Chapter 2

TREES

2.1 TREE DEFINITIONS

Let $G(V, E)$ be an (undirected), finite or infinite graph. We say that G is *circuit-free* if there are no simple circuits in G. G is called a *tree* if it is connected and circuit-free.

Theorem 2.1: The following four conditions are equivalent:

(a) G is a tree.
(b) G is circuit-free, but if any new edge is added to G, a circuit is formed.
(c) G contains no self-loops and for every two vertices there is a unique simple path connecting them.
(d) G is connected, but if any edge is deleted from G, the connectivity of G is interrupted.

Proof: We shall prove that conditions (a) $\Rightarrow$ (b) $\Rightarrow$ (c) $\Rightarrow$ (d) $\Rightarrow$ (a).

(a) $\Rightarrow$ (b): We assume that G is connected and circuit-free. Let e be a new edge, that is $e \notin E$; the two endpoints of e, a and b, are elements of V. If $a = b$, then e forms a self-loop and therefore a circuit exists. If $a \neq b$, there is a path in G (without e) between a and b; if we add e, this path with e forms a circuit.

(b) $\Rightarrow$ (c): We assume that G is circuit-free and that no edge can be added to G without creating a circuit. Let a and b be any two vertices of G. If there is no path between them, then we can add an edge between a and b without creating a circuit. Thus, G must be connected. Moreover, if there are two simple paths, P and P', between a and b, then there is a circuit in G. To see this, assume that $P = e_1, e_2, \ldots, e_l$ and $P' = e_1', e_2', \ldots, e_m'$. Since both paths are simple, one cannot be the beginning of the other. Let i be the first index for which $e_i \neq e_i'$, and let v be the first vertex on $e_i, e_{i+1}, \ldots, e_l$ which is also on $e_i', e_{i+1}', \ldots, e_m'$. The two

disjoint subpaths between the branching off vertex and v form a simple circuit in G.

(c) $\Rightarrow$ (d): We assume the existence of a unique simple path between every pair of vertices of G. This implies that G is connected. Assume now that we delete an edge e from G. Since G has no self-loops, e is not a self-loop. Let a and b be e's endpoints. If there is now (after the deletion of e) a path between a and b, then G has more than one simple path between a and b.

(d) $\Rightarrow$ (a): We assume that G is connected and that no edge can be deleted without interrupting the connectivity. If G contains a simple circuit, any edge on this circuit can be deleted without interrupting the connectivity. Thus, G is circuit-free.

Q.E.D.

There are two more common ways to define a finite tree. These are given in the following theorem.

Theorem 2.2: Let $G(V, E)$ be a finite graph and $n = |V|$. The following three conditions are equivalent:

(a) G is a tree.
(b) G is circuit-free and has $n - 1$ edges.
(c) G is connected and has $n - 1$ edges.

Proof: For $n = 1$ the theorem is trivial. Assume $n \geq 2$. We shall prove that conditions (a) $\Rightarrow$ (b) $\Rightarrow$ (c) $\Rightarrow$ (a).

(a) $\Rightarrow$ (b): Let us prove, by induction on n, that if G is a tree, then its number of edges is $n - 1$. This statement is clearly true for $n = 1$. Assume that it is true for all $n < m$, and let G be a tree with m vertices. Let us delete from G any edge e. By condition (d) of Theorem 2.1, G is not connected any more, and clearly is broken into two connected components each of which is circuit-free and therefore is a tree. By the inductive hypothesis, each component has one edge less than the number of vertices. Thus, both have $m - 2$ edges. Add back e, and the number of edges is $m - 1$.

(b) $\Rightarrow$ (c): We assume that G is circuit-free and has $n - 1$ edges. Let us first show that G has at least two vertices of degree 1. Choose any edge e. An edge must exist since the number of edges is $n - 1$ and $n \geq 2$. Extend the edge into a path by adding new edges to its ends if such exist. A new edge attached at the path's end introduces a new vertex to the path or a

circuit is closed. Thus, our path remains simple. Since the graph is finite, this extension must terminate on both sides of e, yielding two vertices of degree 1.

Now, the proof that G is connected proceeds by induction on the number of vertices, n. The statement is obviously true for $n = 2$. Assume that it is true for $n = m - 1$, and let G be a circuit-free graph with m vertices and $m - 1$ edges. Eliminate from G a vertex v, of degree 1, and its incident edge. The resulting graph is still circuit-free and has $m - 1$ vertices and $m - 2$ edges; thus, by the inductive hypothesis it is connected. Therefore, G is connected too.

(c) $\Rightarrow$ (a): Assume that G is connected and has $n - 1$ edges. If G contains circuits, we can eliminate edges (without eliminating vertices) and maintain the connectivity. When this process terminates, the resulting graph is a tree, and, by (a) $\Rightarrow$ (b), has $n - 1$ edges. Thus, no edge can be eliminated and G is circuit-free.

Q.E.D.

Let us call a vertex whose degree is 1, a *leaf*. A corollary of Theorem 2.2 and the statement proved in the (b) $\Rightarrow$ (c) part of its proof is the following corollary:

Corollary 2.1: A finite tree, with more than one vertex, has at least two leaves.

2.2 MINIMUM SPANNING TREE

A graph $G'(V', E')$ is called a *subgraph* of a graph $G(V, E)$, if $V' \subseteq V$ and $E' \subseteq E$. Clearly, an arbitrary choice of $V' \subseteq V$ and $E' \subseteq E$ may not yield a subgraph, simply because it may not be a graph; that is, some of the endpoints of edges in E' may not be in V'.

Assume $G(V, E)$ is a finite, connected (undirected) graph and each edge $e \in E$ has a known length $l(e) > 0$. Assume we want to find a connected subgraph $G'(V, E')$ whose length, $\Sigma_{e \in E} l(e)$, is minimum; or, in other words, we want to remove from G a subset of edges whose total length is maximum, and which leaves it still connected. It is clear that such a subgraph is a tree. For G' is assumed to be connected, and since its length is minimum, none of its edges can be removed without destroying its connectivity. By Theorem 2.1 (see part (d)) G' is a tree. A subgraph of G, which contains all of its vertices and is a tree is called a *spanning tree* of G. Thus, our problem is that of finding a minimum-length spanning tree of G.

There are many known algorithms for the minimum spanning tree problem, but they all hinge on the following theorem:

Theorem 2.3: Let $U \subset V$ and e be of minimum length among the edges with one endpoint in U and the other endpoint in $V - U$. There exists a minimum spanning tree T such that e is in T.

Proof: Let T_0 be a minimum spanning tree. If e is not in T_0, add e to T_0. By Theorem 2.1 (part (b)) a circuit is formed. This circuit contains e and at least one more edge $u \overset{e'}{-} v$, where $u \in U$ and $v \in V - U$. Now, $l(e) \leq l(e')$, since e is of minimum length among the edges connecting U with $V - U$. We can delete e' from $T_0 + e$. The resulting subgraph is still connected and by Theorem 2.2 is a tree, since it has the right number of edges. Also, the length of this new tree, which contains e, is less than or equal to that of T_0. Thus, it is optimal.

Q.E.D.

Let $G(V, E)$ be the given graph, where $V = \{1, 2, \ldots, n\}$. We assume that there are no parallel edges, for all but the shortest can be eliminated. Thus, let $l(i, j)$ be $l(e)$ if there is an edge $i \overset{e}{-} j$, and infinity otherwise. The following algorithm is due to Prim [1]:

(1) $t \leftarrow 1$, $T \leftarrow \emptyset$ and $U \leftarrow \{1\}$.
(2) Let $l(t, u) = \text{Min}_{v \in V-U}\{l(t, v)\}$.
(3) $T \leftarrow T \cup \{e\}$ where e is the edge which corresponds to the length $l(t, u)$.
(4) $U \leftarrow U \cup \{u\}$.
(5) If $U = V$, stop.
(6) For every $v \in V - U$, $l(t, v) \leftarrow \text{Min}\{l(t, v), l(u, v)\}$.
(7) Go to Step (2).

(Clearly $t = 1$ throughout. We used t instead of 1 to emphasize that $l(t, v)$ may not be the original $l(1, v)$ after Step (6) has been applied.)

The algorithm follows directly the hint supplied by Theorem 2.3. The "vertex" t represents the subset U of vertices, and for $v \in V - U$ $l(t, v)$ is the length of a shortest edge from a vertex in U to v. This is affected by Step (6). Thus, in Step (2), a shortest edge connecting U and $V - U$ is chosen.

Although each choice of an edge is "plausible", it is still necessary to prove that in the end, T is a minimum spanning tree.

Let a subgraph $G'(V', E')$ be called an *induced subgraph* if E' contains all the edges of E whose endpoints are in V'; in this case we say that G' is induced by V'.

First observe, that each time we reach Step (5), T is the edge set of a spanning tree of the subraph induced by U. This is easily proved by induction on the number of times we reach Step (5). We start with $U = \{1\}$ and $T = \emptyset$ which is clearly a spanning tree of the subgraph induced by $\{1\}$. After the first application of Steps (2), (3) and (4), we have two vertices in U and an edge in T which connects them. Each time we apply Steps (2), (3) and (4) we add an edge from a vertex of the previous U to a new vertex. Thus the new T is connected too. Also, the number of edges is one less than the number of vertices. Thus, by Theorem 2.2 (part (c)), T is a spanning tree.

Now, let us proceed by induction to prove that if the old T is a subgraph of some minimum spanning tree of G then so is the new one. The proof is similar to that of Theorem 2.3. Let T_0 be a minimum spanning tree of G which contains T as a subgraph, and assume e is the next edge chosen in Step (2) to connect between a vertex of U and $V - U$. If e is not in T_0, add it to T_0 to form $T_0 + e$. It contains a circuit in which there is one more edge, e', connecting a vertex of U with a vertex of $V - U$. By Step (2), $l(e) \leq l(e')$, and if we delete e' from $T_0 + e$, we get an minimum spanning tree which contains both T, as a subgraph, and e, proving that the new T is a subgraph of some minimum spanning tree. Thus, in the end T is a minimum spanning tree of G.

The complexity of the algorithm is $O(|V|^2)$; Step (2) requires at most $|V| - 1$ comparisons and is repeated $|V| - 1$ times, yielding $O(|V|^2)$. Step (6) requires one comparison for each edge; thus, the total time spent on it is $O(|E|)$.

It is possible to improve the algorithm and the interested reader is advised to read the Cheriton and Tarjan paper [2]. We do not pursue this here because an understanding of advanced data structures is necessary. The faster algorithms do not use any graph theory beyond the level of this section.

The analogous problem for diagraphs, namely, that of finding a subset of the edges E' whose total length is minimum among those for which (V, E') is a strongly connected subgraph, is much harder. In fact, even the case where $l(e) = 1$ for all edges is hard. This will be discussed in Chapter 10.

2.3 CAYLEY'S THEOREM

In a later section we shall consider the question of the number of spanning trees in a given graph. Here we consider the more restricted, and yet

interesting problem, of the number of trees one can define on a given set of vertices, $V = \{1, 2, \ldots, n\}$.

For $n = 3$, there are 3 possible trees, as shown in Figure 2.1. Clearly, for $n = 2$ there is only one tree. The reader can verify, by exhausting all the cases, that for $n = 4$ the number of trees is 16. The following theorem is due to Cayley [3]:

Theorem 2.4: The number of spanning trees for n distinct vertices is n^{n-2}.

The proof to be presented is due to Prüfer [4]. (For a survey of various proofs see Moon [5].)

Proof: Assume $V = \{1, 2, \ldots, n\}$. Let us display a one-to-one correspondence between the set of the spanning trees and the n^{n-2} words of length $n - 2$ over the alphabet $\{1, 2, \ldots, n\}$. The algorithm for finding the word which corresponds to a given tree is as follows:

(1) $i \leftarrow 1$.
(2) Among all leaves of the current tree let j be the least one (i.e., its name is the least integer). Eliminate j and its incident edge e from the tree. The ith letter of the word is the other endpoint of e.
(3) If $i = n - 2$, stop.
(4) Increment i and go to step 2.

For example, assume that $n = 6$ and the tree is as shown in Figure 2.2. On the first turn of Step (2), $j = 2$ and the other endpoint of its incident edge is 4. Thus, 4 is the first letter of the word. The new tree is as shown in Figure 2.3. On the second turn, $j = 3$ and the second letter is 1. On the third, $j = 1$ and the third letter is 6. On the fourth, $j = 5$ and the fourth letter is 4. Now $i = 4$ and the algorithm halts. The resulting word is 4164 (and the current tree consists of one edge connecting 4 and 6).

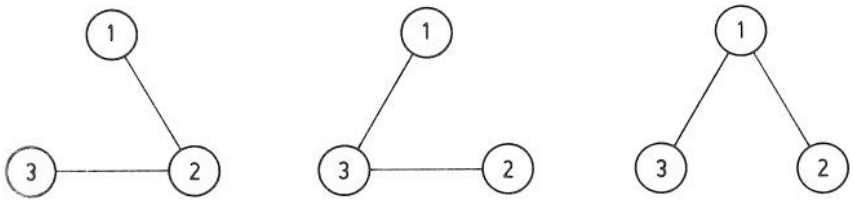

Figure 2.1

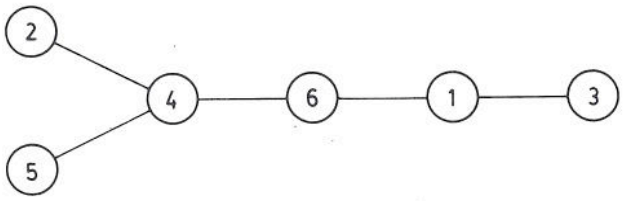

Figure 2.2

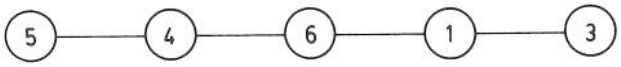

Figure 2.3

By Corollary 2.1, Step (2) can always be performed, and therefore for every tree a word of length $n - 2$ is produced. It remains to be shown that no word is produced by two different trees and that every word is generated from some tree. We shall achieve both ends by showing that the mapping has an inverse; i.e., for every word there is a unique tree which produces it.

Let $w = a_1 a_2 \ldots a_{n-2}$ be a word over V. If T is a tree for which the algorithm produces w then the degree of vertex k, $d(k)$, in T, is equal to the number of times k appears in w, plus 1. This follows from the observation that when each, but the last, of the edges incident to k is deleted, k is written as a letter of w; the last edge may never be deleted, if k is one of the two vertices remaining in the tree, or if it is deleted, k is now the removed leaf, and the adjacent vertex, not k, is the written letter. Thus, if w is produced by the algorithm, for some tree, then the degrees of the vertices in the tree must be as stated.

For example, if $w = 4164$ then $d(1) = 2$, $d(2) = 1$, $d(3) = 1$, $d(4) = 3$, $d(5) = 1$ and $d(6) = 2$ in a tree which produced w.

Given this data, apply the following algorithm:

(1) $i \leftarrow 1$.
(2) Let j be the least vertex for which $d(j) = 1$. Construct an edge j—a_i, $d(j) \leftarrow 0$ and $d(a_i) \leftarrow d(a_i) - 1$.
(3) If $i = n - 2$, construct an edge between the two vertices whose degree is 1 and stop.
(4) Increment i and go to step 2.

It is easy to see that this algorithm picks the same vertex j as the original algorithm, and constructs a tree (the proof is by induction). Also, each step

of the reconstruction is forced, therefore it is the only tree which yields w, and for every word this algorithm produces a tree.

In our example, for $i = 1, j = 2$ and since $a_1 = 4$ we connect 2—4, as shown in Figure 2.4. Now, $d(1) = 2$, $d(2) = 0$, $d(3) = 1$, $d(4) = 2$, $d(5) = 1$ and $d(6) = 2$. For $i = 2, j = 3$ and since $a_2 = 1$ we connect 3—1, as shown in Figure 2.5. Now $d(1) = 1$, $d(2) = 0$, $d(3) = 0$, $d(4) = 2$, $d(5) = 1$ and $d(6) = 2$. For $i = 3, j = 1$ and since $a_3 = 6$ we connect 1—6 as shown in Figure 2.6. Now, $d(1) = d(2) = d(3) = 0$, $d(4) = 2$ and $d(5) = d(6) = 1$. Finally, $i = 4, j = 5$ and since $a_4 = 4$ we connect 5—4, as shown in Figure 2.7. Now, $d(1) = d(2) = d(3) = d(5) = 0$ and $d(4) = d(6) = 1$. By step 3, we connect 4—6 and stop. The resulting graph is as in Figure 2.2.

Q.E.D.

A similar problem, stated and solved by Lempel and Welch [6], is that of finding the number of ways m labeled (distinct) edges can be joined by unlabeled endpoints to form a tree. Their proof is along the lines of Prüfer's proof of Cayley's theorem and is therefore constructive, in the sense that one can use the inverse transformation to generate all the trees after the words are generated. However, a much simpler proof was pointed out to me by A. Pnueli and is the subject of Problem 2.5.

Figure 2.4.

Figure 2.5.

Figure 2.6

Figure 2.7

2.4 DIRECTED TREE DEFINITIONS

A digraph $G(V, E)$ is said to have a *root* r if $r \in V$ and every vertex $v \in V$ is *reachable* from r; i.e., there is a directed path which starts in r and ends in v.

A digraph (finite or infinite) is called a *directed tree* if it has a root and its underlying undirected graph is a tree.

Theorem 2.5: Assume G is a digraph. The following five conditions are equivalent:

(a) G is a directed tree.
(b) G has a root from which there is a unique directed path to every vertex.
(c) G has a root r for which $d_{in}(r) = 0$ and for every other vertex v, $d_{in}(v) = 1$.
(d) G has a root and the deletion of any edge (but no vertices) interrupts this condition.
(e) The underlying undirected graph of G is connected and G has one vertex r for which $d_{in}(r) = 0$, while for every other vertex v, $d_{in}(v) = 1$.

Proof: We prove that (a) $\Rightarrow$ (b) $\Rightarrow$ (c) $\Rightarrow$ (d) $\Rightarrow$ (e) $\Rightarrow$ (a).

(a) $\Rightarrow$ (b): We assume that G has a root, say r, and its underlying undirected graph G' is a tree. Thus, by Theorem 2.1, part (c), there is a unique simple path from r to every vertex in G'; also, G' is circuit-free. Thus, a directed path from r to a vertex v, in G, must be simple and unique.

(b) $\Rightarrow$ (c): Here we assume that G has a root, say r, and a unique directed path from it to every vertex v. First, let us show that $d_{in}(r) = 0$. Assume there is an edge $u \xrightarrow{e} r$. There is a directed path from r to u, and it can be continued, via e, back to r. Thus, in addition to the empty path from r to itself (containing no edges), there is one more, in contradiction of the assumption of the path uniqueness. Now, we have to show that if $v \neq r$ then $d_{in}(v) = 1$. Clearly, $d_{in}(v) > 0$ for it must be reachable from r. If $d_{in}(v) > 1$, then there are at least two edges, say $v_1 \xrightarrow{e_1} v$ and $v_2 \xrightarrow{e_2} v$. Since there is a directed path P_1 from r to v_1, and a directed path P_2 from r to v_2, by adding e_1 to P_1 and e_2 to P_2 we get two different paths from r to v. (This proof is valid even if $v_1 = v_2$.)

(c) $\Rightarrow$ (d): This proof is trivial, for the deletion on any edge $u \xrightarrow{e} v$ will make v unreachable from r.

(d) ⇒ (e): We assume that G has a root, say r, and the deletion of any edge interrupts this condition. First $d_{in}(r) = 0$, for any edge entering r could be deleted without interrupting the condition that r is a root. For every other vertex v, $d_{in}(v) > 0$, for it is reachable from r. If $d_{in}(v) > 1$, let $v_1 \xrightarrow{e_1} v$ and $v_2 \xrightarrow{e_2} v$ be two edges entering v. Let P be a simple directed path from r to v. It cannot use both e_1 and e_2. The one which is not used in P can be deleted without interrupting the fact that r is a root. Thus, $d_{in}(v) = 1$.

(e) ⇒ (a): We assume that the underlying undirected graph of G, G', is connected, $d_{in}(r) = 0$ and for $v \neq r$, $d_{in}(v) = 1$. First let us prove that r is a root. Let P' be a simple path connecting r and v in G'. This must correspond to a directed path P from r to v in G, for if any of the edges points in the wrong direction it would either imply that $d_{in}(r) > 0$ or that for some u, $d_{in}(u) > 1$. Finally, G' must be circuit-free, for a simple circuit in G' must correspond to a simple directed circuit in G (again using $d_{in}(r) = 0$ and $d_{in}(v) = 1$ for $v \neq r$), and at least one of its vertices, u, must have $d_{in}(u) > 1$, since the vertices of the circuit are reachable from r.

Q.E.D.

In case of finite digraphs one more useful definition of a directed tree is possible:

Theorem 2.6: A finite digraph G is a directed tree if and only if its underlying undirected graph, G', is circuit-free, one of its vertices, r, satisfies $d_{in}(r) = 0$, and for all other vertices v, $d_{in}(v) = 1$.

Proof: The "only if" part follows directly from the definition of a directed tree and Theorem 2.5, part (c).

To prove the "if" part we first observe that the number of edges is $n - 1$. Thus, by Theorem 2.2, (b) ⇒ (c), G' is connected. Thus, by Theorem 2.5, (e) ⇒ (a), G is a directed tree.

Q.E.D.

Let us say that a digraph is *arbitrated* (Berge [7] calls it quasi strongly connected) if for every two vertices v_1 and v_2 there is a vertex v_1 called an *arbiter* of v_1 and v_2, such that there are directed paths from v to v_1 and from v to v_2. There are infinite digraphs which are arbitrated but do not have a root. For example, see the digraph of Figure 2.8. However, for finite digraphs the following theorem holds:

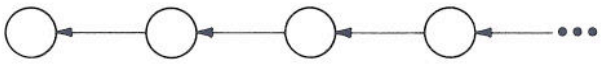

Figure 2.8

Theorem 2.7: If a finite digraph is arbitrated then it has a root.

Proof: Let $G(V, E)$ be a finite arbitrated digraph, where $V = \{1, 2, \ldots, n\}$. Let us prove, by induction, that every set $\{1, 2, \ldots, m\}$, where $m \leq n$, has an arbiter; i.e., a vertex a_m such that every $1 \leq i \leq m$ is reachable from a_m. By definition, a_2 exists. Assume a_{m-1} exists. Let a_m be the arbiter of a_{m-1} and m. Since a_{m-1} is reachable from a_m and every $1 \leq i \leq m - 1$ is reachable from a_{m-1}, every $1 \leq i \leq m - 1$ is also reachable from a_m.
Q.E.D.

Thus, for finite digraphs, the condition that it has a root, as in Theorem 2.5 part *a*, *b*, *c* and *d*, can be replaced by it being arbitrated.

2.5 THE INFINITY LEMMA

The following is known as König's Infinity Lemma [8]:

Theorem 2.8: If G is an infinite digraph, with a root r and finite out-degrees for all its vertices, then G has an infinite directed path, starting in r.

Before we present the proof let us point out the necessity of the finiteness of the out-degrees of the vertices. For if we allow a single vertex to be of infinite out-degree, the conclusion does not follow. Consider the digraph of Figure 2.9. The root is connected to vertices v_1^1, v_1^2, v_1^3, ..., where v_1^k is the second vertex on a directed path of length k. It is clear that the tree is infinite, and yet it has no infinite path. Furthermore, the replacement of the condition of finite degrees by the condition that for every k the tree has a path of length k, does not work either, as the same example shows.

Proof: First let us restrict our attention to a directed tree T which is an infinite subgraph of G. T's root is r. All vertices of distance 1 away from r in G are also of distance 1 away from r in T. In general, if a vertex v is of distance l away from r in G it is also of distance l away from r in T; all the edges entering v in G are now dropped, except one which connects a vertex of distance $l - 1$ to v. It is sufficient to show that in T there is an infinite directed path from r. Clearly, since T is a subgraph of G, all its vertices are of finite outdegrees too.

In T, r has infinitely many descendants (vertices reachable from r). Since r is of finite out-degree, at least one of its sons (the vertices reachable via one edge), say r_1, must have infinitely many descendants. One of r_1's sons

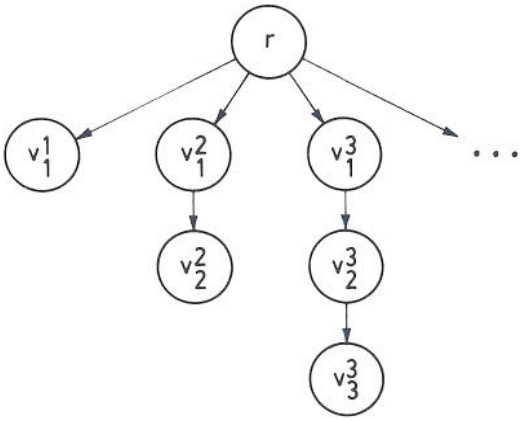

Figure 2.9

has infinitely many descendants, too, and so we continue to construct an infinite directed path $r, r_1, r_2, \ldots$.

Q.E.D.

In spite of the simplicity of the theorem, it is useful. For example, if we conduct a search on a directed tree of finite degrees (where a bound on the degree may not be known) for which it is known that it has no infinite directed paths, then the theorem ensures us that the tree is finite and our search will terminate.

An interesting application of Theorem 2.8 was made by Wang [9]. Consider the problem of tiling the plane with square tiles, all of the same size (Wang calls the tiles "dominoes"). There is a finite number of tile families. The sides of the tiles are labeled by letters of an alphabet, and all the tiles of one family have the same labels, thus are indistinguishable. Tiles may not be rotated or reflected, and the labels are specified for their north side, south side, and so on. There is an infinite supply of tiles of each family. The tiles may be put one next to another, the sides converging only if these two sides have the same labels. For example, if the tile families are as shown in Figure 2.10, then we can construct the "torus" shown in Figure 2.11. Now, by repeating this torus infinitely many times horizontally and vertically, we can tile the whole plane.

Wang proved that if it is possible to tile the upper right quadrant of the plane with a given finite set of tile families, then it is possible to tile the whole plane. The reader should realize that a southwest shift of the upper-right tiled quadrant cannot be used to cover the whole plane. In fact, if the number of tile families is not restricted to be finite, one can find sets

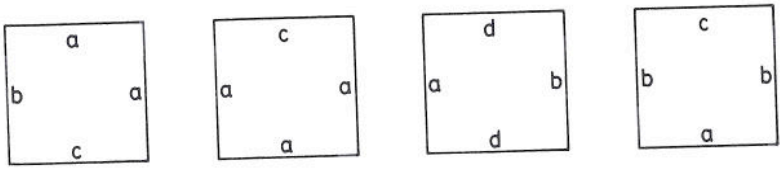

Figure 2.10

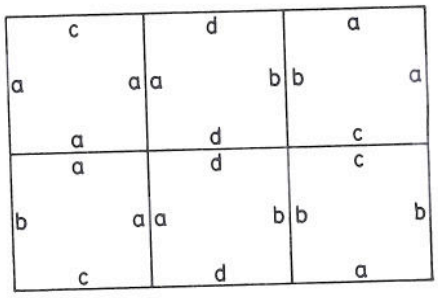

Figure 2.11

of families for which the upper-right quadrant is tileable, while the whole plane is not.

Consider the following directed tree T: The root r is connected to vertices, each representing one of the tile families, i.e., a square 1×1 tiled with the tile of that family. For every k, each one of the legitimate ways of tiling a $(2k + 1) \times (2k + 1)$ square is represented by a vertex in T; its father is the vertex which represents the tiling of a $(2k - 1) \times (2k - 1)$ square, identical to the center part of the square represented by the son.

Now, if the upper-right quadrant is tilable, then T has infinitely many vertices. Since the number of families is finite, the out-degree of each vertex is finite (although, may not be bounded). By Theorem 2.8, there is an infinite directed path in T. Such a path describes a way to tile the whole plane.

2.6 THE NUMBER OF SPANNING TREES

A subgraph H of a finite digraph G is called a *directed spanning tree* of G if H is a directed tree which includes all the vertices of G. If r is the root of H, then it is clearly a root of G. Also, if r is a root of G, then a spanning

directed tree H of G exists with root r. This is simply observed by constructing H, edge by edge, starting from r and adding each time an edge of G from a vertex already reachable from r in H to one which is not reachable yet.

We shall now describe a method of Tutte [10] for computing the number of spanning directed trees of a given digraph with a given specified root. (For historical details, see reference 11.)

Let us define the **in-degree matrix D** of a digraph $G(V, E)$, where $V = \{1, 2, \ldots, n\}$, as follows:

$$D(i, j) = \begin{cases} d_{\text{in}}(i) & \text{if } i = j, \\ -k & \text{if } i \neq j, \text{ where } k \text{ is the number of edges in } G \text{ from } i \text{ to } j. \end{cases}$$

Lemma 2.1: A finite digraph $G(V, E)$, with no self-loops is a directed tree with root r if and only if its in-degree matrix D has the following two properties:

(1) $$D(i, i) = \begin{cases} 0 & \text{if } i = r, \\ 1 & \text{if } i \neq r. \end{cases}$$

(2) The minor, resulting from erasing the rth row and column from D and computing the determinant, is 1.

Proof: Assume that $G(V, E)$ is a directed tree with root r. By Theorem 2.5, part c, D satisfies property (1). Now, renumber the vertices in such a way that 1 is the root and if $i \rightarrow j$ then $i < j$. This can be achieved by numbering the vertices of unit distance from 1 as 2, 3, ... Next number the vertices of distance two, three, etc. The new in-degree matrix is derivable from the previous one by performing some permutation on the rows, and the same permutation on the columns. Since such a permutation does not change the determinant, the two minors are the same. The new in-degree matrix D' satisfies the following properties:

$$\begin{aligned} &D'(1, 1) = 0, \\ &D'(i, i) = 1 \quad \text{for} \quad i = 2, 3, \ldots, n, \\ &D'(i, j) = 0 \quad \text{if} \quad i > j. \end{aligned}$$

Thus, the minor, resulting from the erasure of the first row and the first column from D' and computing the determinant, is 1.

Now assume that D satisfies properties (1) and (2). By property (1) and Theorem 2.6, if G is not a directed tree then its underlying undirected graph contains a simple circuit. The vertex r cannot be one of the vertices of the circuit, for this would imply that either $d_{in}(r) > 0$ or for some other vertex v $d_{in}(v) > 1$, contrary to property (1). The circuit must be of the form:

$$i_1 \rightarrow i_2 \rightarrow \cdots \rightarrow i_l \rightarrow i_1,$$

where l is the length of the circuit, and no vertex appears on it twice. Also, there may be other edges out of $i_1, i_2, \ldots, i_l$, but none can enter. Thus, each of the columns of D, corresponding to one on this vertices, has exactly one $+1$ (on the main diagonal of D) and one -1 and all the other entries are 0. Also, each of the rows of this submatrix is either all zeros, or there is one $+1$ and one -1. The sum of these columns is therefore a zero column, and thus, the minor is 0. This contradicts property (2).

Q.E.D.

As a side result of our proof, we have the additional property that the minor of a graph whose in-degree matrix satisfies property (1) is 0 if the graph is not a directed tree with root r.

Theorem 2.9: The number of directed spanning trees with root r of a digraph with no self-loops is given by the minor of its in-degree matrix which results from the erasure of the rth row and column.

The proof of this theorem follows immediately from Lemma 2.1, the comment following it, and the linearity of the determinant function with respect to its columns. Let us demonstrate this by the following example. Consider the graph shown in Figure 2.12. Its in-degree matrix D is as follows:

$$D = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & -2 \\ -1 & 0 & 3 \end{bmatrix}$$

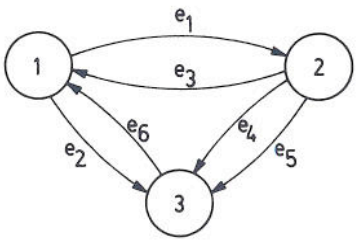

Figure 2.12

Assume that we want to compute the number of directed spanning trees with root 2. We erase the second row and column. The resulting determinant is

$$\begin{vmatrix} 2 & -1 \\ -1 & 3 \end{vmatrix} = 5$$

Now let us decompose this determinant into columns which represent one edge in every column. First, the 2×2 determinant can be written as

$$\begin{vmatrix} 2 & 0 & -1 \\ -1 & 1 & -2 \\ -1 & 0 & 3 \end{vmatrix}$$

We have returned the second row of D except its second entry, which must be made equal to 1 (in this case its value did not change). All other entries in the second column are changed into zero. Next, we decompose each column, except the second, into columns which consist of a single $+1$ and a single -1, as follows:

$$\begin{vmatrix} 2 & 0 & -1 \\ -1 & 1 & -2 \\ -1 & 0 & 3 \end{vmatrix} = \begin{vmatrix} 1 & 0 & -1 \\ -1 & 1 & -2 \\ 0 & 0 & 3 \end{vmatrix} + \begin{vmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ -1 & 0 & 3 \end{vmatrix}$$

$$= \begin{vmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} + \begin{vmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{vmatrix}$$

$$+ \begin{vmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{vmatrix} + \begin{vmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{vmatrix}$$

$$+ \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{vmatrix} + \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{vmatrix}$$

These six determinants correspond to the following selections of sets of edges, respectively: $\{e_3, e_2\}$, $\{e_3, e_4\}$, $\{e_3, e_5\}$, $\{e_6, e_2\}$, $\{e_6, e_4\}$, $\{e_6, e_5\}$. After erasing the second row and column, this corresponds to

$$\begin{vmatrix} 2 & -1 \\ -1 & 3 \end{vmatrix} = \begin{vmatrix} 1 & -1 \\ 0 & 1 \end{vmatrix} + \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} + \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

$$+ \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} + \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix} + \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix}$$

Each of these six determinants corresponds to a selection of $n - 1$ edges of the original graph. By Lemma 2.1, the resulting subgraph is a directed tree with root 2 if and only if the corresponding determinant is equal to one. Otherwise, it is zero. Thus, we get the number of directed trees with 2 as a root. Clearly, in our case, the only set which does not yield a directed tree is $\{e_6, e_2\}$ and indeed

$$\begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} = 0.$$

Q.E.D.

Let us now consider the question of the number of spanning trees of a given undirected graph $G(V, E)$. Consider the digraph $G'(V, E')$ defined as follows: For every edge $u \overset{e}{—} v$ in G define the two edges $u \overset{e'}{\rightarrow} v$ and $v \overset{e''}{\rightarrow} u$ in G'. Let r be a vertex. There is a one-one correspondence between the set of spanning trees of G and the set of directed spanning trees of G' with root r: Let T be a spanning tree of G. If the edge $u \overset{e}{—} v$ is in T and if u is closer than v to r in T then pick e' for T'; if v is closer, pick e''. Also, given T', it is easy to find the corresponding T by simply ignoring the directions; i.e., the existence of either e' or e'' in T' implies that e is in T. Thus, we can compute the number of spanning trees of G by writing the in-degree matrix of G', and computing the minor with respect to r. Clearly, the choice of r cannot make any difference. Now, the in-degree matrix of G' is given by

$$D(i, j) = \begin{cases} d(i) & \text{in } G \text{ if } i = j, \\ -k & \text{where } k \text{ is the number of edges connecting } i \text{ and } j \text{ in } G. \end{cases}$$

This matrix is called the *degree matrix* of G. Hence, we have the following theorem:

Theorem 2.10: The number of spanning trees of an undirected graph with no self-loops is equal to any of the minors of its degree matrix which results from the erasure of a row and a corresponding column.

We can now use Theorem 2.10 to describe another proof of Cayley's Theorem (2.4). The number of spanning trees that can be constructed with vertices $1, 2, \ldots, n$ is equal to the number of spanning trees of the *complete* graph of n vertices; that is, the graph $G(V, E)$ with $V = \{1, 2, \ldots, n\}$ and for every $i \neq j$ there is one edge i—j. Its degree matrix is

$$\begin{bmatrix} n-1 & -1 & \ldots & -1 \\ -1 & n-1 & & -1 \\ \vdots & & & \vdots \\ -1 & -1 & \ldots & n-1 \end{bmatrix}.$$

After erasing one row and the corresponding column, the matrix looks the same, except that it is now $(n-1) \times (n-1)$. We can now add to any

column (or row) a linear combination of the others, without changing its determinant. First subtract the first column from every other. We get:

$$\begin{bmatrix} n-1 & -n & -n & \dots & -n \\ -1 & n & 0 & & 0 \\ -1 & 0 & n & & 0 \\ \vdots & & & & \vdots \\ -1 & 0 & 0 & & n \end{bmatrix}$$

Now add every one of the other rows to the first:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & n & 0 & & 0 \\ -1 & 0 & n & & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ -1 & 0 & 0 & & n \end{bmatrix}$$

Clearly, the determinant of this matrix is n^{n-2}.

2.7 OPTIMUM BRANCHINGS AND DIRECTED SPANNING TREES

A subgraph $B(V, E')$ of a finite digraph $G(V, E)$ is called a *branching* if it is circuit-free and $d_{\text{in}}(v) \leq 1$ for every $v \in V$. Clearly, if for only one vertex r, $d_{\text{in}}(r) = 0$ and for all the rest of the vertices, v, $d_{\text{in}}(v) = 1$ then, by Theorem 2.6, the branching is a directed tree with root r.

Let each edge e have a cost $c(e)$. Our problem is to find a branching $B(V, E')$ for which the sum of the edge costs, $\Sigma_{e \in E'} c(e)$, is maximum. This problem was solved independently by a number of authors [12, 13, 14]. We shall follow here Karp's paper [15]. First, we will show how to find a maximum branching. Then, we shall point out the simple modification for finding a minimum branching and a minimum spanning tree.

Let us call an edge $u \xrightarrow{e} v$, of G, *critical* if

(i) $c(e) > 0$ and
(ii) for all other edges $u' \xrightarrow{e'} v$, $c(e) \geq c(e')$.

Let H be a set of critical edges, where for each vertex one entering critical edge is chosen, if any exist. The graph (V, H) is called *critical*.

Lemma 2.2: If a critical graph (V, H) is circuit-free then it is a maximum branching.

Proof: Clearly (V, H) is a branching if it is circuit-free. If vertex v has no positive edges entering it in G and if B is a branching, then either B has no edge entering v, or if it has one, we can drop it without reducing B's total cost. Clearly, H contains no edge which enters v either. If vertex v has positive edges which enter it in G, then the one in H is of maximum cost, and therefore no branching can do better here either. Since H is at least as good as B in each vertex, (V, H) is a maximum branching.

Q.E.D.

If a critical graph contains circuits then it is not a branching. Let us study some of its properties.

Lemma 2.3: Each vertex in a critical graph is on at most one circuit.

Proof: If a vertex v is on two directed circuits then there must be a vertex u for which $d_{in}(u) \geq 2$; a contradiction. Such a vertex can be found by tracing backwards on one of the circuits which passes through v.

Q.E.D.

Let $B(V, E')$ be a branching and $u \xrightarrow{e} v$ an edge not in B. Then e is *eligible* relative to B if the set

$$E'' = E' \cup \{e\} - \{e' \mid e' \in E' \text{ and it enters } v\}$$

yields a branching (V, E'').

Lemma 2.4: Let $B(V, E')$ be a branching and $e \in E - E'$. $u \xrightarrow{e} v$ is eligible relative to B if and only if there is no directed path in B from v to u.

Proof: If there is a directed path from v to u in B, then when we add e a directed circuit is formed. The deletion of the edge entering v in B, if any, cannot open this circuit. Thus, e is not eligible.

If there is no directed path from v to u in B, then the addition of e cannot close a directed circuit. However, the resulting edge set may not