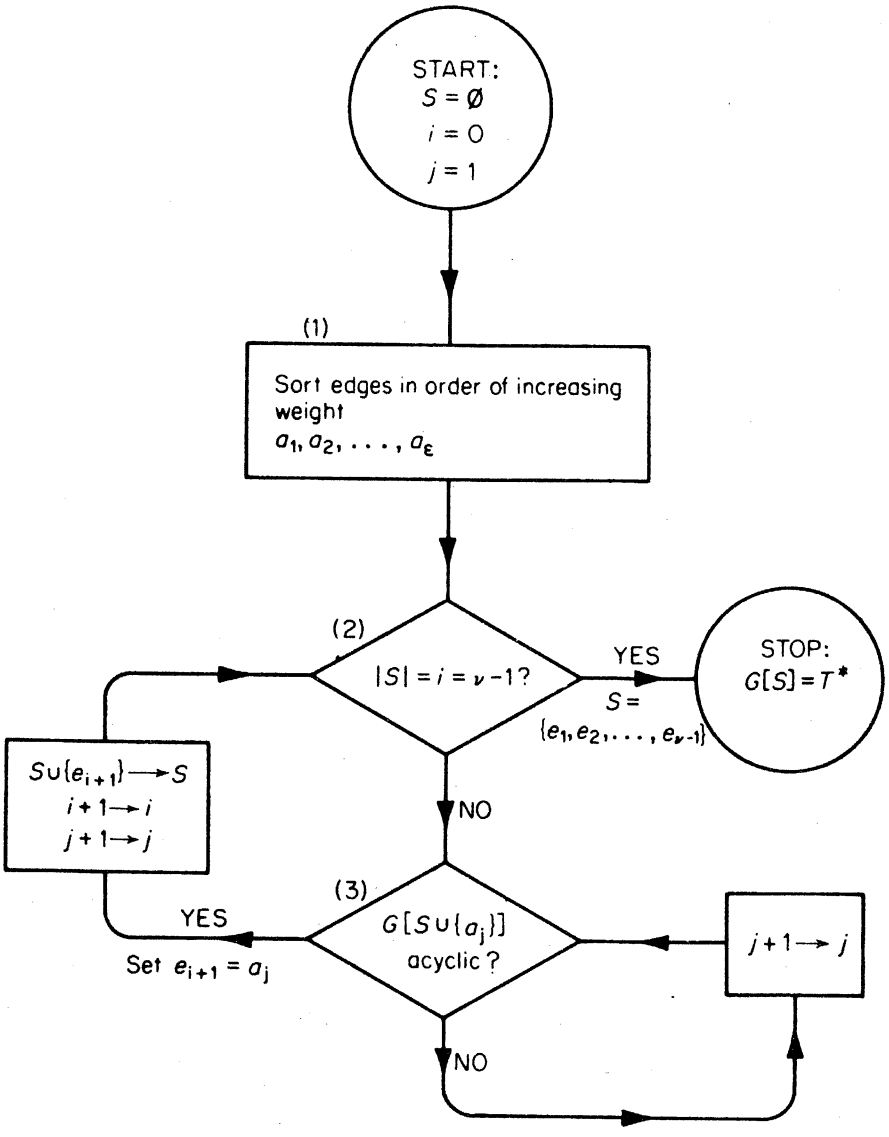


Figure 2.12. Kruskal's algorithm

if they have the same label; initially, vertex v_l is assigned the label l, $1 \leq l \leq \nu$. With this labelling scheme, $G[S \cup \{a_j\}]$ is acyclic if and only if the ends of a_j have different labels. If this is the case, a_j is taken as e_{i+1}; otherwise, a_j is discarded and a_{j+1}, the next candidate for e_{i+1}, is tested. Once e_{i+1} has been added to S, the vertices in the two components of $G[S]$ that contain the ends of e_{i+1} are relabelled with the smaller of their two labels. For each edge, one comparison suffices to check whether its ends have the same or different labels; this takes ε computations. After edge e_{i+1} has been added to S, the relabelling of vertices takes at most ν comparisons; hence, for all $\nu - 1$ edges $e_1, e_2, \ldots, e_{\nu-1}$ we need $\nu(\nu-1)$ computations. Kruskal's algorithm is therefore a good algorithm.

Exercises

2.5.1 Show, by applying Kruskal's algorithm, that the tree indicated in figure 2.10 is indeed optimal.

2.5.2 Adapt Kruskal's algorithm to solve the *connector problem with preassignments*: construct, at minimum cost, a network linking a number of towns, with the additional requirement that certain selected pairs of towns be directly linked.

2.5.3 Can Kruskal's algorithm be adapted to find

(*a*) a *maximum*-weight tree in a weighted connected graph?
(*b*) a minimum-weight maximal *forest* in a weighted graph?

If so, how?

2.5.4 Show that the following Kruskal-type algorithm does not necessarily yield a minimum-weight spanning *path* in a weighted complete graph:

1. Choose a link e_1 such that $w(e_1)$ is as small as possible.
2. If edges $e_1, e_2, \ldots, e_i$ have been chosen, then choose an edge e_{i+1} from $E\backslash\{e_1, e_2, \ldots, e_i\}$ in such a way that
   (i) $G[\{e_1, e_2, \ldots, e_{i+1}\}]$ is a union of disjoint paths;
   (ii) $w(e_{i+1})$ is as small as possible subject to (i).
3. Stop when step 2 cannot be implemented further.

2.5.5 The *tree graph* of a connected graph G is the graph whose vertices are the spanning trees $T_1, T_2, \ldots, T_\tau$ of G, with T_i and T_j joined if and only if they have exactly $\nu - 2$ edges in common. Show that the tree graph of any connected graph is connected.

REFERENCES

Cayley, A. (1889). A theorem on trees. *Quart. J. Math.*, **23,** 376–78

Knuth, D. E. (1973). *The Art of Computer Programming*, vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., p. 184

Kruskal, J. B. Jr. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, **7,** 48–50

Nash-Williams, C. St. J. A. (1961). Edge-disjoint spanning trees of finite graphs. *J. London Math. Soc.*, **36,** 445–50

Prüfer, H. (1918). Neuer Beweis eines Satzes über Permutationen. *Arch. Math. Phys.*, **27,** 742–44

Tutte, W. T. (1961). On the problem of decomposing a graph into n connected factors. *J. London Math. Soc.*, **36,** 221–30