3

Counting Problems in Graph Theory

Graph theory is a branch of discrete mathematics that studies networks composed of a number of sites (vertices) linked together by connecting arcs (edges). This chapter studies some enumeration problems that arise in graph theory. We begin by defining fundamental graph-theoretic concepts such as walks, paths, cycles, vertex degrees, connectivity, forests, and trees. This will lead to a discussion of various enumeration problems involving different kinds of trees. Aided by ideas from matrix theory, we will count walks in a graph, spanning trees of a graph, and Eulerian tours. We also investigate the chromatic polynomial of a graph, which counts the number of ways of coloring the vertices such that no two vertices joined by an edge receive the same color.

3.1 Graphs and Digraphs

Intuitively, a graph is a mathematical model for a network consisting of a collection of nodes and connections that link certain pairs of nodes. For example, the nodes could be cities and the connections could be roads between cities. For another example, the nodes could be computers and the connections could be network links between computers. For a third example, the nodes could be species in an ecosystem and the connections could be predator-prey relationships between species. Or the nodes could be tasks and the connections could be dependencies among the tasks. There is an unlimited variety of such applications that lead naturally to graph models. We now give the formal mathematical definitions underlying such models.

3.1. Definition: Graphs. A *graph* is an ordered triple $G = (V, E, \epsilon)$, where: $V = V(G)$ is a finite, nonempty set called the *vertex set* of G; $E = E(G)$ is a finite set called the *edge set* of G; and $\epsilon : E \to \mathcal{P}(V)$ is a function called the *endpoint function* such that, for all $e \in E$, $\epsilon(e)$ is either a one-element subset of V or a two-element subset of V. If $\epsilon(e) = \{v\}$, we call the edge e a *loop at vertex* v. If $\epsilon(e) = \{v, w\}$, we call v and w the *endpoints of* e and say that e is an edge *from* v *to* w. We also say that v and w are *adjacent in* G, v and w are *joined by* e, and e is *incident to* v and w.

We visualize a graph $G = (V, E, \epsilon)$ by drawing a collection of dots labeled by the elements $v \in V$. For each edge $e \in E$ with $\epsilon(e) = \{v, w\}$, we draw a line or curved arc labeled e between the two dots labeled v and w. Similarly, if $\epsilon(e) = \{v\}$, we draw a loop labeled e based at the dot labeled v.

3.2. Example. The left-hand drawing in Figure 3.1 represents the graph defined formally by the ordered triple

$$G_1 = (\{1, 2, 3, 4, 5\}, \{a, b, c, d, e, f, g, h, i\}, \epsilon),$$

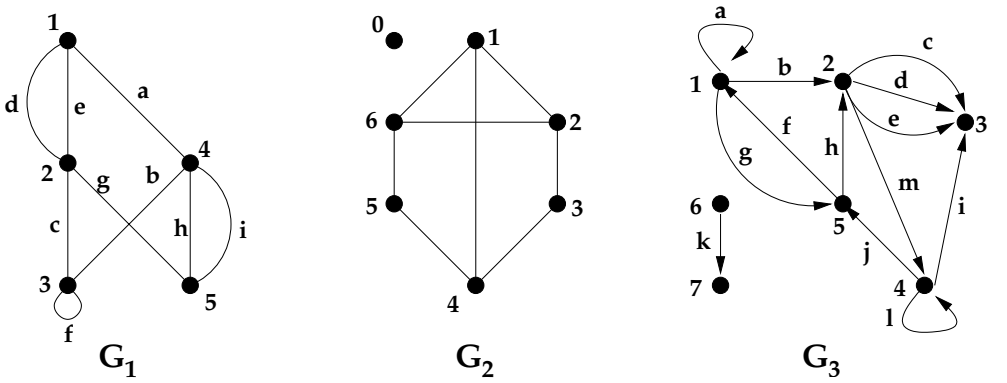


FIGURE 3.1
A graph, a simple graph, and a digraph.

where ϵ acts as follows:

$$a \mapsto \{1,4\}, \quad b \mapsto \{4,3\}, \quad c \mapsto \{2,3\}, \quad d \mapsto \{1,2\}, \quad e \mapsto \{1,2\},$$
$$f \mapsto \{3\}, \quad g \mapsto \{2,5\}, \quad h \mapsto \{4,5\}, \quad i \mapsto \{4,5\}.$$

Edge f is a loop at vertex 3; edges h and i both go between vertices 4 and 5; vertices 1 and 4 are adjacent, but vertices 2 and 4 are not.

In many applications, there are no loop edges, and there is never more than one edge between the same two vertices. This means that the endpoint function ϵ is a one-to-one map into the set of two-element subsets of V. So we can identify each edge e with its set of endpoints $\epsilon(e)$. This leads to the following simplified model in which edges are not explicitly named and there is no explicit endpoint function.

3.3. Definition: Simple Graphs. A *simple graph* is a pair $G = (V, E)$, where V is a finite nonempty set and E is a set of two-element subsets of V. We continue to use all the terminology introduced in 3.1.

3.4. Example. The central drawing in Figure 3.1 depicts the simple graph G_2 with vertex set $V(G_2) = \{0,1,2,3,4,5,6\}$ and edge set

$$E(G_2) = \{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\},\{1,6\},\{2,6\},\{1,4\}\}.$$

To model certain situations (such as predator-prey relationships, or one-way streets in a city), we need to introduce a *direction* on each edge. This leads to the notion of a digraph (directed graph).

3.5. Definition: Digraphs. A *digraph* is an ordered triple $D = (V, E, \epsilon)$, where V is a finite nonempty set of *vertices*, E is a finite set of *edges*, and $\epsilon : E \to V \times V$ is the *endpoint function*. If $\epsilon(e)$ is the *ordered pair* (v, w), we say that e is an edge *from* v *to* w.

In a digraph, an edge from v to w is not an edge from w to v when $v \neq w$, since $(v, w) \neq (w, v)$. On the other hand, in a graph, an edge from v to w is also an edge from w to v, since $\{v, w\} = \{w, v\}$.

3.6. Example. The right-hand drawing in Figure 3.1 displays a typical digraph. In this digraph, $\epsilon(j) = (4,5)$, $\epsilon(a) = (1,1)$, and so on. There are three edges from 2 to 3, but no edges from 3 to 2. There are edges in both directions between vertices 1 and 5.

As before, we can eliminate specific reference to the endpoint function of a digraph if there are no multiple edges with the same starting vertex and ending vertex.

3.7. Definition: Simple Digraphs. A *simple digraph* is an ordered pair $D = (V, E)$, where V is a finite, nonempty set and E is a subset of $V \times V$. Each ordered pair $(v, w) \in E$ represents an edge in D from v to w. Note that we do allow loops ($v = w$) in a simple digraph.

When investigating structural properties of graphs, the names of the vertices and edges are often irrelevant. The concept of graph isomorphism lets us identify graphs that are "the same" except for the names used for the vertices and edges.

3.8. Definition: Graph Isomorphism. Given two graphs $G = (V, E, \epsilon)$ and $H = (W, F, \eta)$, a *graph isomorphism* from G to H consists of two bijections $f : V \to W$ and $g : E \to F$ such that, for all $e \in E$, if $\epsilon(e) = \{v, w\}$ then $\eta(g(e)) = \{f(v), f(w)\}$ (we allow $v = w$ here). Digraph isomorphisms are defined similarly: $\epsilon(e) = (v, w)$ implies $\eta(g(e)) = (f(v), f(w))$. We say G and H are *isomorphic*, written $G \cong H$, iff there exists a graph isomorphism from G to H.

In the case of *simple* graphs $G = (V, E)$ and $H = (W, F)$, a graph isomorphism can be viewed as a bijection $f : V \to W$ that induces a bijection between the edge sets E and F. More specifically, this means that for all $v, w \in V$, $\{v, w\} \in E$ iff $\{f(v), f(w)\} \in F$.

3.2 Walks and Matrices

We can travel through a graph by following a succession of edges. Formalizing this idea leads to the concept of a walk.

3.9. Definition: Walks, Paths, Cycles. Let $G = (V, E, \epsilon)$ be a graph or digraph. A *walk in G* is a sequence

$$W = (v_0, e_1, v_1, e_2, v_2, \ldots, e_s, v_s)$$

where $s \geq 0$, $v_i \in V$ for all i, $e_i \in E$ for all i, and e_i is an edge from v_{i-1} to v_i for $1 \leq i \leq s$. We say that W is a walk *of length s from v_0 to v_s*. The walk W is *closed* iff $v_0 = v_s$. The walk W is a *path* iff the vertices $v_0, v_1, \ldots, v_s$ are pairwise distinct (which forces the edges e_i to be distinct as well). The walk W is a *cycle* iff $s > 0$, $v_1, \ldots, v_s$ are distinct, $e_1, \ldots, e_s$ are distinct, and $v_0 = v_s$. A *k-cycle* is a cycle of length k. In the case of simple graphs and simple digraphs, the edges e_i are determined uniquely by their endpoints. So, in the simple case, we can regard a walk as a sequence of vertices $(v_0, v_1, \ldots, v_s)$ such that there is an edge from v_{i-1} to v_i in G for $1 \leq i \leq s$.

3.10. Remark. When considering cycles in a digraph, we usually *identify* two cycles that are cyclic shifts of one another (unless we need to keep track of the starting vertex of the cycle). Similarly, we identify cycles in a graph that are cyclic shifts or reversals of one another.

3.11. Example. In the graph G_1 from Figure 3.1,

$$W_1 = (2, c, 3, f, 3, b, 4, h, 5, i, 4, i, 5)$$

is a walk of length 6 from vertex 2 to vertex 5. In the simple graph G_2 in the same figure,

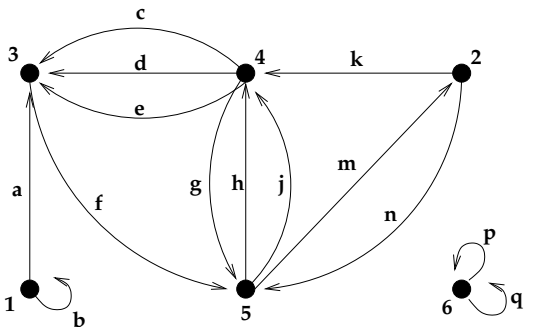


FIGURE 3.2
Digraph used to illustrate adjacency matrices.

$W_2 = (1, 6, 2, 3, 4, 5)$ is a walk and a path of length 5, whereas $C = (6, 5, 4, 3, 2, 6)$ is a 5-cycle. We usually identify C with the cycles $(5, 4, 3, 2, 6, 5)$, $(6, 2, 3, 4, 5, 6)$, etc. In the digraph G_3,

$$W_3 = (1, a, 1, g, 5, h, 2, m, 4, j, 5, h, 2, d, 3)$$

is a walk from vertex 1 to vertex 3; $(5, h, 2, m, 4, j, 5)$ is a 3-cycle; $(4, l, 4)$ is a 1-cycle; and $(5, f, 1, g, 5)$ is a 2-cycle. Observe that 1-cycles are the same as loop edges, and 2-cycles cannot exist in simple graphs or simple digraphs. For any vertex v in a graph or digraph, (v) is a walk of length zero from v to v, which is a path but not a cycle.

We can now formulate our first counting problem: how many walks of a given length are there between two given vertices in a graph or digraph? We will develop an algebraic solution to this problem in which concatenation of walks is modeled by multiplication of suitable matrices.

3.12. Definition: Adjacency Matrix. Let G be a graph or digraph with vertex set $X = \{x_i : 1 \leq i \leq n\}$. The *adjacency matrix of* G (relative to the given indexing of the vertices) is the $n \times n$ matrix A whose i, j-entry $A(i, j)$ is the number of edges in G from x_i to x_j.

3.13. Example. The adjacency matrix for the digraph G in Figure 3.2 is

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}.$$

3.14. Example. If G is a graph, edges from v to w are the same as edges from w to v. So, the adjacency matrix for G is a *symmetric* matrix ($A(i, j) = A(j, i)$ for all i, j). If G is a simple graph, the adjacency matrix consists of all 1's and 0's with zeroes on the main diagonal. For example, the adjacency matrix of the simple graph in Figure 3.3 is

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

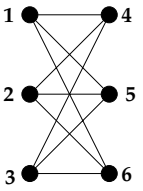


FIGURE 3.3
A simple graph.

Recall from linear algebra the definition of the product of two matrices.

3.15. Definition: Matrix Multiplication. Suppose A is an $m \times n$ matrix and B is an $n \times p$ matrix. Then AB is the $m \times p$ matrix whose i, j-entry is

$$(AB)(i,j) = \sum_{k=1}^{n} A(i,k)B(k,j) \qquad (1 \le i \le m, 1 \le j \le p). \tag{3.1}$$

Matrix multiplication is associative, so we can write a product of three or more (compatible) matrices without any parentheses. The next theorem gives a formula for the general entry in such a product.

3.16. Theorem: Product of Several Matrices. Assume $A_1, \ldots, A_s$ are matrices such that A_i has dimensions $n_{i-1} \times n_i$. Then $A_1 A_2 \cdots A_s$ is the $n_0 \times n_s$ matrix whose k_0, k_s-entry is

$$(A_1 A_2 \cdots A_s)(k_0, k_s) = \sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \cdots \sum_{k_{s-1}=1}^{n_{s-1}} A_1(k_0,k_1)A_2(k_1,k_2)A_3(k_2,k_3)\cdots A_s(k_{s-1},k_s) \tag{3.2}$$

for all $1 \le k_0 \le n_0, 1 \le k_s \le n_s$.

Proof. We use induction on s. The case $s = 1$ is immediate, and the case $s = 2$ is the definition of matrix multiplication (after a change in notation). Assume $s > 2$ and that (3.2) is known to hold for the product $B = A_1 A_2 \cdots A_{s-1}$. We can think of the given product $A_1 A_2 \cdots A_{s-1} A_s$ as the binary product BA_s. Therefore, using (3.1), the k_0, k_s-entry of $A_1 A_2 \cdots A_s$ is

$$\begin{aligned}
(BA_s)(k_0,k_s) &= \sum_{k=1}^{n_{s-1}} B(k_0,k)A_s(k,k_s) \\
&= \sum_{k=1}^{n_{s-1}} \left(\sum_{k_1=1}^{n_1} \cdots \sum_{k_{s-2}=1}^{n_{s-2}} A_1(k_0,k_1)A_2(k_1,k_2)\cdots A_{s-1}(k_{s-2},k) \right) A_s(k,k_s) \\
&= \sum_{k_1=1}^{n_1} \cdots \sum_{k_{s-2}=1}^{n_{s-2}} \sum_{k=1}^{n_{s-1}} A_1(k_0,k_1)A_2(k_1,k_2)\cdots A_{s-1}(k_{s-2},k)A_s(k,k_s).
\end{aligned}$$

Replacing k by k_{s-1} in the innermost summation, we obtain the result. □

Taking all A_i's in the theorem to be the same matrix A, we obtain the following formula for the entries of the powers of a given square matrix.

3.17. Corollary: Powers of a Matrix. Suppose A is an $n \times n$ matrix. For each integer $s > 0$,

$$A^s(i,j) = \sum_{k_1=1}^{n} \cdots \sum_{k_{s-1}=1}^{n} A(i,k_1)A(k_1,k_2)\cdots A(k_{s-2},k_{s-1})A(k_{s-1},j) \qquad (1 \le i,j \le n). \tag{3.3}$$

The preceding formula may appear unwieldy. But it is precisely the tool we need to count walks in graphs.

3.18. Theorem: Enumeration of Walks. Let G be a graph or digraph with vertex set $X = \{x_1, \ldots, x_n\}$, and let A be the adjacency matrix of G. For all $i, j \le n$ and all $s \ge 0$, the i,j-entry of A^s is the number of walks of length s in G from x_i to x_j.

Proof. The result holds for $s = 0$, since $A^0 = I_n$ (the $n \times n$ identity matrix) and there is exactly one walk of length zero from any vertex to itself. Now suppose $s > 0$. A walk of length s from x_i to x_j will visit $s-1$ intermediate vertices (not necessarily distinct from each other or from x_i or x_j). Let $(x_i, x_{k_1}, \ldots, x_{k_{s-1}}, x_j)$ be the ordered list of vertices visited by the walk. To build such a walk, we choose any edge from x_i to x_{k_1} in $A(i,k_1)$ ways; then we choose any edge from x_{k_1} to x_{k_2} in $A(k_1,k_2)$ ways; and so on. By the product rule, the total number of walks associated to this vertex sequence is $A(i,k_1)A(k_1,k_2)\cdots A(k_{s-1},j)$. This formula holds even if there are no walks with this vertex sequence, since some term in the product will be zero in this case. Applying the sum rule produces the right side of (3.3), and the result follows. □

3.19. Example. Consider again the adjacency matrix A of the digraph G in Figure 3.2. Some matrix computations show that

$$A^2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 3 & 2 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 2 & 3 & 0 \\ 0 & 0 & 6 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix}, \qquad A^3 = \begin{bmatrix} 1 & 1 & 1 & 2 & 1 & 0 \\ 0 & 1 & 6 & 3 & 6 & 0 \\ 0 & 0 & 6 & 1 & 3 & 0 \\ 0 & 3 & 6 & 7 & 3 & 0 \\ 0 & 3 & 3 & 6 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}.$$

So, for example, there are six walks of length 2 from vertex 5 to vertex 3, and there are seven walks of length 3 that start and end at vertex 4.

3.3 DAGs and Nilpotent Matrices

Next we consider the question of counting *all* walks (of any length) between two vertices in a digraph. The question is uninteresting for graphs, since the number of walks between two distinct vertices v, w in a graph is either zero or infinity. This follows since a walk is allowed to repeatedly traverse a particular edge along a path from v to w, which leads to arbitrarily long walks from v to w. Similarly, if G is a digraph that contains a cycle, we obtain arbitrarily long walks between two vertices on the cycle by going around the cycle again and again. To rule out these possibilities, we restrict attention to the following class of digraphs.

3.20. Definition: DAGs. A *DAG* is a digraph with no cycles. (The acronym DAG stands for "directed acyclic graph.")

To characterize adjacency matrices of DAGs, we need another concept from matrix theory.

3.21. Definition: Nilpotent Matrices. An $n \times n$ matrix A is called *nilpotent* iff $A^s = 0$ for some integer $s \geq 1$. The least such integer s is called the *index of nilpotence* of A.

Note that if $A^s = 0$ and $t \geq s$, then $A^t = 0$ also.

3.22. Example. The zero matrix is the unique $n \times n$ matrix with index of nilpotence 1. The square of the nonzero matrix $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ is zero, so A is nilpotent of index 2. Similarly, any matrix

$$B = \begin{bmatrix} 0 & x & y \\ 0 & 0 & z \\ 0 & 0 & 0 \end{bmatrix} \qquad (x, y, z \in \mathbb{R})$$

satisfies

$$B^2 = \begin{bmatrix} 0 & 0 & xz \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B^3 = 0,$$

so we obtain examples of matrices that are nilpotent of index 3. The next result generalizes this example.

3.23. Theorem: Nilpotence of Strictly Triangular Matrices. Suppose A is an $n \times n$ *strictly upper-triangular* matrix, which means $A(i, j) = 0$ for all $i \geq j$. Then A is nilpotent of index at most n. A similar result holds for strictly lower-triangular matrices.

Proof. It suffices to show that A^n is the zero matrix. By (3.3), we have

$$A^n(k_0, k_n) = \sum_{k_1=1}^{n} \cdots \sum_{k_{n-1}=1}^{n} A(k_0, k_1)A(k_1, k_2) \cdots A(k_{n-1}, k_n)$$

for all $k_0, k_n \leq n$. We claim that each term in this sum is zero. Otherwise, there would exist $k_1, \ldots, k_{n-1}$ such that $A(k_{t-1}, k_t) \neq 0$ for $1 \leq t \leq n$. But since A is strictly upper-triangular, we would then have

$$k_0 < k_1 < k_2 < \cdots < k_n.$$

This cannot occur, since all the k_t's are integers between 1 and n. □

The next theorem reveals the connection between nilpotent matrices and DAGs.

3.24. Theorem: DAGs and Nilpotent Matrices. Let G be a digraph with vertex set $X = \{x_1, \ldots, x_n\}$ and adjacency matrix A. G is a DAG iff A is nilpotent. When G is a DAG, there exists an ordering of the vertex set X that makes A a strictly lower-triangular matrix.

Proof. Assume first that G is not a DAG, so that G has at least one cycle. Let x_i be any fixed vertex involved in this cycle, and let $c \geq 1$ be the length of this cycle. By going around the cycle zero or more times, we obtain walks from x_i to x_i of lengths $0, c, 2c, 3c, \ldots$. By 3.18, it follows that the (i, i)-entry of A^{kc} is at least 1, for all $k \geq 0$. This fact prevents any positive power of A from being the zero matrix, so A is not nilpotent.

Conversely, assume that A is not nilpotent. Then, in particular, $A^n \neq 0$, so there exist indices k_0, k_n with $A^n(k_0, k_n) \neq 0$. Using 3.18 again, we deduce that there is a walk in G visiting a sequence of vertices

$$(x_{k_0}, x_{k_1}, x_{k_2}, \ldots, x_{k_{n-1}}, x_{k_n}).$$

Since G has only n vertices, not all of the $n+1$ vertices just listed are distinct. If we choose i minimal and then $j > i$ minimal such that $x_{k_i} = x_{k_j}$, then there is a cycle in G visiting the vertices $(x_{k_i}, x_{k_{i+1}}, \ldots, x_{k_j})$. So G is not a DAG.

We prove the statement about lower-triangular matrices by induction on n. A one-vertex DAG must have adjacency matrix (0), so the result holds for $n = 1$. Suppose $n > 1$ and the result is known for DAGs with $n-1$ vertices. Create a walk $(v_0, e_1, v_1, e_2, v_2, \ldots)$ in G by starting at any vertex and repeatedly following any edge leading away from the current vertex. Since G has no cycles, the vertices v_i reached by this walk are pairwise distinct. Since there are only n available vertices, our walk must terminate at a vertex v_j with no outgoing edges. Let $x'_1 = v_j$. Deleting x'_1 and all edges leading into this vertex will produce an $(n-1)$-vertex digraph G' that is also a DAG, as one immediately verifies. By induction, there is an ordering $x'_2, \ldots, x'_n$ of the vertices of G' such that the associated adjacency matrix A' is strictly lower-triangular. Now, relative to the ordering $x'_1, x'_2, \ldots, x'_n$ of the vertices of G, the adjacency matrix of G has the form

$$\left[\begin{array}{c|c} 0 & 0 \ \cdots \ 0 \\ \hline * & \\ \vdots & A' \\ * & \end{array}\right],$$

and this matrix is strictly lower-triangular. □

The next result will allow us to count walks of any length in a DAG.

3.25. Theorem: Inverse of $I - A$ for Nilpotent A. Suppose A is a nilpotent $n \times n$ matrix with $A^s = 0$. Let I be the $n \times n$ identity matrix. Then $I - A$ is an invertible matrix with inverse

$$(I - A)^{-1} = I + A + A^2 + A^3 + \cdots + A^{s-1}.$$

Proof. Let $B = I + A + A^2 + \cdots + A^{s-1}$. By the distributive law,

$$(I - A)B = IB - AB = (I + A + A^2 + \cdots + A^{s-1}) - (A + A^2 + \cdots + A^{s-1} + A^s) = I - A^s.$$

Since $A^s = 0$, we see that $(I - A)B = I$. A similar calculation shows that $B(I - A) = I$. Therefore B is the two-sided matrix inverse of $I - A$. □

3.26. Remark. The previous result for matrices can be remembered by noting the analogy to the geometric series formula for real numbers:

$$(1 - r)^{-1} = \frac{1}{1 - r} = 1 + r + r^2 + \cdots + r^{s-1} + r^s + \cdots \qquad (|r| < 1).$$

3.27. Theorem: Counting Paths in a DAG. Let G be a DAG with vertex set $\{x_1, \ldots, x_n\}$ and adjacency matrix A. For all $i, j \leq n$, the total number of paths from x_i to x_j in G (of any length) is the i, j-entry of $(I - A)^{-1}$.

Proof. By 3.18, the number of walks of length $t \geq 0$ from x_i to x_j is $A^t(i, j)$. Because G is a DAG, we have $A^t = 0$ for all $t \geq n$. By the sum rule, the total number of walks from x_i to x_j is $\sum_{t=0}^{n-1} A^t(i, j)$. By 3.25, this number is precisely the i, j-entry of $(I - A)^{-1}$. Finally, one readily confirms that every walk in a DAG must actually be a path. □

3.28. Example. Consider the DAG shown in Figure 3.4. Its adjacency matrix is

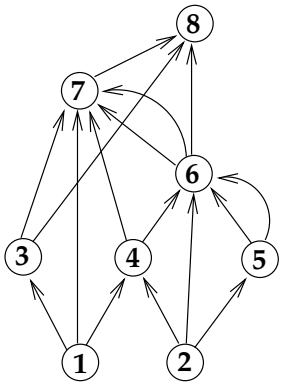


FIGURE 3.4
Example of a DAG.

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

Using a computer algebra system, we compute

$$(I - A)^{-1} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 5 & 7 \\ 0 & 1 & 0 & 1 & 1 & 4 & 9 & 13 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 0 & 1 & 3 & 4 \\ 0 & 0 & 0 & 0 & 1 & 2 & 4 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

So, for example, there are 13 paths from vertex 2 to vertex 8, and 4 paths from vertex 5 to vertex 7.

3.4 Vertex Degrees

In many situations, one needs to know how many edges lead into or out of each vertex in a digraph.

3.29. Definition: Indegree and Outdegree. Let $G = (V, E, \epsilon)$ be a digraph. For each $v \in V$, the *outdegree* of v, denoted $\text{outdeg}_G(v)$, is the number of edges $e \in E$ *from* v; the *indegree* of v, denoted $\text{indeg}_G(v)$, is the number of edges $e \in E$ *to* v. Formally,

$$\text{outdeg}_G(v) = \sum_{w \in V} \sum_{e \in E} \chi(\epsilon(e) = (v, w)); \quad \text{indeg}_G(v) = \sum_{w \in V} \sum_{e \in E} \chi(\epsilon(e) = (w, v)).$$

A *source* is a vertex of indegree zero. A *sink* is a vertex of outdegree zero.

3.30. Example. Let G_3 be the digraph on the right in Figure 3.1. We have

$$\begin{aligned}(\text{indeg}_{G_3}(1), \ldots, \text{indeg}_{G_3}(7)) &= (2,2,4,2,2,0,1);\\ (\text{outdeg}_{G_3}(1), \ldots, \text{outdeg}_{G_3}(7)) &= (3,4,0,3,2,1,0).\end{aligned}$$

Vertex 6 is the only source, whereas vertices 3 and 7 are sinks. A loop edge at v contributes 1 to both the indegree and outdegree of v. The sum of all the indegrees is 13, which is also the sum of all the outdegrees, and is also the number of edges in the digraph. This phenomenon is explained in the next theorem.

3.31. Theorem: Degree-Sum Formula for Digraphs. In any digraph $G = (V, E, \epsilon)$,

$$\sum_{v \in V} \text{indeg}_G(v) = |E| = \sum_{v \in V} \text{outdeg}_G(v).$$

Proof. By the formal definition of indegree, we have

$$\begin{aligned}\sum_{v \in V} \text{indeg}_G(v) &= \sum_{v \in V} \sum_{w \in V} \sum_{e \in E} \chi(\epsilon(e) = (w, v))\\ &= \sum_{e \in E} \left(\sum_{w \in V} \sum_{v \in V} \chi(\epsilon(e) = (w, v)) \right)\end{aligned}$$

For each $e \in E$, the term in brackets is equal to one for exactly one ordered pair (w, v), and is zero otherwise. So the sum evaluates to $\sum_{e \in E} 1 = |E|$. The formula involving outdegree is proved similarly. □

Next we give analogous definitions and results for graphs.

3.32. Definition: Degree. Let $G = (V, E, \epsilon)$ be a graph. For each $v \in V$, the *degree of v in G*, denoted $\deg_G(v)$, is the number of edges in E with v as an endpoint, where each loop edge at v is counted twice. Formally,

$$\deg_G(v) = \sum_{e \in E} 2\chi(\epsilon(e) = \{v\}) + \sum_{\substack{w \in V \\ w \neq v}} \sum_{e \in E} \chi(\epsilon(e) = \{v, w\}).$$

The *degree sequence of G*, denoted $\deg(G)$, is the multiset $[\deg_G(v) : v \in V]$. G is called *k-regular* iff every vertex in G has degree k. G is *regular* iff G is k-regular for some $k \geq 0$.

3.33. Example. For the graph G_1 in Figure 3.1, we have

$$(\deg_G(1), \ldots, \deg_G(5)) = (3,4,4,4,3); \qquad \deg(G) = [4,4,4,3,3].$$

The graph in Figure 3.3 is 3-regular. In both of these graphs, the sum of all vertex degrees is 18, which is twice the number of edges in the graph.

3.34. Theorem: Degree-Sum Formula for Graphs. For any graph $G = (V, E, \epsilon)$,

$$\sum_{v \in V} \deg_G(v) = 2|E|.$$

Proof. First assume G has no loop edges. Let X be the set of pairs (v, e) such that $v \in V$, $e \in E$, and v is an endpoint of e. On one hand,

$$|X| = \sum_{v \in V} \sum_{e \in E} \chi((v, e) \in X) = \sum_{v \in V} \deg_G(v).$$

On the other hand,

$$|X| = \sum_{e \in E} \sum_{v \in V} \chi((v, e) \in X) = \sum_{e \in E} 2 = 2|E|$$

since every edge has two distinct endpoints. So the result holds in this case.

Next, if G has k loop edges, let G' be G with these loops deleted. Then

$$\sum_{v \in V} \deg_G(v) = \sum_{v \in V} \deg_{G'}(v) + 2k,$$

since each loop edge increases some vertex degree in the sum by 2. Using the result for loopless graphs,

$$\sum_{v \in V} \deg_G(v) = 2|E(G')| + 2k = 2|E(G)|,$$

since G has k more edges than G'. □

Vertices of low degree are given special names.

3.35. Definition: Isolated Vertices. An *isolated vertex* in a graph is a vertex of degree zero.

3.36. Definition: Leaves. A *leaf* is a vertex of degree one.

The following result will be used later in our analysis of trees.

3.37. Two-Leaf Lemma. Suppose G is a graph. One of the following three alternatives must occur: (i) G has a cycle; (ii) G has no edges; or (iii) G has at least two leaves.

Proof. Suppose that G has no cycles and G has at least one edge; we prove that G has two leaves. Since G has no cycles, we can assume G is simple. Let $P = (v_0, v_1, \ldots, v_s)$ be a path of maximum length in G. Such a path exists, since G has only finitely many vertices and edges. Observe that $s > 0$ since G has an edge, and $v_0 \neq v_s$. Note that $\deg(v_s) \geq 1$ since $s > 0$. Assume v_s is not a leaf. Then there exists a vertex $w \neq v_{s-1}$ that is adjacent to v_s. Now, w is different from all v_j with $0 \leq j < s - 1$, since otherwise $(v_j, v_{j+1}, \ldots, v_s, w = v_j)$ would be a cycle in G. But this means $(v_0, v_1, \ldots, v_s, w)$ is a path in G longer than P, contradicting maximality of P. So v_s must be a leaf. A similar argument shows that v_0 is also a leaf. □

3.5 Functional Digraphs

We can obtain structural information about functions $f : V \to V$ by viewing these functions as certain kinds of digraphs.

3.38. Definition: Functional Digraphs. A *functional digraph on V* is a simple digraph G with vertex set V such that $\text{outdeg}_G(v) = 1$ for all $v \in V$.

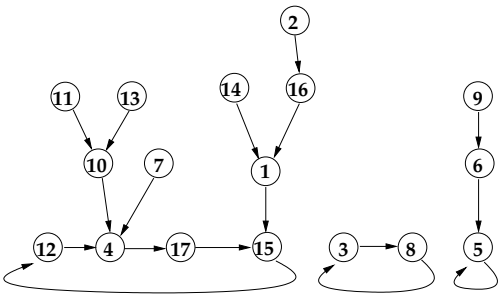


FIGURE 3.5
A functional digraph.

A function $f : V \to V$ can be thought of as a set E_f of ordered pairs such that for each $x \in V$, there exists exactly one $y \in V$ with $(x, y) \in E_f$, namely $y = f(x)$. Then (V, E_f) is a functional digraph on V. Conversely, a functional digraph $G = (V, E)$ determines a unique function $g : V \to V$ by letting $g(v)$ be the other endpoint of the unique edge in E departing from v. These comments establish a bijection between the set of functions on V and the set of functional digraphs on V.

3.39. Example. Figure 3.5 displays the functional digraph associated to the following function:

$$\begin{array}{lllll} f(1) = 15; & f(2) = 16; & f(3) = 8; & f(4) = 17; & f(5) = 5; \\ f(6) = 5; & f(7) = 4; & f(8) = 3; & f(9) = 6; & f(10) = 4; \\ f(11) = 10; & f(12) = 4; & f(13) = 10; & f(14) = 1; & f(15) = 12; \\ f(16) = 1; & f(17) = 15. & & & \end{array}$$

We wish to understand the structure of functional digraphs. Consider the digraph $G = (V, E)$ shown in Figure 3.5. Some of the vertices in this digraph are involved in cycles, which are drawn at the bottom of the figure. These cycles have length one or greater, and any two distinct cycles involve disjoint sets of vertices. The other vertices in the digraph all feed into these cycles at different points. We can form a set partition of the vertex set of the digraph by collecting together all vertices that feed into a particular vertex on a particular cycle. Each such collection can be viewed as a smaller digraph that has no cycles. We will show that these observations hold for all functional digraphs. To do this, we need a few more defintions.

3.40. Definition: Cyclic Vertices. Let G be a functional digraph on V. A vertex $v \in V$ is called *cyclic* iff v belongs to some cycle of G; otherwise, v is called *acyclic*.

3.41. Example. The cyclic elements for the functional digraph in Figure 3.5 are 3, 4, 5, 8, 12, 15, and 17.

Let $f : V \to V$ be the function associated to the functional digraph G. Then $v \in V$ is cyclic iff $f^i(v) = v$ for some $i \geq 1$, where f^i denotes the composition of f with itself i times. This fact follows since the only possible cycle involving v in G must look like $(v, f(v), f^2(v), f^3(v), \ldots)$.

3.42. Definition: Rooted Trees. A digraph G is called a *rooted tree with root* v_0 iff G is a functional digraph and v_0 is the unique cyclic vertex of G.

3.43. Theorem: Structure of Functional Digraphs. Let G be a functional digraph on V with associated function $f : V \to V$. Let $C \subseteq V$ denote the set of cyclic vertices of G. C is nonempty, and each $v \in C$ belongs to exactly one cycle of G. Also, there exists a unique indexed set partition $\{S_v : v \in C\}$ of V such that the following hold for all $v \in C$: (i) $v \in S_v$; (ii) $x \in S_v$ and $x \neq v$ implies $f(x) \in S_v$; (iii) if $g : S_v \to S_v$ is defined by $g(x) = f(x)$ for $x \neq v$, $g(v) = v$, then the functional digraph of g is a rooted tree with root v.

Proof. First, suppose $v \in C$. Since every vertex of G has exactly one outgoing edge, the only possible cycle involving v must be $(v, f(v), f^2(v), \ldots, f^i(v) = v)$. So each cyclic vertex (if any) belongs to a unique cycle of G. This implies that distinct cycles of G involve disjoint sets of vertices and edges.

Next we define a surjection $r : V \to C$. The existence of r will show that $C \neq \emptyset$, since $V \neq \emptyset$. Fix $u \in V$. By repeatedly following outgoing arrows, we obtain for each $k \geq 0$ a unique walk $(u = u_0, u_1, u_2, \ldots, u_k)$ in G of length k. Since V is finite, there must exist $i < j$ with $u_i = u_j$. Take i minimal and then j minimal with this property; then $(u_i, u_{i+1}, \ldots, u_j)$ is a cycle in G. We define $r(u) = u_i$, which is the first element on this cycle reached from u. One may check that $r(u) = u$ for all $u \in C$; this implies that r is surjective. On the other hand, if $u \notin C$, the definition of r shows that $r(u) = r(u_1) = r(f(u))$.

How shall we construct a set partition with the stated properties? For each $v \in C$, consider the "fiber" $S_v = r^{-1}(\{v\}) = \{w \in V : r(w) = v\}$; then $\{S_v : v \in C\}$ is a set partition of V indexed by C. The remarks at the end of the last paragraph show that this set partition satisfies (i) and (ii). To check (iii) for some $v \in C$, first note that the map g defined in (iii) does map S_v into S_v by (i) and (ii). Suppose $W = (w_0, w_1, \ldots, w_k)$ is a cycle in the functional digraph for g. Since $r(w_0) = v$, we will eventually reach v by following outgoing arrows starting at w_0. On the other hand, following these arrows keeps us on the cycle W, so some $w_i = v$. Since $g(v) = v$, the only possibility is that W is the 1-cycle (v). Thus (iii) holds for each $v \in C$.

To see that $\{S_v : v \in C\}$ is unique, let $\mathcal{P} = \{T_v : v \in C\}$ be another set partition with properties (i), (ii), and (iii). It is enough to show that $S_v \subseteq T_v$ for each $v \in C$. Fix $v \in C$ and $z \in S_v$. By (ii), every element in the sequence $(z, f(z), f^2(z), \ldots)$ belongs to the same set of $\mathcal{P}$, say T_w. Then $v = r(z) = f^i(z) \in T_w$, so (i) forces $w = v$. Thus $z \in T_v$ as desired. □

We can informally summarize the previous result by saying that *every functional digraph uniquely decomposes into disjoint rooted trees feeding into one or more disjoint cycles.* There are two extreme cases of this decomposition that are especially interesting — the case where there are no trees (i.e., $C = V$), and the case where the whole digraph is a rooted tree (i.e., $|C| = 1$). We study these types of functional digraphs in the next two sections.

3.6 Cycle Structure of Permutations

The functional digraph of a bijection (permutation) has a particularly nice structure.

3.44. Example. Figure 3.6 displays the digraph associated to the following bijection:

$$\begin{array}{lllll} h(1) = 7; & h(2) = 8; & h(3) = 4; & h(4) = 3; & h(5) = 10; \\ h(6) = 2; & h(7) = 5; & h(8) = 6; & h(9) = 9; & h(10) = 1. \end{array}$$

We see that the digraph for h contains only cycles; there are no trees feeding into these cycles. To see why this happens, compare this digraph to the digraph for the non-bijective

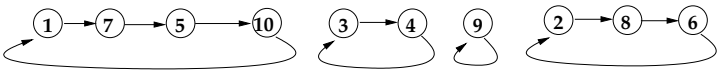


FIGURE 3.6
Digraph associated to a permutation.

function f in Figure 3.5. The digraph for f has a rooted tree feeding into the cyclic vertex 15. Accordingly, f is not injective since $f(17) = 15 = f(1)$. Similarly, if we move backwards through the trees in the digraph of f, we reach vertices with indegree zero (namely 2, 7, 9, 11, 13, and 14). The existence of such vertices shows that f is not surjective. Returning to the digraph of h, consider what happens if we reverse the direction of all the edges in the digraph. We obtain another functional digraph corresponding to the following function:

$$\begin{array}{lllll} h'(1) = 10; & h'(2) = 6; & h'(3) = 4; & h'(4) = 3; & h'(5) = 7; \\ h'(6) = 8; & h'(7) = 1; & h'(8) = 2; & h'(9) = 9; & h'(10) = 5. \end{array}$$

One sees immediately that h' is the two-sided inverse for h.

The next theorem explains the observations in the last example.

3.45. Theorem: Cycle Decomposition of Permutations. Let $f : V \to V$ be a function with functional digraph G. The map f is a bijection iff every $v \in V$ is a cyclic vertex in V. In this situation, G is a disjoint union of cycles.

Proof. Suppose $u \in V$ is a non-cyclic vertex. By 3.43, u belongs to a rooted tree S_v whose root v belongs to a cycle of G. Following edges outward from u will eventually lead to v; let y be the vertex in S_v just before v on this path. Let z be the vertex just before v in the unique cycle involving v. We have $y \neq z$, but $f(y) = v = f(z)$. Thus, f is not injective.

Conversely, suppose all vertices in V are cyclic. Then the digraph G is a disjoint union of directed cycles. So every $v \in V$ has indegree 1 as well as outdegree 1. Reversing the direction of every edge in G therefore produces another *functional* digraph G'. Let $f' : V \to V$ be the function associated to this new digraph. For all $a, b \in V$, we have $b = f(a)$ iff $(a, b) \in G$ iff $(b, a) \in G'$ iff $a = f'(b)$. It follows that f' is the two-sided inverse for f, so that f and f' are both bijections. □

Recall that $S(n, k)$, the Stirling number of the second kind, is the number of set partitions of an n-element set into k blocks. Let $c(n, k)$ be the number of permutations of an n-element set whose functional digraph consists of k disjoint cycles. We will show that $c(n, k) = s'(n, k)$, the signless Stirling number of the first kind. Recall from 2.66 that the numbers $s'(n, k)$ satisfy the recursion $s'(n, k) = s'(n - 1, k - 1) + (n - 1)s'(n - 1, k)$ for $0 < k < n$, with initial conditions $s'(n, 0) = \chi(n = 0)$ and $s'(n, n) = 1$.

3.46. Theorem: Recursion for $c(n, k)$. We have $c(n, 0) = \chi(n = 0)$ and $c(n, n) = 1$ for all $n \geq 0$. For $0 < k < n$, we have

$$c(n, k) = c(n - 1, k - 1) + (n - 1)c(n - 1, k).$$

Therefore, $c(n, k) = s'(n, k)$ for all $0 \leq k \leq n$.

Proof. The identity map is the unique permutation of an n-element set with n cycles (which must each have length 1), so $c(n, n) = 1$. The only permutation with zero cycles is the empty function on the empty set, so $c(n, 0) = \chi(n = 0)$. Now suppose $0 < k < n$. Let A, B, C be

the sets of permutations counted by $c(n,k)$, $c(n-1,k-1)$, and $c(n-1,k)$, respectively. Note that A is the disjoint union of the two sets

$$A_1 = \{f \in A : f(n) = n\} \text{ and } A_2 = \{f \in A : f(n) \neq n\}.$$

For each $f \in A_1$, we can restrict f to the domain $\{1,2,\ldots,n-1\}$ to obtain a permutation of these $n-1$ elements. Since f has k cycles, one of which involves n alone, the restriction of f must have $k-1$ cycles. Since $f \in A_1$ is uniquely determined by its restriction to $\{1,2,\ldots,n-1\}$, we have a bijection from A_1 onto B.

On the other hand, let us build a typical element $f \in A_2$ by making two choices. First, choose a permutation $g \in C$ in $c(n-1,k)$ ways. Second, choose an element $i \in \{1,2,\ldots,n-1\}$ in $n-1$ ways. Let j be the unique number such that $g(j) = i$. Modify the digraph for g by removing the arrow from j to i and replacing it by an arrow from j to n and an arrow from n to i. Informally, we are splicing n into the cycle just before i. Let f be the permutation associated to the new digraph. Evidently, the splicing process does not change the number of cycles of g, and f satisfies $f(n) \neq n$. Thus, $f \in A_2$, and every element of A_2 arises uniquely by the choice process we have described. By the sum and product rules,

$$c(n,k) = |A| = |A_1| + |A_2| = c(n-1,k-1) + (n-1)c(n-1,k).$$

So $c(n,k)$ and $s'(n,k)$ satisfy the same recursion and initial conditions. A routine induction argument now shows that $c(n,k) = s'(n,k)$ for all n and k. □

3.7 Counting Rooted Trees

Our goal in this section is to count rooted trees (see 3.42) with a fixed root vertex.

3.47. Theorem: Enumeration of Rooted Trees. For all $n > 1$, there are n^{n-2} rooted trees on the vertex set $\{1,2,\ldots,n\}$ with root 1.

Proof. Let B be the set of rooted trees mentioned in the theorem. Let A be the set of all functions $f : \{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$ such that $f(1) = 1$ and $f(n) = n$. The product rule shows that $|A| = n^{n-2}$. It therefore suffices to define maps $\phi : A \to B$ and $\phi' : B \to A$ that are mutual inverses. To define ϕ, fix $f \in A$. Let $G_f = (\{1,2,\ldots,n\}, \{(i,f(i)) : 1 \leq i \leq n\})$ be the functional digraph associated with f. By 3.43, we can decompose the vertex set $\{1,2,\ldots,n\}$ of this digraph into some disjoint cycles $C_0, C_1, \ldots, C_k$ and (possibly) some trees feeding into these cycles. For $0 \leq i \leq k$, let ℓ_i be the largest vertex in cycle C_i, and write $C_i = (r_i, \ldots, \ell_i)$. We can choose the indexing of the cycles so that the numbers ℓ_i satisfy $\ell_0 > \ell_1 > \ell_2 > \cdots > \ell_k$. Since $f(1) = 1$ and $f(n) = n$, 1 and n belong to cycles of length 1, so that $\ell_0 = r_0 = n$, $C_0 = (n)$, $\ell_k = r_k = 1$, $C_k = (1)$, and $k > 0$. To obtain $\phi(f)$, modify the digraph G_f by removing all edges of the form (ℓ_i, r_i) and adding new edges (ℓ_i, r_{i+1}), for $0 \leq i < k$. One may check that $\phi(f)$ is always a rooted tree with root 1.

3.48. Example. Suppose $n = 20$ and f is the function defined as follows:

$$\begin{array}{lllll} f(1)=1; & f(2)=19; & f(3)=8; & f(4)=17; & f(5)=5; \\ f(6)=5; & f(7)=4; & f(8)=3; & f(9)=6; & f(10)=1; \\ f(11)=18; & f(12)=4; & f(13)=18; & f(14)=20; & f(15)=15; \\ f(16)=1; & f(17)=12; & f(18)=4; & f(19)=20; & f(20)=20. \end{array}$$

We draw the digraph of f in such a way that all vertices involved in cycles occur in a horizontal row at the bottom of the figure, and the largest element in each cycle is the rightmost

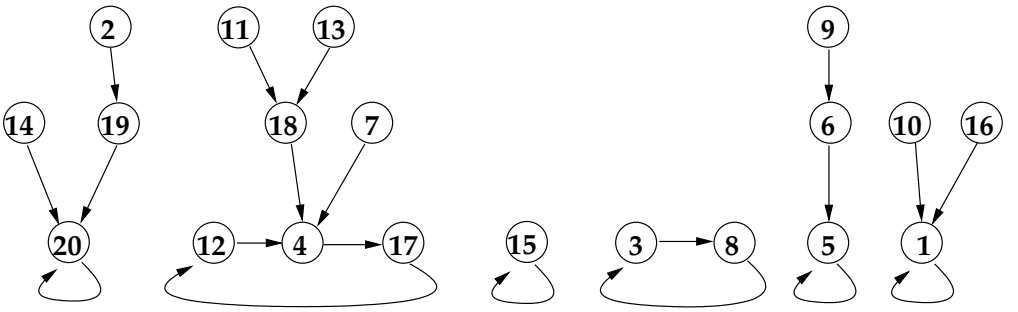


FIGURE 3.7
A functional digraph with cycles arranged in canonical order.

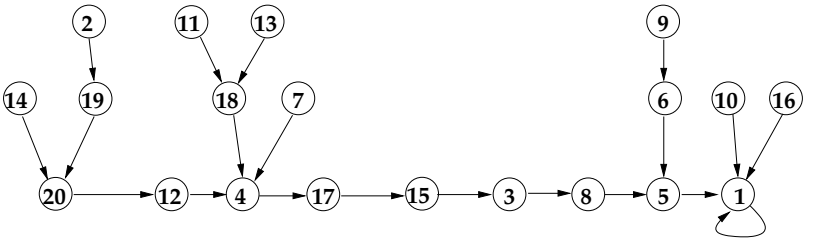


FIGURE 3.8
Conversion of the digraph to a rooted tree.

element of its cycle. We arrange these cycles so that these largest elements decrease from left to right; in particular, vertex n is always at the far left, and vertex 1 at the far right. See Figure 3.7. To compute $\phi(f)$, we cut the "back-edges" leading left from ℓ_i to r_i (which are loops if $\ell_i = r_i$) and add new edges leading right from ℓ_i to r_{i+1}. See Figure 3.8.

Continuing the proof, let us see why ϕ is invertible. Let T be a rooted tree on $\{1, 2, \ldots, n\}$ with root 1. Following outgoing edges from n must eventually lead to the unique cyclic vertex 1. Let $P = (v_0, v_1, \ldots, v_s)$ be the vertices encountered on the way from $v_0 = n$ to $v_s = 1$. We recursively recover the numbers $\ell_0, \ell_1, \ldots, \ell_k$ as follows. Let $\ell_0 = n$. Define ℓ_1 to be the largest number in P following ℓ_0. In general, after ℓ_{i-1} has been found, define ℓ_i to be the largest number in P following ℓ_{i-1}. After finitely many steps, we will get $\ell_k = 1$ for some k. Next, let $r_0 = n$, and for $i > 0$, let r_i be the vertex immediately following ℓ_{i-1} on the path P. Modify T by deleting the edges (ℓ_i, r_{i+1}) and adding edges of the form (ℓ_i, r_i), for $0 \le i < k$. One can verify that every vertex in the resulting digraph G' has outdegree exactly 1, and there are loop edges in G' at vertex 1 and vertex n. Thus, G' is a functional digraph that determines a function $f = \phi'(T) \in A$. It follows from the definition of ϕ' that ϕ' is the two-sided inverse of ϕ. □

3.49. Example. Suppose $n = 9$ and T is the rooted tree shown on the left in Figure 3.9. We first redraw the picture of T so that the vertices on the path P from n to 1 occur in a horizontal row at the bottom of the picture, with n on the left and 1 on the right. We recover ℓ_i and r_i by the procedure above, and then delete the appropriate edges of T and add appropriate back-edges to create cycles. The resulting functional digraph appears on

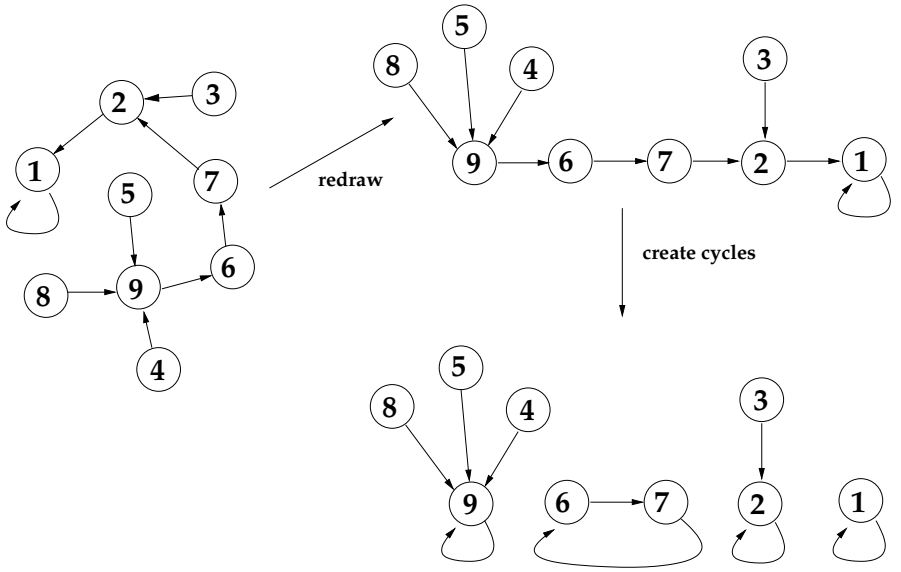


FIGURE 3.9
Conversion of a rooted tree to a functional digraph.

the bottom right in Figure 3.9. So $\phi'(T)$ is the function g defined as follows:

$$g(1) = 1; \quad g(2) = 2; \quad g(3) = 2; \quad g(4) = 9; \quad g(5) = 9;$$
$$g(6) = 7; \quad g(7) = 6; \quad g(8) = 9; \quad g(9) = 9.$$

3.8 Connectedness and Components

In many applications of graphs, it is important to know whether every vertex is reachable from every other vertex.

3.50. Definition: Connectedness. Let $G = (V, E, \epsilon)$ be a graph or digraph. G is *connected* iff for all $u, v \in V$, there is a walk in G from u to v.

3.51. Example. The graph G_1 in Figure 3.1 is connected, but the simple graph G_2 and the digraph G_3 in that figure are not connected.

Connectedness can also be described using paths instead of walks.

3.52. Theorem: Walks vs. Paths. Let $G = (V, E, \epsilon)$ be a graph or digraph, and let $u, v \in V$. There is a walk in G from u to v iff there is a path in G from u to v.

Proof. Let $W = (v_0, e_1, v_1, \ldots, e_s, v_s)$ be a walk in G from u to v. We describe an algorithm to convert the walk W into a path from u to v. If all the vertices v_i are distinct, then the edges e_i must also be distinct, so W is already a path. Otherwise, choose i minimal such that v_i appears more than once in W, and then choose j maximal such that $v_i = v_j$. Then $W_1 = (v_0, e_1, v_1, \ldots, e_i, v_i, e_{j+1}, v_{j+1}, \ldots, e_s, v_s)$ is a walk from u to v of shorter length than

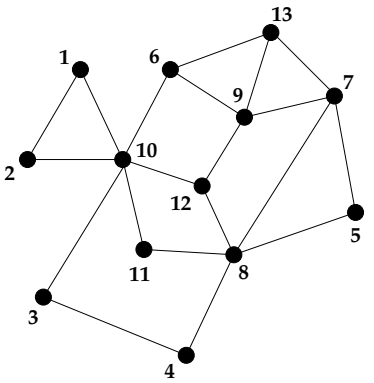


FIGURE 3.10
Converting a walk to a path.

W. If W_1 is a path, we are done. Otherwise, we repeat the argument to obtain a walk W_2 from u to v that is shorter than W_1. Since the lengths keep decreasing, this process must eventually terminate with a path W_k from u to v. (W_k has length zero if $u = v$.) The converse is immediate, since every path in G from u to v is a walk in G from u to v. □

3.53. Example. In the simple graph shown in Figure 3.10, consider the walk

$$W = (11, 10, 1, 2, 10, 3, 4, 8, 11, 8, 12, 10, 6, 9, 7, 13, 9, 12, 8, 5).$$

First, the repetition $v_0 = 11 = v_8$ leads to the walk

$$W_1 = (11, 8, 12, 10, 6, 9, 7, 13, 9, 12, 8, 5).$$

Eliminating the multiple visits to vertex 8 leads to the walk

$$W_2 = (11, 8, 5).$$

W_2 is a path from 11 to 5.

3.54. Corollary: Connectedness and Paths. A graph or digraph $G = (V, E, \epsilon)$ is connected iff for all $u, v \in V$, there is *at least one path* in G from u to v.

By looking at pictures of graphs, it becomes visually evident that any graph decomposes into a disjoint union of connected pieces, with no edge joining vertices in two separate pieces. These pieces are called the (connected) components of the graph. The situation for digraphs is more complicated, since there may exist directed edges between different components. To give a formal development of these ideas, we introduce the following equivalence relation.

3.55. Definition: Interconnection Relation. Let $G = (V, E, \epsilon)$ be a graph or digraph. Define a binary relation $\leftrightarrow_G$ on the vertex set V by setting $u \leftrightarrow_G v$ iff there exist walks in G from u to v and from v to u.

In the case of *graphs*, note that $u \leftrightarrow_G w$ iff there is a walk in G from u to w, since the reversal of such a walk is a walk in G from w to u. Now, for a graph or digraph G, let us verify that $\leftrightarrow_G$ is indeed an equivalence relation on V. First, for all $u \in V$, (u) is a walk of length 0 from u to u, so $u \leftrightarrow_G u$ and $\leftrightarrow_G$ is reflexive on V. Second, the symmetry of $\leftrightarrow_G$ is automatic from the way we defined $\leftrightarrow_G$: $u \leftrightarrow_G v$ implies $v \leftrightarrow_G u$ for all $u, v \in V$. Finally,

to check transitivity, suppose $u, v, w \in V$ satisfy $u \leftrightarrow_G v$ and $v \leftrightarrow_G w$. Let W_1, W_2, W_3, and W_4 be walks in G from u to v, from v to w, from v to u, and from w to v, respectively. Then the concatenation of W_1 followed by W_2 is a walk in G from u to w, whereas the concatenation of W_4 followed by W_3 is a walk in G from w to u. Hence $u \leftrightarrow_G w$, as desired.

3.56. Definition: Components. Let $G = (V, E, \epsilon)$ be a graph or digraph. The *components* of G are the equivalence classes of the interconnection equivalence relation $\leftrightarrow_G$. Components are also called *connected components* or (in the case of digraphs) *strong components*.

Since $\leftrightarrow_G$ is an equivalence relation on V, the components of G form a set partition of the vertex set V. Given a component C of G, consider the graph or digraph (C, E', ϵ') obtained by retaining those edges in E with both endpoints in C and restricting ϵ to this set of edges. One may check that this graph or digraph is connected.

3.57. Example. The components of the graph G_2 in Figure 3.1 are $\{0\}$ and $\{1, 2, 3, 4, 5, 6\}$. The components of the digraph G_3 in that figure are $\{1, 2, 4, 5\}$, $\{3\}$, $\{6\}$, and $\{7\}$.

The next theorems describe how the addition or deletion of an edge affects the components of a graph.

3.58. Theorem: Edge Deletion and Components. Let $G = (V, E, \epsilon)$ be a graph with components $\{C_i : i \in I\}$. Let $e \in E$ be an edge with endpoints $v, w \in C_j$. Let $G' = (V, E', \epsilon')$ where $E' = E \sim \{e\}$ and ϵ' is the restriction of ϵ to E'.
(a) If e appears in some cycle of G, then G and G' have the same components.
(b) If e appears in no cycle of G, then G' has one more component than G. More precisely, the components of G' are the C_k with $k \neq j$, together with two disjoint sets A and B such that $A \cup B = C_j$, $v \in A$, and $w \in B$.

Proof. For (a), let $(v_0, e_1, v_1, e_2, \ldots, v_s)$ be a cycle of G containing e. Cyclically shifting and reversing the cycle if needed, we can assume $v_0 = v = v_s$, $e_1 = e$, and $v_1 = w$. Statement (a) will follow if we can show that the interconnection relations $\leftrightarrow_G$ and $\leftrightarrow_{G'}$ coincide. First, for all $y, z \in V$, $y \leftrightarrow_{G'} z$ implies $y \leftrightarrow_G z$ since every walk in the smaller graph G' is also a walk in G. On the other hand, does $y \leftrightarrow_G z$ imply $y \leftrightarrow_{G'} z$? We know there is a walk W from y to z in G. If W does not use the edge e, W is a walk from y to z in G'. Otherwise, we can modify W as follows. Every time W goes from $v = v_s = v_0$ to $w = v_1$ via e, replace this part of the walk by the sequence $(v_s, e_s, \ldots, e_2, v_1)$ obtained by taking a detour around the cycle. Make a similar modification each time W goes from w to v via e. This produces a walk in G' from y to z.

For (b), let us compute the equivalence classes of $\leftrightarrow_{G'}$. First, fix $z \in C_k$ where $k \neq j$. The set C_k consists of all vertices in V reachable from z by walks in G. One readily checks that none of these walks can use the edge e, so C_k is also the set of all vertices in V reachable from z by walks in G'. So C_k is the equivalence class of z relative to both $\leftrightarrow_G$ and $\leftrightarrow_{G'}$.

Next, let A and B be the equivalence classes of v and w (respectively) relative to $\leftrightarrow_{G'}$. By definition, A and B are two of the components of G' (possibly the same component). We now show that A and B are disjoint and that their union is C_j. If the equivalence classes A and B are not disjoint, then they must be equal. By 3.52, there must be a path $(v_0, e_1, v_1, \ldots, e_s, v_s)$ in G' from v to w. Appending e, v_0 to this path would produce a cycle in G involving the edge e, which is a contradiction. Thus A and B are disjoint. Let us show that $A \cup B \subseteq C_j$. If $z \in A$, then there is a walk in G' (and hence in G) from v to z. Since C_j is the equivalence class of v relative to $\leftrightarrow_G$, it follows that $z \in C_j$. Similarly, $z \in B$ implies $z \in C_j$ since C_j is also the equivalence class of w relative to $\leftrightarrow_G$. Next, we check that $C_j \subseteq A \cup B$. Let $z \in C_j$, and let $W = (w_0, e_1, w_1, \ldots, w_t)$ be a walk in G from v to z. If W does not use the edge e, then $z \in A$. If W does use e, then the portion of W following the

last appearance of the edge e is a walk from either v or w to z in G'; thus $z \in A \cup B$. Since the union of A, B, and the C_k with $k \neq j$ is all of V, we have found all the components of G'. □

The previous result suggests the following terminology.

3.59. Definition: Cut-Edges. An edge e in a graph G is a *cut-edge* iff e does not appear in any cycle of G.

3.60. Theorem: Edge Addition and Components. Let $G = (V, E, \epsilon)$ be a graph with components $\{C_i : i \in I\}$. Let $G^+ = (V, E^+, \epsilon^+)$ be the graph obtained from G by adding a new edge e with endpoints $v \in C_j$ and $w \in C_k$.
(a) If v and w are in the same component C_j of G, then e is involved in a cycle of G^+, and G and G^+ have the same components.
(b) If v and w are in different components of G, then e is a cut-edge of G^+, and the components of G^+ are $C_j \cup C_k$ and the C_i with $i \neq j, k$.

This theorem follows readily from 3.58, so we leave the proof to the reader.

3.9 Forests

3.61. Definition: Forests. A *forest* is a graph with no cycles. Such a graph is also called *acyclic*.

A forest cannot have any loops or multiple edges between the same two vertices. So we can assume, with no loss of generality, that forests are *simple* graphs.

3.62. Example. Figure 3.11 displays a forest.

Recall from 3.54 that a graph G is connected iff there exists *at least one* path between any two vertices of G. The next result gives an analogous characterization of forests.

3.63. Theorem: Forests and Paths. A graph G is acyclic iff G has no loops and for all u, v in $V(G)$, there is *at most one* path from u to v in G.

Proof. We prove the contrapositive in both directions. First suppose that G has a cycle $C = (v_0, e_1, v_1, \ldots, e_s, v_s)$. If $s = 1$, G has a loop. If $s > 1$, then $(v_1, e_2, \ldots, e_s, v_s)$ and

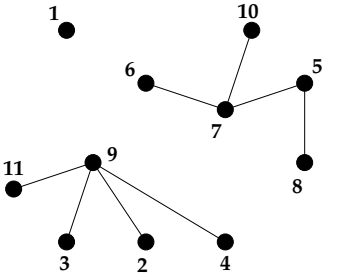


FIGURE 3.11
A forest.

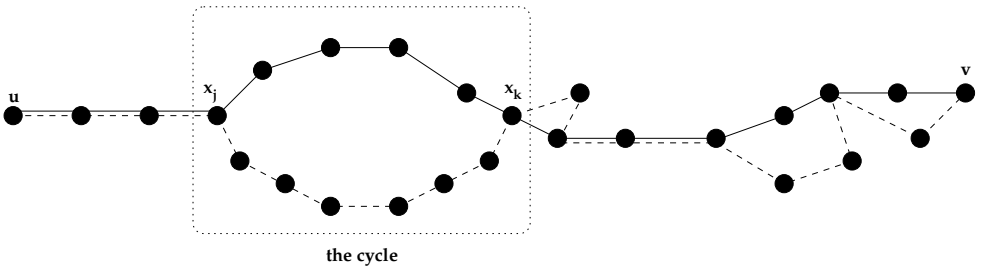


FIGURE 3.12
Extracting a cycle from two paths.

(v_1, e_1, v_0) are two distinct paths in G from v_1 to v_0. For the converse, we can assume G is simple. Suppose u and v are vertices of G and $P = (x_0, x_1, \ldots, x_s)$, $Q = (y_0, y_1, \ldots, y_t)$ are two distinct paths in G from u to v, where all x_i's and y_j's are in $V(G)$. We will use these paths to construct a cycle in G. Note that the concatenation of P and the reversal of Q is a walk in G that starts and ends at u. But this walk need not be a cycle, since it may involve repeated edges or vertices.

Since P and Q are distinct, there is an index j such that $x_i = y_i$ for $0 \leq i \leq j$, but $x_{j+1} \neq y_{j+1}$. Since P and Q both end at v, there must exist a least index $k \geq j+1$ such that x_k is a vertex in Q, say $x_k = y_r$. It follows from the choice of j and k that either $k = j+1$ and $r > j+1$, or $k > j+1$ and $r \geq j+1$. In any case,

$$C = (x_j, x_{j+1}, \ldots, x_k = y_r, y_{r-1}, \ldots, y_{j+1}, y_j)$$

is a cycle in G. Figure 3.12 illustrates this argument. □

The following result gives a formula for the number of components in a forest.

3.64. Theorem: Components of a Forest. Let G be a forest with n vertices and k edges. The number of connected components of G is $n - k$.

Proof. We use induction on k. The result holds for $k = 0$, since G consists of n isolated vertices in this case. Assume that $k > 0$ and the result is already known for forests with n vertices and $k-1$ edges. Given a forest G with n vertices and k edges, remove one edge e from G to get a new graph H. The graph H is acyclic and has n vertices and $k-1$ edges. By induction, H has $n - (k-1) = n - k + 1$ components. On the other hand, e must be a cut-edge since G has no cycles. It follows from 3.58 that H has one more component than G. Thus, G has $n - k$ components, as desired. □

3.10 Trees

3.65. Definition: Trees. A *tree* is a connected graph with no cycles.

3.66. Example. Figure 3.13 displays a tree.

Every component of a forest is a tree, so every forest is a disjoint union of trees. The next result is an immediate consequence of 3.37.

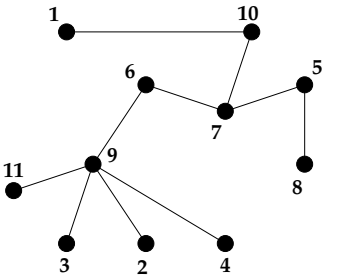


FIGURE 3.13
A tree.

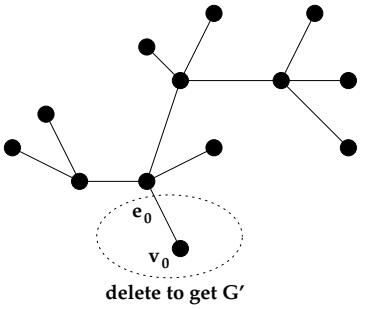


FIGURE 3.14
Pruning a leaf from a tree gives another tree.

3.67. Theorem: Trees Have Leaves. If T is a tree with more than one vertex, then T has at least two leaves.

3.68. Definition: Pruning. Suppose $G = (V, E)$ is a simple graph, v_0 is a leaf in G, and e_0 is the unique edge incident to the vertex v_0. The *graph obtained by pruning* v_0 *from* G is the graph $(V \sim \{v_0\}, E \sim \{e_0\})$.

3.69. Pruning Lemma. If T is an n-vertex tree, v_0 is a leaf of T, and T' is obtained from T by pruning v_0, then T' is a tree with $n-1$ vertices.

Proof. First, T has no cycles, and the deletion of v_0 and the associated edge e_0 will not create any cycles. So T' is acyclic. Second, let $u, w \in V(T')$. There is a path from u to w in T. Since $u \neq v_0 \neq w$, this path will not use the edge e_0 or the vertex v_0. Thus there is a path from u to w in T', so T' is connected. □

Figure 3.14 illustrates the pruning lemma.

To illustrate an application of pruning, we now prove a fundamental relationship between the number of vertices and edges in a tree (this also follows from 3.64).

3.70. Theorem: Number of Edges in a Tree. If G is a tree with $n > 0$ vertices, then G has $n-1$ edges.

Proof. We argue by induction on n. If $n = 1$, then G must have $0 = n - 1$ edges. Assume $n > 1$ and that the result holds for trees with $n-1$ vertices. Let T be a tree with n vertices. We know that T has at least one leaf; let v_0 be one such leaf. Let T' be the graph obtained

by pruning v_0 from T. By the pruning lemma, T' is a tree with $n-1$ vertices. By induction, T' has $n-2$ edges. Hence, T has $n-1$ edges. □

3.71. Theorem: Characterizations of Trees. Let G be a graph with n vertices. The following conditions are logically equivalent:
(a) G is a tree (i.e., G is connected and acyclic).
(b) G is connected and has $\leq n-1$ edges.
(c) G is acyclic and has $\geq n-1$ edges.
(d) G has no loop edges, and for all $u, v \in V(G)$, there is a unique path in G from u to v.
Moreover, when these conditions hold, G has $n-1$ edges.

Proof. First, (a) implies (b) and (a) implies (c) by 3.70. Second, (a) is equivalent to (d) by virtue of 3.54 and 3.63. Third, let us prove (b) implies (a). Assume G is connected with $k \leq n-1$ edges. If G has a cycle, delete one edge on some cycle of G. The resulting graph is still connected (by 3.58) and has $k-1$ edges. Continue to delete edges in this way, one at a time, until there are no cycles. If we deleted i edges total, the resulting graph is a tree with $k-i \leq n-1-i$ edges and n vertices. By 3.70, we must have $i=0$ and $k=n-1$. So no edges were deleted, and G itself is in fact a tree.

Fourth, let us prove (c) implies (a). Assume G is acyclic with $k \geq n-1$ edges. If G is not connected, add an edge joining two distinct components of G. The resulting graph is still acyclic (by 3.60) and has $k+1$ edges. Continue to add edges in this way, one at a time, until the graph becomes connected. If we added i edges total, the resulting graph is a tree with $k+i \geq n-1+i$ edges and n vertices. By 3.70, we must have $i=0$ and $k=n-1$. So no edges were added, and G itself is in fact a tree. □

3.11 Counting Trees

The next theorem, usually attributed to Cayley, counts n-vertex trees.

3.72. Theorem: Enumeration of Trees. For all $n \geq 1$, there are n^{n-2} trees with vertex set $\{1, 2, \ldots, n\}$.

3.73. Example. Figure 3.15 displays all $4^{4-2} = 16$ trees with vertex set $\{1, 2, 3, 4\}$.

Theorem 3.72 is an immediate consequence of 3.47 and the following bijection.

3.74. Theorem: Trees vs. Rooted Trees. Let V be a finite set and $v_0 \in V$. There is a bijection from the set A of trees with vertex set V to the set B of rooted trees with vertex set V and root v_0.

Proof. We define maps $f : A \to B$ and $g : B \to A$ that are two-sided inverses. First, given $T = (V, E) \in A$, construct $f(T) = (V, E')$ as follows. For each $v \in V$ with $v \neq v_0$, there exists a unique path from v to v_0 in T. Letting $e = \{v, w\}$ be the first edge on this path, we add the directed edge (v, w) to E'. Also, we add the loop edge (v_0, v_0) to E'. Since T has no cycles, the only possible cycle in the resulting functional digraph $f(T)$ is the 1-cycle (v_0). It follows that $f(T)$ is a rooted tree on V with root v_0 (see 3.42).

Next, given a rooted tree $S \in B$, define $g(S)$ by deleting the unique loop edge (v_0, v_0) and replacing every directed edge (v, w) by an undirected edge $\{v, w\}$. The resulting graph $g(S)$ has n vertices and $n-1$ edges. To see that $g(S)$ is connected, fix $y, z \in V$. Following outgoing edges from y (resp. z) in S produces a directed path from y (resp. z) to v_0 in S.

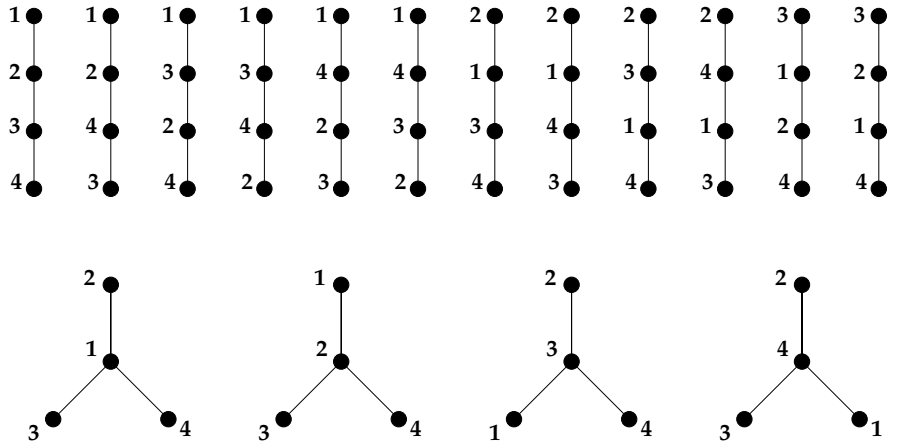


FIGURE 3.15
The 16 trees on four vertices.

In the undirected graph $g(S)$, we can concatenate the path from y to v_0 with the reverse of the path from z to v_0 to get a walk from y to z. It follows that $g(S)$ is a tree.

It is routine to check that $g \circ f = \mathrm{id}_A$, since f assigns a certain orientation to each edge of the original tree, and this orientation is then forgotten by g. It is somewhat less routine to verify that $f \circ g = \mathrm{id}_B$; we leave this to the reader. (One must check that the edge orientations in $f(g(S))$ agree with the edge orientations in S, for each $S \in B$.) □

A different bijective proof of Cayley's theorem, which employs parking functions, is presented in §12.5. We next prove a refinement of Cayley's theorem that counts the number of trees such that each vertex has a specified degree. We give an algebraic proof first, and then convert this to a bijective proof in the next section.

3.75. Theorem: Counting Trees with Specified Degrees. Suppose $n \geq 2$ and $d_1, \ldots, d_n \geq 0$ are fixed integers. If $d_1 + \cdots + d_n = 2n - 2$, then there are

$$\binom{n-2}{d_1 - 1, d_2 - 1, \ldots, d_n - 1} = \frac{(n-2)!}{\prod_{j=1}^{n}(d_j - 1)!}$$

trees with vertex set $\{v_1, v_2, \ldots, v_n\}$ such that $\deg(v_j) = d_j$ for all j. If $d_1 + \cdots + d_n \neq 2n-2$, then there are no such trees.

Proof. The last statement holds because any tree T on n vertices has $n-1$ edges, and thus $\sum_{i=1}^{n} \deg(v_i) = 2(n-1)$. Assume henceforth that $d_1 + \cdots + d_n = 2n - 2$. We prove the result by induction on n. First consider the case $n = 2$. If $d_1 = d_2 = 1$, there is exactly one valid tree, and $\binom{n-2}{d_1-1, d_2-1} = 1$. For any other choice of d_1, d_2 adding to 2, there are no valid trees, and $\binom{n-2}{d_1-1, d_2-1} = 0$.

Now assume $n > 2$ and that the theorem is known to hold for trees with $n-1$ vertices. Let A be the set of trees T with $V(T) = \{v_1, \ldots, v_n\}$ and $\deg(v_j) = d_j$ for all j. If $d_j = 0$ for some j, then A is empty and the formula in the theorem is zero by convention. Now suppose $d_j > 0$ for all j. We must have $d_i = 1$ for some i, for otherwise $d_1 + \cdots + d_n \geq 2n > 2n - 2$. Fix an i with $d_i = 1$. Note that v_i is a leaf in T for every $T \in A$. Now define

$$A_k = \{T \in A : \{v_i, v_k\} \in E(T)\} \qquad (1 \leq k \leq n, k \neq i).$$

A_k is the set of trees in A in which the leaf v_i is attached to the vertex v_k. A is the disjoint union of the sets A_k.

Fix $k \neq i$. Pruning the leaf v_i gives a bijection between A_k and the set B_k of all trees with vertex set $\{v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n\}$ such that $\deg(v_j) = d_j$ for $j \neq i, k$ and $\deg(v_k) = d_k - 1$. By induction hypothesis,

$$|B_k| = \frac{(n-3)!}{(d_k-2)! \prod_{\substack{1 \leq j \leq n \\ j \neq i,k}} (d_j - 1)!}.$$

Therefore,

$$\begin{aligned} |A| &= \sum_{k \neq i} |A_k| = \sum_{k \neq i} |B_k| = \sum_{k \neq i} \frac{(n-3)!}{(d_k-2)! \prod_{j \neq k,i} (d_j-1)!} \\ &= \sum_{k \neq i} \frac{(n-3)!(d_k-1)}{\prod_{j \neq i} (d_j-1)!} = \sum_{k \neq i} \frac{(n-3)!(d_k-1)}{\prod_{j=1}^{n} (d_j-1)!} \\ &\qquad \text{(since } (d_i-1)! = 0! = 1\text{)} \\ &= \frac{(n-3)!}{\prod_{j=1}^{n} (d_j-1)!} \sum_{k \neq i} (d_k - 1). \end{aligned}$$

Now, since $d_i = 1$, $\sum_{k \neq i} (d_k - 1) = \sum_{k=1}^{n} (d_k - 1) = (2n-2) - n = n - 2$. Inserting this into the previous formula, we see that

$$|A| = \frac{(n-2)!}{\prod_{j=1}^{n} (d_j - 1)!},$$

which completes the induction proof. □

3.76. Corollary: Second Proof of 3.72. Let us sum the previous formula over all possible degree sequences $(d_1, \ldots, d_n)$. Making the change of variables $c_i = d_i - 1$ and invoking the multinomial theorem 2.12, we see that the total number of trees on this vertex set is

$$\begin{aligned} \sum_{\substack{d_1 + \cdots + d_n = 2n-2 \\ d_i \geq 0}} \binom{n-2}{d_1 - 1, d_2 - 1, \ldots, d_n - 1} &= \sum_{\substack{c_1 + \cdots + c_n = n-2 \\ c_i \geq 0}} \binom{n-2}{c_1, c_2, \ldots, c_n} 1^{c_1} 1^{c_2} \cdots 1^{c_n} \\ &= (1 + 1 + \cdots + 1)^{n-2} = n^{n-2}. \end{aligned}$$

3.12 Pruning Maps

We now develop a bijective proof of 3.75. Suppose $n \geq 2$ and $d_1, \ldots, d_n$ are positive integers that sum to $2n-2$. Let $V = \{v_1, \ldots, v_n\}$ be a vertex set consisting of n positive integers. Let A be the set of trees T with vertex set V such that $\deg(v_i) = d_i$ for all i. Let B be the set of words $\mathcal{R}(v_1^{d_1-1} v_2^{d_2-1} \cdots v_n^{d_n-1})$ as in 1.44. Each word $w \in B$ has length $n-2$ and consists of $d_j - 1$ copies of v_j for all j. To prove 3.75, it suffices to define a bijection $f : A \to B$.

Given a tree $T \in A$, we compute $f(T)$ by repeatedly pruning off the largest leaf of T, recording for each leaf the vertex adjacent to it in T. More formally, for i ranging from 1 to $n-1$, let x be the largest leaf of T; define w_i to be the unique neighbor of x in T; then modify T by pruning the leaf x. This process produces a word $w_1 \cdots w_{n-1}$; we define $f(T) = w_1 \cdots w_{n-2}$.

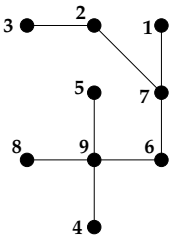


FIGURE 3.16
A tree with $(d_1, \ldots, d_9) = (1, 2, 1, 1, 1, 2, 3, 1, 4)$.

3.77. Example. Let T be the tree shown in Figure 3.16. To compute $f(T)$, we prune leaves from T in the following order: $8, 5, 4, 9, 6, 3, 2, 7$. Recording the neighbors of these leaves, we see that $w = f(T) = 9996727$. Observe that the algorithm computes $w_{n-1} = 1$, but this letter is not part of the output word w. Also observe that $w \in \mathcal{R}(1^0 2^1 3^0 4^0 5^0 6^1 7^2 8^0 9^3) = \mathcal{R}(1^{d_1-1} \cdots 9^{d_9-1})$.

The observations in the last example hold in general. For, given any $T \in A$, repeatedly pruning leaves from T will produce a sequence of smaller trees, by the pruning lemma 3.69. By 3.67, each such tree (except the last tree) has at least two leaves, so vertex v_1 will never be chosen for pruning. In particular, v_1 is always the last vertex left, so that w_{n-1} is always v_1. Furthermore, if v_j is any vertex different from v_1, then the number of occurrences of v_j in $w_1 w_2 \cdots w_{n-1}$ is exactly $d_j - 1$. For, every time a pruning operation removes an edge touching v_j, we set $w_i = v_j$ for some i, *except* when we are removing the last remaining edge touching v_j (which occurs when v_j has become the largest leaf and is being pruned). The same reasoning shows that v_1 (which never gets pruned) appears d_1 times in $w_1 \cdots w_{n-1}$. Since $w_{n-1} = v_1$, *every* vertex v_j occurs $d_j - 1$ times in the output word $w_1 \cdots w_{n-2}$.

To see that f is a bijection, we argue by induction on the number of vertices. The result holds when $n = 2$, since in this case, A consists of a single tree with two nodes, and B consists of a single word (the empty word). Now suppose $n > 2$ and the maps f (defined for trees with fewer than n vertices) are already known to be bijections. Given $w = w_1 \cdots w_{n-2} \in B$, we will show there exists exactly one $T \in A$ with $f(T) = w$. If such T exists, the leaves of T are precisely the vertices in $V(T)$ that do *not* appear in w. Thus, the first leaf that gets pruned when computing $f(T)$ must be the largest element z of $V(T) \sim \{w_1, \ldots, w_{n-2}\}$. By induction hypothesis, there exists exactly one tree T' on the vertex set $V(T) \sim \{z\}$ (with the appropriate vertex degrees) such that $f(T') = w_2 \cdots w_{n-2}$. This given, we will have $f(T) = w$ iff T is the tree obtained from T' by attaching a new leaf z as a neighbor of vertex w_1. One readily confirms that this graph *is* in A (i.e., the graph is a tree with the correct vertex degrees). This completes the induction argument. The proof also yields a recursive algorithm for computing $f^{-1}(w)$. The key point is to use the letters *not* seen in w (and its suffixes) to determine the identity of the leaf that was pruned at each stage.

3.78. Example. Given $w = 6799297$ and $V = \{1, 2, \ldots, 9\}$, let us compute the tree $f^{-1}(w)$ with vertex set V. The leaves of this tree must be $\{1, 3, 4, 5, 8\}$, which are the elements of V not seen in w. Leaf 8 was pruned first and was adjacent to vertex 6. So now we must compute the tree $f^{-1}(799297)$ with vertex set $V \sim \{8\}$. Here, leaf 6 was pruned first and was adjacent to vertex 7. Continuing in this way, we deduce that the leaves were pruned in the order $8, 6, 5, 4, 3, 2, 9, 7$; and the neighbors of these leaves (reading from w) were $6, 7, 9, 9, 2, 9, 7, 1$. Thus, $f^{-1}(w)$ is the tree shown in Figure 3.17.

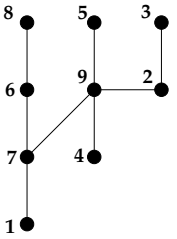


FIGURE 3.17
The tree associated to $w = 6799297$ by f^{-1}.

3.13 Ordered Trees and Terms

In this section, we study combinatorial structures called *ordered trees*, which are defined recursively as follows.

3.79. Definition: Ordered Trees. The symbol 0 is an ordered tree. If $n > 0$ and $T_1, \ldots, T_n$ is a sequence of ordered trees, then the $(n+1)$-tuple

$$(n, T_1, T_2, \ldots, T_n)$$

is an ordered tree. All ordered trees arise by applying these two rules a finite number of times.

We often think of ordered trees pictorially. The ordered tree 0 is depicted as a single node. The ordered tree $(n, T_1, T_2, \ldots, T_n)$ is drawn by putting a single "root node" at the top of the picture with n edges leading down. At the ends of these edges, reading from left to right, we recursively draw pictures of the trees $T_1, T_2, \ldots, T_n$ in this order. The term "ordered tree" emphasizes the fact that the left-to-right order of the children of each node is significant. Note that an ordered tree is *not* a tree in the graph-theoretic sense, and ordered trees are not the same as rooted trees.

3.80. Example. Figure 3.18 illustrates the ordered tree

$$T = (4, (2, 0, (1, 0)), 0, (3, 0, (3, 0, 0, 0), 0), 0).$$

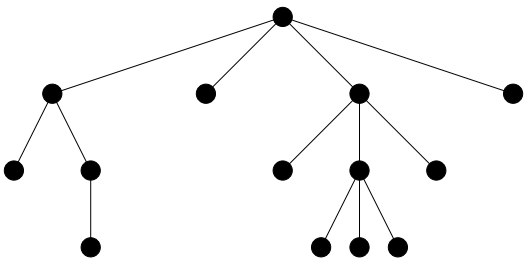


FIGURE 3.18
Picture of an ordered tree.

Ordered trees can be used to model algebraic expressions that are built up by applying functions to lists of arguments. For example, the tree T in the previous example represents the syntactic structure of the following algebraic expression:

$$f(g(x_1, h(x_2)), x_3, k(x_4, j(x_5, x_6, x_7), x_8), x_9).$$

More specifically, if we replace each function symbol f, g, h, k, j by its *arity* (number of arguments) and replace each variable x_i by zero, we obtain

$$4(2(0, 1(0)), 0, 3(0, 3(0, 0, 0), 0), 0).$$

This becomes T if we move each left parenthesis to the left of the positive integer immediately preceding it and put a comma in its original location.

Surprisingly, it turns out that the syntactic structure of such an algebraic expression is uniquely determined even if we erase all the parentheses. To prove this statement, we introduce a combinatorial object called a *term* that is like an ordered tree, but contains no parentheses. For example, the algebraic expression above will be modeled by the term 42010030300000.

3.81. Definition: Words and Terms. A *word* is a finite sequence of natural numbers. We define *terms* recursively as follows. The word 0 is a term. If $n > 0$ and $T_1, T_2, \ldots, T_n$ are terms, then the word $nT_1T_2\cdots T_n$ is a term. All terms arise by applying these two rules a finite number of times.

We see from this definition that every term is a *nonempty* word.

3.82. Definition: Weight of a Word. Given a word $w = w_1w_2\cdots w_s$, the *weight* of w is $\mathrm{wt}(w) = w_1 + w_2 + \cdots + w_s - s$.

For example, $\mathrm{wt}(42010030300000) = 13 - 14 = -1$. Note that $\mathrm{wt}(vw) = \mathrm{wt}(v) + \mathrm{wt}(w)$ for all words v, w. The next result uses weights to characterize terms.

3.83. Theorem: Characterization of Terms. A word $w = w_1w_2\cdots w_s$ is a term iff $\mathrm{wt}(w) = -1$ and $\mathrm{wt}(w_1w_2\cdots w_k) \geq 0$ for all $k < s$.

Proof. We argue by strong induction on the length s of the word. First suppose $w = w_1w_2\cdots w_s$ is a term of length s. If $w = 0$, then the weight condition holds. Otherwise, we must have $w = nT_1T_2\cdots T_n$ where $n > 0$ and $T_1, T_2, \ldots, T_n$ are terms. Since each T_i has length less than s, the induction hypothesis shows that $\mathrm{wt}(T_i) = -1$ and every proper prefix of T_i has nonnegative weight. So, first of all, $\mathrm{wt}(w) = \mathrm{wt}(n) + \mathrm{wt}(T_1) + \cdots + \mathrm{wt}(T_n) = (n-1) - n = -1$. On the other hand, consider a proper prefix $w_1w_2\cdots w_k$ of w. If $k = 1$, the weight of this prefix is $n - 1$, which is nonnegative since $n > 0$. If $k > 1$, we must have $w_1w_2\cdots w_k = nT_1\cdots T_iz$ where $0 \leq i < n$ and z is a proper prefix of T_{i+1}. Using the induction hypothesis, the weight of $w_1w_2\cdots w_k$ is therefore $(n-1)-i+\mathrm{wt}(z) \geq (n-i)-1 \geq 0$.

For the converse, we also use strong induction on the length of the word. Let $w = w_1w_2\cdots w_s$ satisfy the weight conditions. The empty word has weight zero, so $s > 0$. If $s = 1$, then $\mathrm{wt}(w_1) = -1$ forces $w = 0$, so that w is a term in this case. Now suppose $s > 1$. The first symbol w_1 must be an integer $n > 0$, lest the proper prefix w_1 of w have negative weight. Observe that appending one more letter to any word decreases the weight by at most 1. Since $\mathrm{wt}(w_1) = n - 1$ and $\mathrm{wt}(w_1w_2\cdots w_s) = -1$, there exists a least integer k_1 with $\mathrm{wt}(w_1w_2\cdots w_{k_1}) = n - 2$. Now if $n \geq 2$, there exists a least integer $k_2 > k_1$ with $\mathrm{wt}(w_1w_2\cdots w_{k_2}) = n - 3$. We continue similarly, obtaining integers $k_1 < k_2 < \cdots < k_n$ such that k_i is the least index following k_{i-1} such that $\mathrm{wt}(w_1w_2\cdots w_{k_i}) = n - 1 - i$. Because w satisfies the weight conditions, we must have $k_n = s$. Now define n subwords

$T_1 = w_2 w_3 \cdots w_{k_1}$, $T_2 = w_{k_1+1} w_{k_1+2} \cdots w_{k_2}$, ..., $T_n = w_{k_{n-1}+1} \cdots w_s$. Evidently $w = nT_1T_2 \cdots T_n$. For all $i \leq n$, $nT_1T_2 \cdots T_{i-1}$ has weight $n-i$, $nT_1T_2 \cdots T_i$ has weight $n-i-1$, and (by minimality of k_i) no proper prefix of $nT_1T_2 \cdots T_i$ has weight less than $n-i$. It follows that T_i has weight -1 but every proper prefix of T_i has nonnegative weight. Thus each T_i satisfies the weight conditions and has length less than w. By induction, every T_i is a term. Then $w = nT_1T_2 \cdots T_n$ is also a term, completing the induction. □

3.84. Corollary. No proper prefix of a term is a term.

3.85. Theorem: Unique Readability of Terms. For every term w, there exists a unique integer $n \geq 0$ and unique terms $T_1, \ldots, T_n$ such that $w = nT_1 \cdots T_n$.

Proof. Existence follows from the recursive definition of terms. We prove uniqueness by induction on the length of w. Suppose $w = nT_1 \cdots T_n = mT'_1 \cdots T'_m$ where $n, m \geq 0$ and every T_i and T'_j is a term. We must prove $n = m$ and $T_i = T'_i$ for all $i \leq n$. First, $n = w_1 = m$. If $T_1 \neq T'_1$, then one of T_1 and T'_1 must be a proper prefix of the other, in violation of the preceding corollary. So $T_1 = T'_1$. Then if $T_2 \neq T'_2$, one of T_2 and T'_2 must be a proper prefix of the other, in violation of the corollary. Continuing similarly, we see that $T_i = T'_i$ for all i. □

Using the previous theorem and induction, one readily proves that erasing all parentheses defines a bijection from ordered trees to terms. Therefore, to enumerate various collections of ordered trees, it suffices to enumerate the corresponding collections of terms. This technique will be used in the next section.

3.14 Ordered Forests and Lists of Terms

We continue our study of ordered trees and terms by introducing two more general objects: ordered forests and lists of terms.

3.86. Definition: Ordered Forests. For $n \geq 0$, an *ordered forest of n trees* is a list $(T_1, T_2, \ldots, T_n)$, where each T_i is an ordered tree.

3.87. Definition: Lists of Terms. For $n \geq 0$, a *list of n terms* is a word w of the form $w = T_1T_2 \cdots T_n$, where each T_i is a term.

3.88. Theorem: Weight Characterization of Lists of Terms. A word $w = w_1w_2 \cdots w_s$ is a list of n terms iff $\mathrm{wt}(w) = -n$ and $\mathrm{wt}(w_1w_2 \cdots w_k) > -n$ for all $k < s$.

Proof. First suppose w is a list of n terms, say $w = T_1T_2 \cdots T_n$. Then $nw = nT_1T_2 \cdots T_n$ is a single term. This term has weight -1, by 3.83, so w has weight $-1 - \mathrm{wt}(n) = -n$. If $\mathrm{wt}(w_1 \cdots w_k) \leq -n$ for some $k < s$, then the proper prefix $nw_1 \cdots w_k$ of the term nw would have negative weight, contradicting 3.83.

Conversely, suppose w satisfies the weight conditions in 3.88. Then the word nw satisfies the weight conditions in 3.83, as one immediately verifies. So nw is a term, which must have the form $nT_1T_2 \cdots T_n$ for suitable terms $T_1, \ldots, T_n$. Then $w = T_1T_2 \cdots T_n$ is a list of n terms. □

3.89. Theorem: Unique Readability of Lists of Terms. If $w = T_1T_2 \cdots T_n$ is a list of n terms, then n and the terms T_i are uniquely determined by w.

Proof. First, $n = -\operatorname{wt}(w)$ is uniquely determined by w. To see that the T_i are unique, add an n to the beginning of w and then appeal to 3.85. □

We deduce that erasing parentheses gives a bijection between ordered forests of n trees and lists of n terms.

The next lemma reveals a key property that will allow us to enumerate lists of terms.

3.90. Cycle Lemma for Lists of Terms. Suppose $w = w_1 w_2 \cdots w_s$ is a word of weight $-n < 0$. There exist exactly n indices $i \leq s$ such that the cyclic rotation

$$R_i(w) = w_i w_{i+1} \cdots w_s w_1 w_2 \cdots w_{i-1}$$

is a list of n terms.

Proof. Step 1: We prove the result when w itself is a list of n terms. Say $w = T_1 T_2 \cdots T_n$ where T_j is a term of length k_j. Then $R_i(w)$ is a list of n terms for the n indices $i \in \{1, k_1+1, k_1+k_2+1, \ldots, k_1+k_2+\cdots+k_{n-1}+1\}$. Suppose i is another index (different from those just listed) such that $R_i(w)$ is a list of n terms. For some $j \leq n$, we must have

$$R_i(w) = y T_{j+1} \cdots T_n T_1 \cdots T_{j-1} z$$

where $T_j = zy$ and z, y are nonempty words. Since $\operatorname{wt}(z) \geq 0$ but $\operatorname{wt}(T_j) = -1$, we must have $\operatorname{wt}(y) < 0$. So

$$\operatorname{wt}(y T_{j+1} \cdots T_{j-1}) = \operatorname{wt}(y) + \operatorname{wt}(T_{j+1}) + \cdots + \operatorname{wt}(T_{j-1}) < -(n-1).$$

Then $y T_{j+1} \cdots T_{j-1}$ is a proper prefix of $R_i(w)$ with weight $\leq -n$, in violation of 3.88.

Step 2: We prove the result for a general word w. It suffices to show that there exists at least one i such that $R_i(w)$ is a list of n terms. For then, since we obtain the same collection of words by cyclically shifting w and $R_i(w)$, the desired result will follow from Step 1.

First note that all cyclic rotations of w have weight $\sum_{j=1}^{s} \operatorname{wt}(w_j) = \operatorname{wt}(w) = -n$. Let m be the minimum weight of any prefix $w_1 w_2 \cdots w_k$ of w, where $1 \leq k \leq s$. Choose k minimal such that $\operatorname{wt}(w_1 w_2 \cdots w_k) = m$. If $k = s$, then $m = -n$, and by minimality of k and 3.88, w itself is already a list of n terms. Otherwise, let $i = k+1$. We claim $R_i(w)$ is a list of n terms. It suffices to check that each proper prefix of $R_i(w)$ has weight $> -n$. On one hand, for all j with $i \leq j \leq s$, the prefix $w_i \cdots w_j$ of $R_i(w)$ cannot have negative weight; otherwise, $\operatorname{wt}(w_1 \cdots w_k w_i \cdots w_j) < m$ violates the minimality of m. So $\operatorname{wt}(w_i \cdots w_j) \geq 0 > -n$. Note that when $j = s$, we have $\operatorname{wt}(w_i \cdots w_s) = \operatorname{wt}(w) - \operatorname{wt}(w_1 \cdots w_k) = -n - m$. Now consider j in the range $1 \leq j < k$. If $\operatorname{wt}(w_i \cdots w_s w_1 \cdots w_j) \leq -n$, then

$$\operatorname{wt}(w_1 \cdots w_j) = \operatorname{wt}(w_i \cdots w_s w_1 \cdots w_j) - \operatorname{wt}(w_i \cdots w_s) \leq -n - (-n - m) = m.$$

But this violates the choice of k as the *least* index such that the prefix ending at k has minimum weight. So $\operatorname{wt}(w_i \cdots w_s w_1 \cdots w_j) > -n$. It now follows from 3.88 that $R_i(w)$ is indeed a list of n terms. □

Suppose w is a list of n terms containing exactly k_i occurrences of i for each $i \geq 0$. We have

$$-n = \operatorname{wt}(w) = \sum_{i \geq 0} k_i \operatorname{wt}(i) = \sum_{i \geq 0} (i-1) k_i = -k_0 + \sum_{i \geq 1} (i-1) k_i.$$

It follows that $k_0 = n + \sum_{i \geq 1} (i-1) k_i$ in this situation. Conversely, if k_0 satisfies this relation, then $\operatorname{wt}(w) = -n$ for all $w \in \mathcal{R}(0^{k_0} 1^{k_1} 2^{k_2} \cdots)$. We now have all the ingredients needed for our main enumeration result.

3.91. Theorem: Enumeration of Lists of Terms. Let $n > 0$ and $k_0, k_1, \ldots, k_t$ be given natural numbers such that $k_0 = n + \sum_{i=1}^{t}(i-1)k_i$. The number N of words w such that w is a list of n terms containing k_i copies of i for $0 \le i \le t$ is

$$\frac{n}{s}\binom{s}{k_0, k_1, \ldots, k_t} = \frac{n(s-1)!}{k_0!k_1!\cdots k_t!},$$

where $s = \sum_{i=0}^{t} k_i = n + \sum_{i=1}^{t} ik_i$ is the common length of all such words.

Proof. Step 1. Let A be the set of all pairs (w, j), where $w \in \mathcal{R}(0^{k_0}1^{k_1}\cdots t^{k_t})$ is a word and $j \le s$ is an index such that the cyclic rotation $R_j(w)$ is a list of n terms. Combining 3.90 and 1.46, we see that $|A| = n\binom{s}{k_0,k_1,\ldots,k_t}$.

Step 2. Let B be the set of all pairs (w, i) where $w \in \mathcal{R}(0^{k_0}1^{k_1}\cdots t^{k_t})$ is a list of n terms (necessarily of length s) and $1 \le i \le s$. By the product rule, $|B| = sN$.

Step 3. To complete the proof, we show that $|A| = |B|$ by exhibiting mutually inverse bijections $f : A \to B$ and $g : B \to A$. We define $f(w, j) = (R_j(w), j)$ for all $(w, j) \in A$, and $g(w, i) = (R_i^{-1}(w), i)$ for all $(w, i) \in B$. □

3.15 Graph Coloring

This section introduces the graph coloring problem and some of its applications.

3.92. Definition: Colorings. Let $G = (V, E)$ be a simple graph, and let C be a finite set. A *coloring* of G using colors in C is a function $f : V \to C$. A coloring f of G is a *proper coloring* iff for every edge $\{u, v\} \in E$, $f(u) \ne f(v)$.

Intuitively, we are coloring each vertex of G using one of the available colors in the set C. For each $v \in V$, $f(v)$ is the color assigned to vertex v. A coloring is proper iff no two adjacent vertices in G are assigned the same color.

3.93. Definition: Chromatic Functions and Chromatic Numbers. Let G be a simple graph. For each positive integer x, let $\chi_G(x)$ be the number of proper colorings of G using colors in $\{1, 2, \ldots, x\}$. The function $\chi_G : \mathbb{N}^+ \to \mathbb{N}$ is called the *chromatic function* of G. The minimal x such that $\chi_G(x) > 0$ is called the *chromatic number* of G.

The chromatic number is the least number of colors required to obtain a proper coloring of G. The function χ_G is often called the *chromatic polynomial of* G because of 3.100 below.

3.94. Example. Suppose G is a simple graph with n vertices and no edges. Then $\chi_G(x) = x^n$ since we can assign any of the x colors to each vertex. The chromatic number for this graph is 1.

3.95. Example. At the other extreme, suppose G is a simple graph with n vertices such that there is an edge joining every pair of distinct vertices. Color the vertices one at a time. The first vertex can be colored in x ways. The second vertex must have a color different from the first, so there are $x-1$ choices. In general, the ith vertex must have a color distinct from all of its predecessors, so there are $x-(i-1)$ choices for the color of this vertex. The product rule gives $\chi_G(x) = x(x-1)(x-2)\cdots(x-n+1) = (x)\downarrow_n$. The chromatic number for this graph is n. Recall from §2.13 that

$$(x)\downarrow_n = \sum_{k=1}^{n} s(n,k)x^k,$$

so that the function χ_G in this example is a polynomial whose coefficients are the signed Stirling numbers of the first kind.

3.96. Example: Cycles. Consider the simple graph

$$G = (\{1,2,3,4\}, \{\{1,2\},\{2,3\},\{3,4\},\{4,1\}\}).$$

G consists of four vertices joined in a 4-cycle. We might attempt to compute $\chi_G(x)$ via the product rule as follows. Color vertex 1 in x ways. Then color vertex 2 in $x-1$ ways, and color vertex 3 in $x-1$ ways. We run into trouble at vertex 4, because we do not know whether vertices 1 and 3 were assigned the same color. This example shows that we cannot always compute χ_G by the product rule alone. In this instance, we can classify proper colorings based on whether vertices 1 and 3 receive the same or different colors. If they receive the same color, the number of proper colorings is $x(x-1)(x-1)$ (color vertices 1 and 3 together, then color vertex 2 a different color, then color vertex 4 a different color from 1 and 3). If vertex 1 and 3 receive different colors, the number of proper colorings is $x(x-1)(x-2)(x-2)$ (color vertex 1, then vertex 3, then vertex 2, then vertex 4). Hence

$$\chi_G(x) = x(x-1)(x-1) + x(x-1)(x-2)(x-2) = x^4 - 4x^3 + 6x^2 - 3x.$$

The chromatic number for this graph is 2.

More generally, consider the graph C_n consisting of n vertices joined in a cycle. It is routine to establish that the chromatic number of C_n is 1 for $n = 1$, is 2 for all even n, and is 3 for all odd $n > 1$. On the other hand, it is not immediately evident how to compute the chromatic function for C_n when $n > 4$. We will deduce a recursion for these functions shortly as a special case of a general recursion for computing chromatic functions.

Here is an application that can be analyzed using graph colorings and chromatic numbers. Suppose we are trying to schedule meetings for a number of committees. If two committees share a common member, they cannot meet at the same time. Consider the graph G whose vertices represent the various committees, and where there is an edge between two vertices iff the corresponding committees share a common member. Suppose there are x available time slots in which meetings may be scheduled. A coloring of G with x colors represents a particular scheduling of committee meetings to time slots. The coloring is proper iff the schedule creates no time conflicts for any committee member. The chromatic number is the least number of time slots needed to avoid all conflicts, while $\chi_G(x)$ is the number of different conflict-free schedules using x (distinguishable) time slots.

3.97. Example. Six committees have members as specified in the following table.

Committee	Members
A	Kemp, Oakley, Saunders
B	Gray, Saunders, Russell
C	Byrd, Oakley, Quinn
D	Byrd, Jenkins, Kemp
E	Adams, Jenkins, Wilson
F	Byrd, Gray, Russell

Figure 3.19 displays the graph G associated to this set of committees. To compute $\chi_G(x)$, consider cases based on whether vertices A and F receive the same color. If A and F are colored the same, the number of proper colorings is $x(x-1)(x-2)(x-1)(x-1)$ [color A and F, then C, D, B, and E]. If A and F receive different colors, the number of proper colorings is $x(x-1)(x-2)(x-3)(x-2)(x-1)$ [color A, F, C, D, B, E]. Thus,

$$\chi_G(x) = x(x-1)^2(x-2)(x-1+(x-2)(x-3)) = x^6 - 8x^5 + 26x^4 - 42x^3 + 33x^2 - 10x.$$

The chromatic number of G is 3.

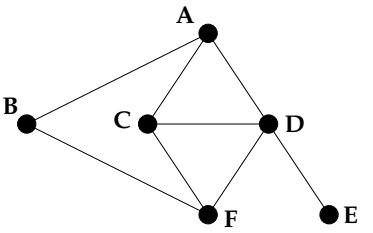


FIGURE 3.19
Conflict graph for six committees.

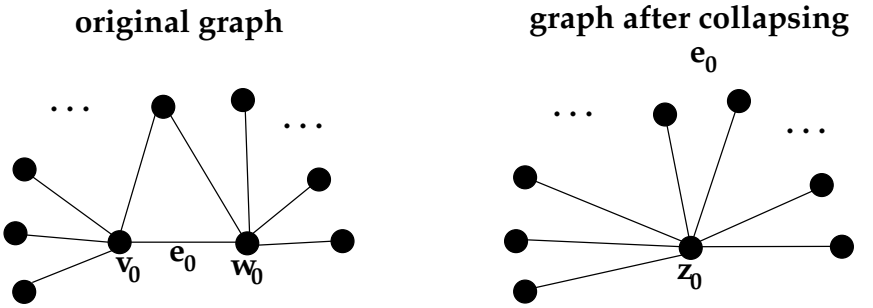


FIGURE 3.20
Collapsing an edge in a simple graph.

We are about to present a general recursion that can be used to compute the chromatic function of a simple graph. The recursion makes use of the following construction.

3.98. Definition: Collapsing an Edge. Let $G = (V, E)$ be a simple graph, and let $e_0 = \{v_0, w_0\}$ be a fixed edge of G. Let z_0 be a new vertex. We define a simple graph H called *the graph obtained from G by collapsing the edge e_0*. The vertex set of H is $(V \sim \{v_0, w_0\}) \cup \{z_0\}$. The edge set of H is

$$\begin{aligned}&\{\{x, y\} : x \neq v_0 \neq y \text{ and } x \neq w_0 \neq y \text{ and } \{x, y\} \in E\}\\ &\cup\{\{x, z_0\} : x \neq v_0 \text{ and } \{x, w_0\} \in E\}\\ &\cup\{\{x, z_0\} : x \neq w_0 \text{ and } \{x, v_0\} \in E\}.\end{aligned}$$

Pictorially, we construct H from G by shrinking the edge e_0 until the vertices v_0 and w_0 coincide. We replace these two overlapping vertices with a single new vertex z_0. All edges touching v_0 or w_0 (except the collapsed edge e_0) now touch z_0 instead. See Figure 3.20.

3.99. Theorem: Chromatic Recursion. Let $G = (V, E)$ be a simple graph. Fix any edge $e = \{v, w\} \in G$. Let $G' = (V, E \sim \{e\})$ be the simple graph obtained by deleting the edge e from G, and let G'' be the simple graph obtained from G by collapsing the edge e. Then

$$\chi_G(x) = \chi_{G'}(x) - \chi_{G''}(x).$$

Proof. Fix $x \in \mathbb{N}^+$, and let A, B, and C denote the set of proper colorings of G, G', and G'' (respectively) using x available colors. Write $B = B_1 \cup B_2$, where $B_1 = \{f \in B : f(v) = f(w)\}$ and $B_2 = \{f \in B : f(v) \neq f(w)\}$. Note that B_1 consists of the proper colorings of G' (if any) in which vertices v and w are assigned the same color. Let z be the new vertex

in G'' that replaces v and w. Given a proper coloring $f \in B_1$, we define a corresponding coloring f'' of G'' by setting $f''(z) = f(v) = f(w)$ and $f''(u) = f(u)$ for all $u \in V$ different from v and w. Since f is proper, it follows from the definition of the edge set of G'' that f'' is a proper coloring as well. Thus we have a map $f \mapsto f''$ from B_1 to C. This map is invertible, since the color of z in a coloring of G'' determines the common color of v and w in a coloring of G' belonging to B_1. We conclude that $|B_1| = |C|$.

On the other hand, B_2 consists of the proper colorings of G' in which vertices v and w are assigned different colors. These are precisely the proper colorings of G (since G has an edge between v and w, and G is otherwise identical to G'). Thus, $B_2 = A$. It follows that

$$\chi_G(x) = |A| = |B_2| = |B| - |B_1| = |B| - |C| = \chi_{G'}(x) - \chi_{G''}(x). \qquad \square$$

3.100. Corollary: Polynomiality of Chromatic Functions. For any graph G, $\chi_G(x)$ is a polynomial in x with integer coefficients. (This justifies the terminology *chromatic polynomial*.)

Proof. We use induction on the number of edges in G. If G has k vertices and no edges, the product rule gives $\chi_G(x) = x^k$, which is a polynomial in x. Now assume G has $m > 0$ edges. Fix such an edge e, and define G' and G'' as in the preceding theorem. G' has one fewer edge than G. When passing from G to G'', we lose the edge e and possibly identify other edges in G (e.g., if both endpoints of e are adjacent to a third vertex). In any case, G'' has fewer edges than G. By induction on m, we may assume that both $\chi_{G'}(x)$ and $\chi_{G''}(x)$ are polynomials in x with integer coefficients. So $\chi_G(x) = \chi_{G'}(x) - \chi_{G''}(x)$ is also a polynomial with integer coefficients. $\square$

3.101. Remark. We can use the chromatic recursion to compute χ_G recursively for any graph G. The base case of the calculation is a graph with k vertices and no edges, which has chromatic polynomial x^k. If G has more than one edge, G' and G'' both have strictly fewer edges than G. Thus, the recursive calculation will terminate after finitely many steps. However, this is quite an inefficient method for computing χ_G if G has many vertices and edges. Thus, direct counting arguments using the sum and product rules may be preferable to repeatedly applying the chromatic recursion.

3.102. Example. Consider the graph G shown on the left in Figure 3.21. We compute $\chi_G(x)$ by applying the chromatic recursion to the edge $e = \{d, h\}$. The graphs G' and G'' obtained by deleting and collapsing this edge are shown on the right in Figure 3.21. Direct arguments using the product rule show that

$$\chi_{G'}(x) = x(x-1)(x-2)(x-2)(x-1)(x-1) \qquad \text{(color a, c, d, f, b, h);}$$

$$\chi_{G''}(x) = x(x-1)(x-2)(x-2)(x-2) \qquad \text{(color z, a, c, f, b).}$$

Therefore,

$$\chi_G(x) = x(x-1)(x-2)^2((x-1)^2 - (x-2)) = x^6 - 8x^5 + 26x^4 - 43x^3 + 36x^2 - 12x.$$

3.103. Chromatic Polynomials for Cycles. For each $n \geq 3$, let C_n denote a graph consisting of n vertices joined in a cycle. Let C_1 denote a one-vertex graph, and let C_2 denote a graph with two vertices joined by an edge. Finally, let $\chi_n(x) = \chi_{C_n}(x)$ be the chromatic polynomials for these graphs. We see directly that

$$\chi_1(x) = x, \qquad \chi_2(x) = x(x-1) = x^2 - x, \qquad \chi_3(x) = x(x-1)(x-2) = x^3 - 3x^2 + 2x.$$

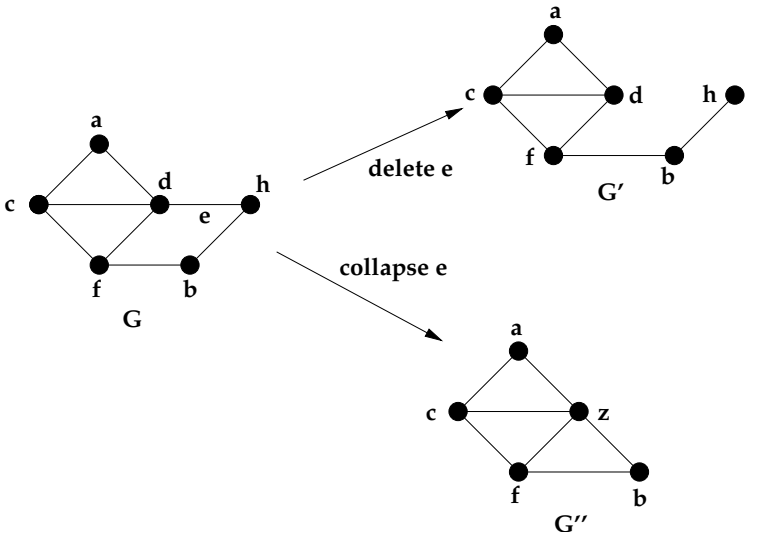


FIGURE 3.21
Using the chromatic recursion.

Fix $n > 3$ and fix any edge e in C_n. Deleting this edge leaves a graph in which n vertices are joined in a line; the chromatic polynomial of such a graph is $x(x-1)^{n-1}$. On the other hand, collapsing the edge e in C_n produces a graph isomorphic to C_{n-1}. The chromatic recursion therefore gives

$$\chi_n(x) = x(x-1)^{n-1} - \chi_{n-1}(x).$$

Using this recursion to compute $\chi_n(x)$ for small n suggests the closed formula

$$\chi_n(x) = (x-1)^n + (-1)^n(x-1) \qquad (n \geq 2).$$

We let the reader prove this formula for $\chi_n(x)$ by induction, using the chromatic recursion.

3.16 Spanning Trees

This section introduces the notion of a spanning tree for a graph. A recursion resembling the chromatic recursion 3.99 will allow us to count the spanning trees for a given graph. This will lead to a remarkable formula, called the matrix-tree theorem, that expresses the number of spanning trees as a certain determinant.

3.104. Definition: Subgraphs. Let $G = (V, E, \epsilon)$ and $H = (W, F, \eta)$ be graphs or digraphs. H is a *subgraph* of G iff $W \subseteq V$, $F \subseteq E$, and $\eta(f) = \epsilon(f)$ for all $f \in F$. H is an *induced subgraph* of G iff H is a subgraph such that F consists of *all* edges in E with both endpoints in W.

3.105. Definition: Spanning Trees. Given a graph $G = (V, E, \epsilon)$, a *spanning tree* for G is a subgraph H with vertex set V such that H is a tree. Let $\tau(G)$ be the number of spanning trees of G.

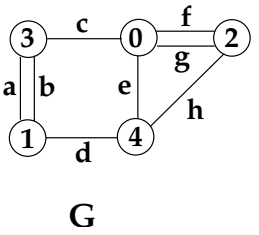


FIGURE 3.22
Graph used to illustrate spanning trees.

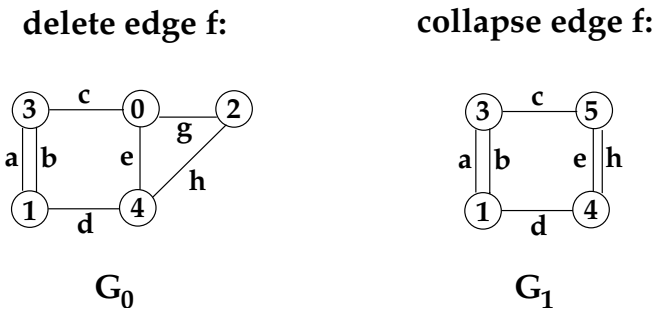


FIGURE 3.23
Effect of deleting or collapsing an edge.

3.106. Example. Consider the graph G shown in Figure 3.22. This graph has 31 spanning trees, which are specified by the following sets of edges:

$$\begin{array}{llllll}
\{a,c,d,f\}, & \{b,c,d,f\}, & \{a,c,d,g\}, & \{b,c,d,g\}, & \{a,c,d,h\}, & \{b,c,d,h\},\\
\{c,d,e,f\}, & \{c,d,e,g\}, & \{c,d,e,h\}, & \{a,c,e,f\}, & \{a,d,e,f\}, & \{b,c,e,f\},\\
\{b,d,e,f\}, & \{a,c,e,g\}, & \{a,d,e,g\}, & \{b,c,e,g\}, & \{b,d,e,g\}, & \{a,c,e,h\},\\
\{a,d,e,h\}, & \{b,c,e,h\}, & \{b,d,e,h\}, & \{a,c,f,h\}, & \{a,d,f,h\}, & \{b,c,f,h\},\\
\{b,d,f,h\}, & \{a,c,g,h\}, & \{a,d,g,h\}, & \{b,c,g,h\}, & \{b,d,g,h\}, & \{c,d,f,h\},\\
\{c,d,g,h\}. & & & & &
\end{array}$$

We see that even a small graph can have many spanning trees. Thus we seek a systematic method for enumerating these trees.

We are going to derive a recursion involving the quantities $\tau(G)$. For this purpose, we need to adapt the ideas of *deleting an edge* and *collapsing an edge* (see 3.98) from simple graphs to general graphs. Since loop edges are never involved in spanning trees, we will only consider graphs without loops. Suppose we are given a graph $G = (V, E, \epsilon)$ and a fixed edge $z \in E$ with endpoints $u, v \in V$. To *delete* z from G, we replace E by $E' = E \sim \{z\}$ and replace ϵ by the restriction of ϵ to E'. To *collapse* the edge z, we act as follows: (i) delete z and any other edges linking u to v; (ii) replace V by $(V \sim \{u, v\}) \cup \{w\}$, where w is a new vertex; (iii) for each edge $y \in E$ that has exactly one endpoint in the set $\{u, v\}$, modify $\epsilon(y)$ by replacing this endpoint with the new vertex w.

3.107. Example. Let G be the graph shown in Figure 3.22. Figure 3.23 displays the graphs obtained from G by deleting edge f and collapsing edge f.

3.108. Theorem: Spanning Tree Recursion. Let $G = (V, E, \epsilon)$ be a graph, and let $z \in E$ be a fixed edge. Let G_0 be the graph obtained from G by deleting z. Let G_1 be the graph obtained from G by collapsing z. Then

$$\tau(G) = \tau(G_0) + \tau(G_1).$$

The initial conditions are: $\tau(G) = 0$ if G is not connected, and $\tau(G) = 1$ if G is a tree with vertex set V.

Proof. For every graph K, let $Sp(K)$ be the set of all spanning trees of K, so $\tau(K) = |Sp(K)|$. Fix the graph G and the edge z. Let X be the set of trees in $Sp(G)$ that do not use the edge z, and let Y be the set of trees in $Sp(G)$ that do use the edge z. $Sp(G)$ is the disjoint union of X and Y, so $\tau(G) = |X| + |Y|$ by the sum rule. Now, it follows from the definition of edge-deletion that the set X is precisely the set $Sp(G_0)$, so $|X| = \tau(G_0)$. To complete the proof, we need to show that $|Y| = \tau(G_1)$. It suffices to define a bijection $F : Y \to Sp(G_1)$.

Suppose $T \in Y$ is a spanning tree of G that uses the edge z with endpoints u, v. Define $F(T)$ to be the graph obtained from T by collapsing the edge z; this graph is a subgraph of G_1. Let n be the number of vertices of G; then T is a connected graph with $n - 1$ edges, one of which is z. It is routine to check that $F(T)$ is still connected. Furthermore, since T is a tree, z is the only edge in T between u and v. It follows from the definition of collapsing that $F(T)$ has exactly $n - 2$ edges. Since G_1 has $n - 1$ vertices, it follows that $F(T)$ is a spanning tree of G_1. We see also that the edge set of $F(T)$ is precisely the edge set of T with z removed. So far, we have shown that F is a well-defined function mapping Y into $Sp(G_1)$.

Next we define a map $H : Sp(G_1) \to Y$ that will be the two-sided inverse of F. Given $U \in Sp(G_1)$ with edge set $E(U)$, let $H(U)$ be the unique subgraph of G with vertex set V and edge set $E(U) \cup \{z\}$. We must check that $H(U)$ does lie in the claimed codomain Y. First, $H(U)$ is a subgraph of G with $n - 1$ edges, one of which is the edge z. Furthermore, one may check that $H(U)$ is connected, since walks in U can be expanded using the edge z if needed to give walks in $H(U)$. Therefore, $H(U)$ is a spanning tree of G using z, and so $H(U) \in Y$. Since F removes z from the edge set while H adds it back, F and H are two-sided inverses of each other. Hence both are bijections, and the proof is complete. □

3.109. Example. Let us use the graphs in Figures 3.22 and 3.23 to illustrate the proof of the spanning tree recursion, taking $z = f$. The graph G_0 on the left of Figure 3.23 has 19 spanning trees; they are precisely the trees listed in 3.106 that do not use the edge f. Applying F to each of the remaining 12 spanning trees on the list produces the following subgraphs of G_1 (specified by their edge sets):

$$\begin{array}{llllll} \{a,c,d\}, & \{b,c,d\}, & \{c,d,e\}, & \{a,c,e\}, & \{a,d,e\}, & \{b,c,e\}, \\ \{b,d,e\}, & \{a,c,h\}, & \{a,d,h\}, & \{b,c,h\}, & \{b,d,h\}, & \{c,d,h\}. \end{array}$$

These are precisely the spanning trees of G_1.

Next, we illustrate the calculation of $\tau(G)$ using the recursion. We first delete and collapse edge f, producing the graphs G_0 and G_1 shown in Figure 3.23. We know that $\tau(G) = \tau(G_0) + \tau(G_1)$. Deletion of edge g from G_0 produces a new graph G_2 (Figure 3.24), while collapsing g in G_0 leads to another copy of G_1. So far, we have $\tau(G) = 2\tau(G_1) + \tau(G_2)$. Continuing to work on G_1, we see that deleting (resp. collapsing) edge h leads to the graph G_3 (resp. G_4) in Figure 3.24. On the other hand, deleting h from G_2 leaves a disconnected graph (which can be discarded), while collapsing h from G_2 produces another copy of G_3. Now we have $\tau(G) = 3\tau(G_3) + 2\tau(G_4)$. Deleting edge e from G_3 gives a graph that has two spanning trees (by inspection), while collapsing e in G_3 leads to G_4 again. So

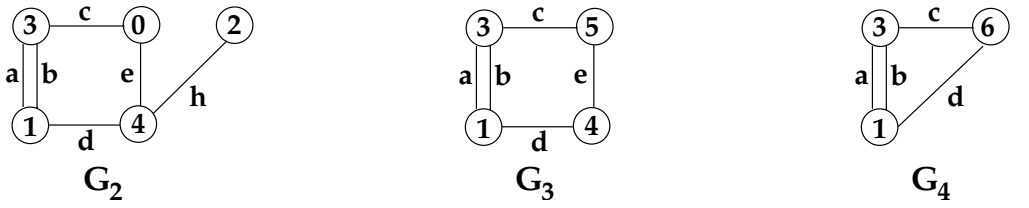


FIGURE 3.24
Auxiliary graphs used in the computation of $\tau(G)$.

$\tau(G) = 3(2 + \tau(G_4)) + 2\tau(G_4) = 6 + 5\tau(G_4)$. Finally, deletion of d from G_4 leaves a graph with two spanning trees, while collapsing d produces a graph with three spanning trees. We conclude that $\tau(G_4) = 5$, so $\tau(G) = 6 + 25 = 31$, in agreement with the enumeration in 3.106.

Next we extend the preceding discussion to rooted spanning trees in digraphs.

3.110. Definition: Rooted Spanning Trees. Let $G = (V, E, \epsilon)$ be a digraph, and let $v_0 \in V$. A *spanning tree of G rooted at v_0* is a rooted tree T with root v_0 and vertex set V such that T (without the loop at v_0) is a subgraph of G. Let $\tau(G, v_0)$ be the number of spanning trees of G rooted at v_0.

The notions of edge deletion and contraction extend in a natural way to digraphs. This given, we have the following recursion for counting rooted spanning trees.

3.111. Theorem: Rooted Spanning Tree Recursion. Let v_0 be a fixed vertex in a digraph G, and let z be a fixed edge leading into v_0. Let G_1 be the digraph obtained from G by deleting z. Let G_2 be the digraph obtained from G by collapsing z, and let the new "collapsed" vertex in G_2 be v_0'. Then

$$\tau(G, v_0) = \tau(G_1, v_0) + \tau(G_2, v_0').$$

Proof. We modify the proof of 3.108. As before, the two terms on the right side count rooted spanning trees of G that do not contain z or do contain z. The reader should check that if T is a rooted spanning tree using the edge z, then the graph obtained from T by collapsing z is a rooted spanning tree of G_2 rooted at v_0'. Similarly, adding z to the edge set of a rooted spanning tree of G_2 rooted at v_0' produces a rooted spanning tree of G rooted at v_0. □

3.112. Remark. Our results for counting (undirected) spanning trees are special cases of the corresponding results for rooted spanning trees. For, given a graph $G = (V, E, \epsilon)$, consider the associated digraph obtained by replacing each $e \in E$ by two directed edges going in opposite directions. Arguing as in 3.74, we see that there is a bijection between the set of rooted spanning trees of this digraph rooted at any given vertex $v_0 \in V$ and the set of spanning trees of G. In the sequel, we shall only treat the case of digraphs.

3.17 Matrix-Tree Theorem

There is a remarkable determinant formula for the number of rooted spanning trees of a digraph. The formula uses the following modified version of the adjacency matrix of the digraph.

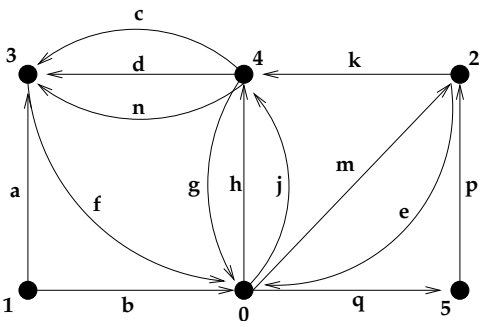


FIGURE 3.25
Digraph used to illustrate the matrix-tree theorem.

3.113. Definition: Laplacian Matrix of a Digraph. Let G be a loopless digraph on the vertex set $V = \{v_0, v_1, \ldots, v_n\}$. The *Laplacian matrix* of G is the matrix $L = (L_{ij} : 0 \leq i, j \leq n)$ such that $L_{ii} = \text{outdeg}(v_i)$ and L_{ij} is the negative of the number of edges from v_i to v_j in G. We let L_0 be the $n \times n$ matrix obtained by erasing the row and column of L corresponding to v_0. The matrix $L_0 = L_0(G)$ is called the *truncated Laplacian matrix of* G (relative to v_0).

3.114. Matrix-Tree Theorem. With the notation of the preceding definition, we have

$$\tau(G, v_0) = \det(L_0(G)).$$

We prove the theorem after considering two examples.

3.115. Example. Let G be the digraph associated to the undirected graph in Figure 3.22. In this case, L_{ii} is the degree of vertex i in the undirected graph, and L_{ij} is minus the number of undirected edges between i and j. So

$$L = \begin{bmatrix} 4 & 0 & -2 & -1 & -1 \\ 0 & 3 & 0 & -2 & -1 \\ -2 & 0 & 3 & 0 & -1 \\ -1 & -2 & 0 & 3 & 0 \\ -1 & -1 & -1 & 0 & 3 \end{bmatrix}.$$

Striking out the row and column corresponding to vertex 0 leaves

$$L_0 = \begin{bmatrix} 3 & 0 & -2 & -1 \\ 0 & 3 & 0 & -1 \\ -2 & 0 & 3 & 0 \\ -1 & -1 & 0 & 3 \end{bmatrix}.$$

We compute $\det(L_0) = 31$, which agrees with our earlier calculation of $\tau(G)$.

3.116. Example. Consider the digraph G shown in Figure 3.25. We compute

$$L = \begin{bmatrix} 4 & 0 & -1 & 0 & -2 & -1 \\ -1 & 2 & 0 & -1 & 0 & 0 \\ -1 & 0 & 2 & 0 & -1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -3 & 4 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}, \qquad L_0 = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -3 & 4 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix},$$

and $\det(L_0) = 16$. So G has 16 spanning trees rooted at 0, as one may confirm by direct enumeration. We will use the matrix L_0 as a running example in the proof below.

3.117. Proof of the Matrix-Tree Theorem. Write $L_0 = L_0(G)$. First we prove that $\tau(G, v_0) = \det(L_0)$ in the case where $\text{indeg}(v_0) = 0$. If v_0 is the only vertex of G, then $\tau(G, v_0) = 1$ and $\det(L_0) = 1$ by the convention that the determinant of a 0×0 matrix is 1. Otherwise, $\tau(G, v_0)$ is zero, and L_0 is a nonempty matrix. Using the condition $\text{indeg}(v_0) = 0$ and the definition of L_0, one sees that every row of L_0 sums to zero. Therefore, letting $\vec{u}$ be a column vector of n ones, we have $L_0\vec{u} = \vec{0}$, so that L_0 is singular and $\det(L_0) = 0$.

For the general case, we argue by induction on the number of edges in G. The case where G has no edges is covered by the previous paragraph. The only case left to consider occurs when $\text{indeg}(v_0) > 0$. Let e be a fixed edge in G that leads from some v_i to v_0. Let G_1 be the graph obtained from G by deleting e, and let G_2 be the graph obtained from G by collapsing e. Both graphs have fewer edges than G, so the induction hypothesis tells us that

$$\tau(G_1, v_0) = \det(L_0(G_1)) \text{ and } \tau(G_2, v_0') = \det(L_0(G_2)), \tag{3.4}$$

where v_0' is the new vertex created after collapsing e. Using 3.111, we conclude that

$$\tau(G, v_0) = \det(L_0(G_1)) + \det(L_0(G_2)). \tag{3.5}$$

Next, let us evaluate the determinant $\det(L_0(G))$. We will use the fact that the determinant of a matrix is a *linear* function of each row of the matrix. More precisely, for a fixed matrix A and row index i, let $A[y]$ denote the matrix A with the ith row replaced by the row vector y; then $\det(A[y + z]) = \det(A[y]) + \det(A[z])$ for all y, z. This linearity property can be proved directly from the definition of the determinant (see 9.37 and 9.45 below). To apply this result, write the ith row of $L_0 = L_0(G)$ in the form $y + z$, where $z = (0, 0, \ldots, 1, 0, \ldots, 0)$ has a one in position i. Then

$$\det(L_0(G)) = \det(L_0[y]) + \det(L_0[z]). \tag{3.6}$$

For example, if G is the digraph in Figure 3.25 and e is the edge from 2 to 0 (so $i = 2$), then $y = (0, 1, 0, -1, 0)$, $z = (0, 1, 0, 0, 0)$,

$$L_0[y] = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -3 & 4 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}, \qquad L_0[z] = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -3 & 4 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}.$$

Comparing equations (3.5) and (3.6), we see that it suffices to prove $\det(L_0(G_1)) = \det(L_0[y])$ and $\det(L_0(G_2)) = \det(L_0[z])$.

How does the removal of e from G affect $L(G)$? Answer: The i, i-entry drops by 1, while the $i, 0$-entry increases by 1. Since the zeroth column is ignored in the truncated Laplacian, we see that we can obtain $L_0(G_1)$ from $L_0(G)$ by decrementing the i, i-entry by 1. In other words, $L_0(G_1) = L_0[y]$, and hence $\det(L_0(G_1)) = \det(L_0[y])$.

Next, let us calculate $\det(L_0[z])$ by expanding the determinant along row i. The only nonzero entry in this row is the 1 in the diagonal position, so $\det(L_0[z]) = (-1)^{i+i}\det(M) = \det(M)$, where M is the matrix obtained from $L_0[z]$ (or equivalently, from L_0) by erasing row i and column i. In our running example,

$$M = \begin{bmatrix} 2 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We claim that $M = L_0(G_2)$, which will complete the proof. Consider the k,j-entry of M, where $k,j \in \{0,1,\ldots,n\} \sim \{0,i\}$. If $k = j$, this entry is $\text{outdeg}_G(v_j)$, which equals $\text{outdeg}_{G_2}(v_j)$ because v_j is not v_0, v_i, or v_0'. For the same reason, if $k \neq j$, the k,j-entry of M is minus the number of edges from v_k to v_j, which is the same in G and G_2.

3.18 Eulerian Tours

3.118. Definition: Eulerian Tours. Let $G = (V, E, \epsilon)$ be a digraph. An *Eulerian tour* in G is a walk $W = (v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n)$ such that W visits every vertex in V, and W uses every edge in E exactly once. Such a tour is called *closed* iff $v_n = v_0$.

3.119. Example. Consider the digraph G shown in Figure 3.26. Here is one closed Eulerian tour of G:

$$W_1 = (0, m, 2, l, 5, e, 1, a, 3, c, 4, b, 3, d, 5, f, 4, g, 5, k, 0, i, 4, h, 5, j, 0).$$

To specify the tour, it suffices to list only the edges in the tour. For instance, here is the edge sequence of another closed Eulerian tour of G:

$$W_2 = (i, g, e, a, d, f, b, c, h, j, m, l, k).$$

3.120. Example. Consider the digraph G shown in Figure 3.2. This graph does not have any closed Eulerian tours, since there is no way to reach vertex 6 from the other vertices. Even if we delete vertex 6 from the graph, there will still be no closed Eulerian tours. For, there is no way that a tour can use both edges leaving vertex 2, since only one edge enters vertex 2.

The previous example indicates two necessary conditions for a digraph to have a closed Eulerian tour: the digraph must be connected, and also *balanced* in the sense that $\text{indeg}(v) = \text{outdeg}(v)$ for every vertex v. We now show that these necessary conditions are also sufficient to guarantee the existence of a closed Eulerian tour.

3.121. Theorem: Existence of Closed Eulerian Tours. A digraph $G = (V, E, \epsilon)$ has a closed Eulerian tour iff G is connected and balanced.

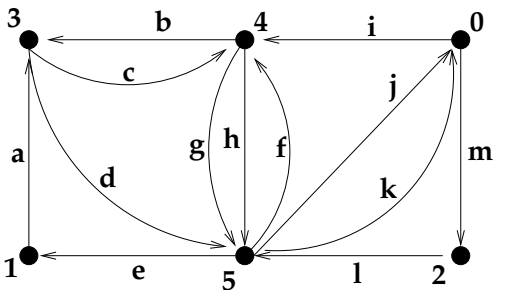


FIGURE 3.26
Digraph used to illustrate Eulerian tours.

Proof. First suppose G has a closed Eulerian tour W starting at v_0. Since W visits every vertex, we can obtain a walk from any vertex to any other vertex by following suitable edges of W. So G is connected. Next, let v be any vertex of G. The walk W arrives at v via an incoming edge exactly as often as the walk leaves v via an outgoing edge; this is true even if $v = v_0$. Since the walk uses every edge exactly once, it follows that $\text{indeg}(v) = \text{outdeg}(v)$.

Conversely, assume that G is connected and balanced. Let $W = (v_0, e_1, v_1, \ldots, e_n, v_n)$ be a walk of maximum length in G that never repeats an edge. We claim that $v_n = v_0$. For, if not, W enters vertex v_n one more time than it leaves v_n. Since $\text{indeg}(v_n) = \text{outdeg}(v_n)$, there must be an outgoing edge from v_n that has not been used by W. So we could use this edge to extend W, contradicting maximality. Next, we claim that W uses *every* edge of G. If not, let e be an edge not used by W. Since G is connected, we can find such an edge that is incident to one of the vertices v_i visited by W. Since $v_n = v_0$, we can cyclically shift the walk W to get a new walk $W' = (v_i, e_{i+1}, v_{i+1}, \ldots, e_n, v_n = v_0, e_1, \ldots, e_i, v_i)$ that starts and ends at v_i. By adding the edge e to the beginning or end of this walk (depending on its direction), we could again produce a longer walk than W with no repeated edges, violating maximality. Finally, W must visit every vertex of G, since W uses every edge of G and (unless G has one vertex and no edges) every vertex has an edge leaving it. □

Our goal in the rest of this section is to prove the following formula for the number of closed Eulerian tours in G starting at a given vertex v_0. Recall that $\tau(G, v_0)$ is the number of rooted spanning trees of G rooted at v_0.

3.122. Theorem: Counting Eulerian Tours. Let $G = (V, E, \epsilon)$ be a connected, balanced digraph. For each $v_0 \in V$, the number of closed Eulerian tours of G starting at v_0 is

$$\tau(G, v_0) \cdot \text{outdeg}(v_0)! \cdot \prod_{v \neq v_0} (\text{outdeg}(v) - 1)!. \tag{3.7}$$

Let $\{v_0, v_1, \ldots, v_n\}$ be the vertex set of G. Let X be the set of all closed Eulerian tours of G starting at v_0. Let $\text{SpTr}(G, v_0)$ be the set of spanning trees of G rooted at v_0. Let Y be the set of all tuples $(T, w_0, w_1, w_2, \ldots, w_n)$ where: $T \in \text{SpTr}(G, v_0)$; w_0 is a permutation of all the edges leaving v_0; and, for $1 \leq i \leq n$, w_i is a permutation of those edges leaving v_i other than the unique outgoing edge from v_i that belongs to T (see 3.42). By the product rule, the cardinality of Y is given by the right side of (3.7). So it will suffice to define a bijection $f : X \to Y$.

Given an Eulerian tour $W \in X$, define $f(W) = (T, w_0, \ldots, w_n)$ as follows. For each i between 0 and n, let w_i' be the permutation of *all* edges leading out of v_i, taken in the order in which they occur in the walk W. Call w_i' the *departure word* of vertex v_i. Next, set $w_0 = w_0'$ and for $i > 0$, let w_i be the word w_i' with the last symbol erased. Finally, let T be the subgraph of G whose edges are given by the last symbols of $w_1', \ldots, w_n'$, augmented by a loop edge at v_0. It is not immediately evident that $T \in \text{SpTr}(G, v_0)$; we will prove this shortly.

Next we define a map $g : Y \to X$ that will be the two-sided inverse of f. Fix $(T, w_0, \ldots, w_n) \in Y$. For every $i > 0$, form w_i' by appending the unique edge of T leaving v_i to the end of the word w_i; let $w_0' = w_0$. Starting at v_0, we use the words w_i' to build a walk through G, one edge at a time, as follows. If we are currently at some vertex v_i, use the next unread symbol in w_i' to determine which edge to follow out of v_i. Repeat this process until the walk reaches a vertex in which all the outgoing edges have already been used. The resulting walk W is $g(T, w_0, \ldots, w_n)$. The edges occurring in W are pairwise distinct, but it is not immediately evident that W must use *all* edges of G; we will prove this shortly.

Once we check that f and g map into their stated codomains, the definitions just given show that $f \circ g$ and $g \circ f$ are both identity maps. Before proving that f maps into Y and g maps into X, let us consider an example.

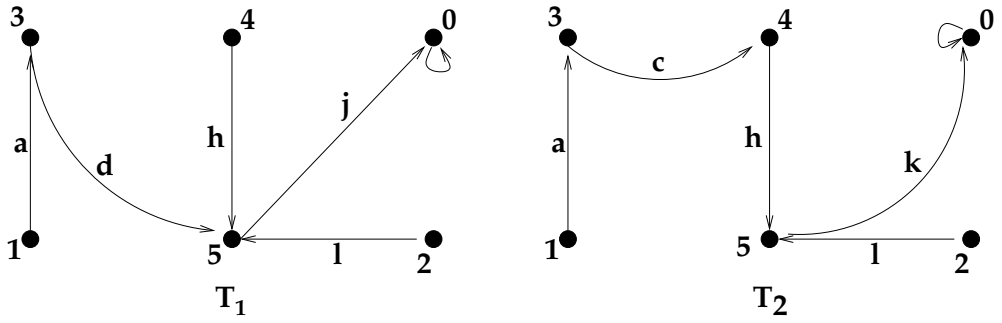


FIGURE 3.27
Rooted spanning trees associated to Eulerian tours.

3.123. Example. We continue the analysis of Eulerian tours in the digraph G from 3.119. The walk W_1 in that example has departure words $w_0' = mi$, $w_1' = a$, $w_2' = l$, $w_3' = cd$, $w_4' = bgh$, and $w_5' = efkj$. Therefore,

$$f(W_1) = (T_1, mi, \cdot, \cdot, c, bg, efk),$$

where $\cdot$ denotes an empty word and T_1 is the graph shown on the left in Figure 3.27. Similarly, for W_2 we compute $w_0' = im$, $w_1' = a$, $w_2' = l$, $w_3' = dc$, $w_4' = gbh$, $w_5' = efjk$, and

$$f(W_2) = (T_2, im, \cdot, \cdot, d, gb, efj).$$

Let us now calculate $g((T_1, im, \cdot, \cdot, c, bg, fke))$. First, we use the edges of T_1 to recreate the departure words $w_0' = im$, $w_1' = a$, $w_2' = l$, $w_3' = cd$, $w_4' = bgh$, and $w_5' = fkej$. We then use these words to guide our tour through the graph. We begin with $0, i, 4$, since i is the first letter of w_0'. Consulting w_4' next, we follow edge b to vertex 3, then edge c to vertex 4, then edge g to vertex 5, and so on. We obtain the tour

$$W_3 = (0, i, 4, b, 3, c, 4, g, 5, f, 4, h, 5, k, 0, m, 2, l, 5, e, 1, a, 3, d, 5, j, 0).$$

Similarly, the reader may check that

$$g((T_2, mi, \cdot, \cdot, d, bg, jfe)) = (m, l, j, i, b, d, f, g, e, a, c, h, k).$$

To complete the proof of 3.122, we must prove two things. First, to show that $f(W) \in Y$ for all $W \in X$, we must show that the digraph T obtained from the last letters of the departure words w_i' $(i > 0)$ is a rooted spanning tree of G rooted at v_0. Since W visits every vertex of G, the definition of T shows that $\text{outdeg}_T(v_i) = 1$ for all $i \geq 0$. We need only show that T has no cycles other than the loop at v_0 (see 3.42). We can view the tour W as a certain permutation of all the edges in G. Let us show that if e, h are two non-loop edges in T with $\epsilon(e) = (x, y)$ and $\epsilon(h) = (y, z)$, then e must precede h in the permutation W. Note that y cannot be v_0, since the only outgoing edge from v_0 in T is a loop edge. Thus, when the tour W uses the edge e to enter y, the following edge in the tour exists and is an outgoing edge from y. Since h is, by definition, the *last* such edge used by the tour, e must precede h in the tour. Now suppose $(z_0, e_1, z_1, \ldots, e_n, z_n)$ is a cycle in T that is not the 1-cycle at v_0. Using the previous remark repeatedly, we see that e_i precedes e_{i+1} in W for all i, and also e_n precedes e_1 in W. These statements imply that e_1 precedes itself in W, which is absurd. We conclude that $f(W) \in Y$.

Second, we must show that g maps Y into X. Fix $(T, w_0, \ldots, w_n) \in Y$ and $W = g(T, w_0, \ldots, w_n)$, and let w_i' be the departure words constructed from T and the w_i's. We know from the definition of g that W is a walk in G starting at v_0 that never repeats an edge. We must show that W ends at v_0 and uses every edge in G. Suppose, at some stage in the construction of W, that W has just reached v_i for some $i > 0$. Then W has entered v_i one more time than it has left v_i. Since G is balanced, there must exist an unused outgoing edge from v_i. This edge corresponds to an unused letter in w_i'. So W does not end at v_i. The only possibility is that W ends at the starting vertex v_0.

To prove that W uses every edge *of* G, we claim that it is enough to prove that W uses every non-loop edge *of* T. For, consider a vertex $v \neq v_0$ of G. If W uses the unique outgoing edge from v that is part of T, then W must have previously used all other outgoing edges from v, by definition of W. Since W ends at v_0, W certainly uses all outgoing edges from v_0. All edges are accounted for in this way, proving the claim.

Finally, to get a contradiction, assume that some edge e in T from x to y is not used by W. Since T is a rooted tree rooted at v_0, we can choose such an e so that the distance from y to v_0 through edges in T is minimal. If $y \neq v_0$, minimality implies that the unique edge leading out of y in T *does* belong to W. Then, as noted in the last paragraph, every outgoing edge from y in G is used in W. Since G is balanced, every incoming edge into y in G must also appear in W, contradicting the assumption that e is not used by W. On the other hand, if $y = v_0$, we see similarly that W uses every outgoing edge from y in G and hence every incoming edge to y in G. Again, this contradicts the assumption that e is not in W. This completes the proof of 3.122.

Summary

Table 3.1 contains brief definitions of the terminology from graph theory used in this chapter.

- *Facts about Matrix Multiplication.* If $A_1, \ldots, A_s$ are matrices such that A_t is $n_{t-1} \times n_t$, then the i, j-entry of the product $A_1 A_2 \cdots A_s$ is

$$\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \cdots \sum_{k_{s-1}=1}^{n_{s-1}} A_1(i, k_1) A_2(k_1, k_2) A_3(k_2, k_3) \cdots A_s(k_{s-1}, j).$$

  If $A^s = 0$ (i.e., A is nilpotent), then $I - A$ is invertible, and

$$(I - A)^{-1} = I + A + A^2 + A^3 + \cdots + A^{s-1}.$$

  This formula applies (with $s = n$) when A is a strictly upper or lower triangular $n \times n$ matrix.

- *Adjacency Matrices and Walks.* Given a graph or digraph G with vertex set $\{v_1, \ldots, v_n\}$, the adjacency matrix of G is the matrix A such that $A(i, j)$ is the number of edges from v_i to v_j in G. For all $s \geq 0$, $A^s(i, j)$ is the number of walks in G of length s from v_i to v_j. G is a DAG iff $A^n = 0$, in which case A will be strictly lower-triangular under a suitable ordering of the vertices. When G is a DAG, $(I - A)^{-1}(i, j)$ is the total number of paths (or walks) from v_i to v_j.

- *Degree-Sum Formulas.* For a digraph $G = (V, E, \epsilon)$,

$$\sum_{v \in V} \text{indeg}_G(v) = |E| = \sum_{v \in V} \text{outdeg}_G(v).$$

TABLE 3.1
Terminology used in graph theory.

Term	Brief Definition
graph	(V, E, ϵ) where $\epsilon(e) = \{v, w\}$ means edge e has endpoints v, w
digraph	(V, E, ϵ) where $\epsilon(e) = (v, w)$ means edge e goes from v to w
simple graph	graph with no loops or multiple edges
simple digraph	digraph with no multiple edges
$G \cong H$	G becomes H under suitable renaming of vertices and edges
walk	$(v_0, e_1, v_1, \ldots, e_s, v_s)$ where each e_i is an edge from v_{i-1} to v_i
closed walk	walk starts and ends at same vertex
path	walk visiting distinct vertices
cycle	closed walk visiting distinct vertices and edges, except at end
DAG	digraph with no cycles
$\text{indeg}_G(v)$	number of edges leading to v in digraph G
$\text{outdeg}_G(v)$	number of edges leading from v in digraph G
$\deg_G(v)$	number of edges incident to v in graph G (loops count as 2)
isolated vertex	vertex of degree zero
leaf	vertex of degree one
functional digraph	simple digraph with $\text{outdeg}(v) = 1$ for all vertices v
cyclic vertex	vertex in functional digraph that belongs to a cycle
rooted tree	functional digraph with a unique cyclic vertex (the root)
G is connected	for all $u, v \in V(G)$, there is a walk in G from u to v
cut-edge of G	edge belonging to no cycle of the graph G
forest	graph with no cycles
acyclic graph	graph with no cycles
tree	connected graph with no cycles
proper coloring	map $f : V(G) \to C$ assigning unequal colors to adjacent vertices
$\chi_G(x)$	number of proper colorings of G using x available colors
chromatic number	least x with $\chi_G(x) > 0$
subgraph of G	graph G' with $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$ (same endpoints)
induced subgraph	subgraph G' where all edges in G with ends in $V(G')$ are kept
spanning tree of G	subgraph of G that is a tree using all vertices
$\tau(G)$	number of spanning trees of G
rooted spanning tree	rooted tree using all vertices of a digraph
$\tau(G, v_0)$	number of rooted spanning trees of G with root v_0
Eulerian tour	walk visiting each vertex that uses every edge once

For a graph $G = (V, E, \epsilon)$,

$$\sum_{v \in V} \deg_G(v) = 2|E|.$$

- *Functional Digraphs.* For a finite set X, every function $f : X \to X$ has an associated functional digraph with vertex set X and edge set $\{(x, f(x)) : x \in X\}$. Every functional digraph decomposes uniquely into one or more disjoint cycles together with disjoint rooted trees rooted at the vertices on these cycles. For each vertex x_0 in a functional digraph, there exist unique walks of each length k starting at x_0, which are found by repeatedly following the unique outgoing edge from the current vertex. Such walks eventually reach a cycle in the functional digraph.

- *Cycle Structure of Permutations.* For X finite, a map $f : X \to X$ is a bijection iff the functional digraph of f is a disjoint union of directed cycles. The signless Stirling number of the first kind, $s'(n, k)$, counts the number of bijections f on an n-element set such that the functional digraph of f has k cycles. We have

$$s'(n, k) = s'(n-1, k-1) + (n-1)s'(n-1, k) \qquad (0 < k < n).$$

- *Connectedness and Components.* The vertex set of any graph or digraph G is the disjoint union of connected components. Two vertices belong to the same component iff each vertex is reachable from the other by a walk. G is connected iff there is only one component iff for all $u, v \in V(G)$ there exists at least one *path* from u to v in G. Deleting a cut-edge splits a component of G in two, whereas deleting a non-cut-edge has no effect on components.

- *Forests.* A graph G is a forest (acyclic) iff G has no loops and for each $u, v \in V(G)$, there is at most one path from u to v. A forest with n vertices and k edges has $n - k$ components.

- *Trees.* The following conditions on an n-vertex simple graph G are equivalent and characterize trees: (a) G is connected with no cycles; (b) G is connected with at most $n-1$ edges; (c) G is acyclic with at least $n-1$ edges; (d) for all $u, v \in V(G)$, there exists a unique path in G from u to v. An n-vertex tree has $n-1$ edges and (for $n > 1$) at least two leaves. Pruning any leaf from a tree gives another tree with one less vertex and one less edge.

- *Tree Enumeration Results.* There are n^{n-2} trees with vertex set $\{1, 2, \ldots, n\}$. There are n^{n-2} rooted trees on this vertex set rooted at 1. For $d_1 + \cdots + d_n = 2(n-1)$, there are $\binom{n-2}{d_1-1,\ldots,d_n-1}$ trees on this vertex set with $\deg(j) = d_j$ for all j. Bijective proofs of these facts use the following ideas:

  - Functions on $\{1, 2, \ldots, n\}$ fixing 1 and n correspond to rooted trees by arranging the cycles of the functional digraph in a certain order, breaking "back edges," and linking the cycles to get a tree (see Figures 3.7 and 3.8).
  - Trees correspond to rooted trees by directing each edge of the tree towards the desired root vertex.
  - Trees with $\deg(j) = d_j$ correspond to words in $\mathcal{R}(1^{d_1-1} \cdots n^{d_n-1})$ by repeatedly pruning the largest leaf and appending the leaf's neighbor to the end of the word.

- *Terms and Ordered Trees.* For every term T, there exist a unique integer $n \geq 0$ and unique terms $T_1, \ldots, T_n$ such that $T = nT_1T_2 \cdots T_n$. A word $w_1 \cdots w_s$ is a term iff $w_1 + \cdots + w_i - i \geq 0$ for all $i < s$ and $w_1 + \cdots + w_s - s = -1$. No proper prefix of a term is a term. Terms correspond bijectively to ordered trees.

- *Lists of Terms and Ordered Forests.* Every list of terms has the form $T_1 \cdots T_n$ for some unique integer $n \geq 0$ and unique terms $T_1, \ldots, T_n$. A word $w_1 \cdots w_s$ is a list of n terms iff $w_1 + \cdots + w_i - i > -n$ for all $i < s$ and $w_1 + \cdots + w_s - s = -n$. Lists of terms correspond bijectively to ordered forests.

- *Cycle Lemma and Enumeration of Lists of Terms.* For a word $w = w_1 \cdots w_s$ with $w_1 + \cdots + w_s - s = -n$, there exist exactly n cyclic shifts of w that are lists of n terms. Consequently, the number of lists of n terms using k_i copies of i (for $0 \leq i \leq t$) is

$$\frac{n}{s}\binom{s}{k_0, k_1, \ldots, k_s},$$

  where $s = \sum_{i=0}^{t} k_i$ and $k_0 = n + \sum_{i=1}^{t}(i-1)k_i$.

- *Chromatic Polynomials.* For any edge e in a simple graph G, the chromatic function of G satisfies the recursion $\chi_G = \chi_{G \sim \{e\}} - \chi_{G_e}$, where $G \sim \{e\}$ is G with e deleted, and G_e is G with e collapsed. It follows that $\chi_G(x)$ is a polynomial function of x. The signed Stirling numbers of the first kind, $s(n,k)$, are the coefficients in the chromatic polynomial for an n-vertex graph with an edge between each pair of vertices.

- *Spanning Tree Recursion.* For any edge e in a graph G, the number $\tau(G)$ of spanning trees of G satisfies the recursion $\tau(G) = \tau(G \sim \{e\}) + \tau(G_e)$, where $G \sim \{e\}$ is G with e deleted, and G_e is G with e collapsed. A similar recursion holds for rooted spanning trees of a digraph.

- *Matrix-Tree Theorem.* Given a digraph G and $v_0 \in V(G)$, let $L_{ii} = \mathrm{outdeg}_G(v_i)$, let $-L_{ij}$ be the number of edges from i to j in G, and let L_0 be the matrix obtained from (L_{ij}) by erasing the row and column indexed by v_0. Then $\det(L_0)$ is the number $\tau(G, v_0)$ of rooted spanning trees of G with root v_0.

- *Eulerian Tours.* A digraph G has a closed Eulerian tour iff G is connected and balanced (indegree equals outdegree at every vertex). In this case, the number of such tours starting at v_0 is

$$\tau(G, v_0) \cdot \mathrm{outdeg}_G(v_0)! \cdot \prod_{v \neq v_0} (\mathrm{outdeg}_G(v) - 1)!.$$

  The proof associates to each tour a rooted spanning tree built from the last departure edge from each vertex, together with (truncated) departure words for each vertex giving the order in which the tour used the other outgoing edges.

Exercises

3.124. Draw pictures of the following simple graphs, which have the indicated nicknames.
(a) the *claw* $C = (\{1,2,3,4\}, \{\{1,2\},\{1,3\},\{1,4\}\})$;
(b) the *paw* $P = (\{1,2,3,4\}, \{\{1,2\},\{1,3\},\{1,4\},\{2,3\}\})$;

(c) the *kite* $K = (\{1,2,3,4\}, \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\}\})$;
(d) the *bull* $B = (\{1,2,3,4,5\}, \{\{1,2\},\{2,3\},\{1,3\},\{1,4\},\{2,5\}\})$.
(e) the *n-path* $P_n = (\{1,2,\ldots,n\}, \{\{i,i+1\} : 1 \leq i < n\})$.
(f) the *n-cycle* $C_n = (\{1,2,\ldots,n\}, \{\{i,i+1\} : 1 \leq i < n\} \cup \{\{1,n\}\})$, where $n \geq 3$.
(g) the *complete graph* $K_n = (\{1,2,\ldots,n\}, \{\{i,j\} : 1 \leq i < j \leq n\})$.

3.125. Let V be an n-element set. (a) How many simple graphs have vertex set V? (b) How many simple digraphs have vertex set V?

3.126. Let V and E be sets with $|V| = n$ and $|E| = m$. (a) How many digraphs have vertex set V and edge set E? (b) How many graphs have vertex set V and edge set E?

3.127. Let V be an n-element set. Define a bijection between the set of simple graphs with vertex set V and the set of symmetric, irreflexive binary relations on V. Conclude that simple graphs can be viewed as certain kinds of simple digraphs.

3.128. Let G, H, and K be graphs (resp. digraphs). (a) Prove $G \cong G$. (b) Prove $G \cong H$ implies $H \cong G$. (c) Prove $G \cong H$ and $H \cong K$ imply $G \cong K$. Thus, graph isomorphism is an equivalence relation on any given set of graphs (resp. digraphs).

3.129. Find all isomorphism classes of simple graphs with at most four vertices.

3.130. Find the adjacency matrices for the graphs in 3.124.

3.131. Let G be the simple graph in Figure 3.10. For $1 \leq k \leq 8$, find the number of walks in G from vertex 1 to vertex 10.

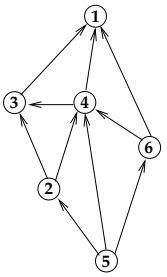
3.132. Let G be the graph in Figure 3.22. Find the number of walks in G of length 5 between each pair of vertices.

3.133. Let G be the digraph in Figure 3.25. Find the number of closed walks in G of length 10 that begin at vertex 0.

3.134. (a) Show that a graph G with a closed walk of odd length must have a cycle of odd length. (b) If G has a closed walk of even length, must G have a cycle?

3.135. Let G be a graph with adjacency matrix A. (a) Find a formula for the number of *paths* in G of length 2 from v_i to v_j. (b) Find a formula for the number of paths in G of length 3 from v_i to v_j.

3.136. Consider the DAG G shown here.



(a) Find all total orderings of the vertices for which the adjacency matrix of G is strictly lower-triangular. (b) How many paths in G go from vertex 5 to vertex 1?

3.137. An irreflexive, transitive binary relation on a set X is called a *strict partial order on* X. Given a strict partial order R on a finite set X, show that the simple digraph (X, R) is a DAG.

3.138. For each of the following sets X and strict partial orders R, draw the associated DAG (see 3.137) and calculate the number of paths from the smallest element to the largest element of the partially ordered set.
(a) $X = \{1, 2, 3, 4, 5\}$ under the ordering $1 < 2 < 3 < 4 < 5$.
(b) $X = \mathcal{P}(\{1, 2, 3\})$, and $(S, T) \in R$ iff $S \subsetneq T$.
(c) X is the set of positive divisors of 60, and $(a, b) \in R$ iff $a < b$ and a divides b.

3.139. Let $X = \{1, 2, \ldots, n\}$ ordered by $1 < 2 < \cdots < n$. In the associated DAG (see 3.137), how many paths go from 1 to n? Can you find a combinatorial (not algebraic) proof of your answer?

3.140. Let X be the set of subsets of $\{1, 2, \ldots, n\}$ ordered by (strict) set inclusion. In the associated DAG (see 3.137), how many paths go from $\emptyset$ to $\{1, 2, \ldots, n\}$?

3.141. Given a digraph G, construct a simple digraph H as follows. The vertices of H are the strong components of G. Given $C, D \in V(H)$ with $C \neq D$, there is an edge from C to D in H iff there exists $c \in C$ and $d \in D$ such that there is an edge from c to d in G. (a) Prove that H is a DAG. (b) Conclude that some strong component C of G has no incoming edges from outside C, and some strong component D has no outgoing edges. (c) Draw the DAGs associated to the digraph G_3 in Figure 3.1 and the functional digraph in Figure 3.5.

3.142. (a) Find the degree sequence for the graph in Figure 3.10, and verify 3.34 in this case. (b) Compute the indegrees and outdegrees at each vertex of the digraph in Figure 3.25, and verify 3.31 in this case.

3.143. Find necessary and sufficient conditions for a multiset $[d_1, d_2, \ldots, d_n]$ to be the degree sequence of a graph G.

3.144. Consider the cycle graph C_n defined in 3.124. (a) What is $\deg(C_n)$? (b) Show that any connected graph with the degree sequence in (a) must be isomorphic to C_n. (c) How many graphs with vertex set $\{1, 2, \ldots, n\}$ are isomorphic to C_n? (d) How many isomorphism classes of graphs have the same degree sequence as C_n? (e) How many isomorphism classes of *simple* graphs have the same degree sequence as C_n?

3.145. Consider the path graph P_n defined in 3.124. (a) What is $\deg(P_n)$? (b) Show that any connected graph with the degree sequence in (a) must be isomorphic to P_n. (c) How many graphs with vertex set $\{1, 2, \ldots, n\}$ are isomorphic to P_n? (d) How many isomorphism classes of graphs have the same degree sequence as P_n?

3.146. Find two simple graphs G and H with the smallest possible number of vertices, such that $\deg(G) = \deg(H)$ but $G \ncong H$.

3.147. Prove or disprove: there exists a simple graph G with more than one vertex such that the degree sequence $\deg(G)$ contains no repetitions.

3.148. Prove or disprove: there exists a graph G with no loops and more than one vertex such that the degree sequence $\deg(G)$ contains no repetitions.

3.149. Given a graph $G = (V, E, \epsilon)$, we can encode the endpoint function ϵ by a $|V| \times |E|$ matrix M, with rows indexed by V and columns indexed by E, such that $M(v, e)$ is 2 if e is a loop edge at v, 1 if e is a non-loop edge incident to v, and 0 otherwise. M is called the *incidence matrix* of G. Prove the degree-sum formula 3.34 by computing the sum of all entries of M in two ways.

3.150. Draw the functional digraphs associated to each of the following functions $f : X \to X$. For each digraph, find the set C of cyclic vertices and the set partition $\{S_v : v \in C\}$ described in 3.43. (a) $X = \{1, 2, 3, 4\}$, f is the identity map on X; (b) $X = \{0, 1, \ldots, 6\}$, $f(x) = (x^2 + 1) \bmod 7$; (c) $X = \{0, 1, \ldots, 12\}$, $f(x) = (x^2 + 1) \bmod 13$; (d) $X = \{0, 1, \ldots, 10\}$, $f(x) = 3x \bmod 11$; (e) $X = \{0, 1, \ldots, 11\}$, $f(x) = 4x \bmod 12$.

3.151. Let $X = \{0, 1, 2, \ldots, 9\}$. (a) Define $f : X \to X$ by setting $f(x) = (3x + 7) \bmod 10$. Draw the functional digraphs for f, f^{-1} and $f \circ f$. What is the smallest integer $k > 0$ such that $f \circ f \circ \cdots \circ f$ (k factors) is the identity map on X? (b) Define $g : X \to X$ by setting $g(x) = (2x + 3) \bmod 10$. Draw the functional digraphs for g and $g \circ g$.

3.152. Let X be a finite set, let $x_0 \in X$, and let $f : X \to X$ be any function. Recursively define $x_{m+1} = f(x_m)$ for all $m \geq 0$. Show that there exists $i > 0$ with $x_i = x_{2i}$.

3.153. Pollard-rho Factoring Algorithm. Suppose $N > 1$ is an integer. Let $X = \{0, 1, \ldots, N - 1\}$, and define $f : X \to X$ by $f(x) = (x^2 + 1) \bmod N$. (a) Show that the following algorithm always terminates and returns a divisor of N greater than 1. (Use 3.152.)

Step 1. Set $u = f(0)$, $v = f(f(0))$, and $d = \gcd(v - u, N)$.
Step 2. While $d = 1$: set $u = f(u)$, $v = f(f(v))$, and $d = \gcd(v - u, N)$.
Step 3. Return d.

(b) Trace the steps taken by this algorithm to factor $N = 77$ and $N = 527$.

3.154. Suppose X is a finite set of size k and $f : X \to X$ is a random function (so for all $x, y \in X$, $P(f(x) = y) = 1/k$, and these events are independent for different choices of x). Let $x_0 \in X$, define $x_{m+1} = f(x_m)$ for all $m \geq 0$, and let S be the least index such that $x_S = x_t$ for some $t < S$. (a) For each $s \geq 0$, find the exact probability that $S > s$. (b) Argue informally that the expected value of S is at most $2\sqrt{k}$. (c) Use (b) to argue informally that the expected number of gcd computations needed by the Pollard-rho factoring algorithm to find a divisor of a composite number N (see 3.153) is bounded above by $2N^{1/4}$.

3.155. Let V be an n-element set, and let $v_0 \notin V$. A function $f : V \to V$ is called *acyclic* iff all cycles in the functional digraph of f have length 1. Count these functions by setting up a bijection between the set of acyclic functions on V and the set of rooted trees on $V \cup \{v_0\}$ with root v_0.

3.156. How many bijections f on an 8-element set are such that the functional digraph of f has (a) five cycles; (b) three cycles; (c) one cycle?

3.157. Let X be an n-element set. Let Y be the set of all functional digraphs for bijections $f : X \to X$. How many equivalence classes does Y have under the equivalence relation of graph isomorphism (see 3.128)?

3.158. How many functional digraphs with vertex set $\{1, 2, \ldots, n\}$ have a_1 cycles of length 1, a_2 cycles of length 2, etc., where $\sum_i i a_i = n$?

3.159. Referring to the proof of 3.47, draw pictures of the set A of functions, the set B of trees, and the bijection $\phi : A \to B$ when $n = 4$.

3.160. Compute the rooted tree associated to the function below by the map ϕ in the proof of 3.47.

$$\begin{array}{lllll}
f(1) = 1; & f(2) = 19; & f(3) = 8; & f(4) = 30; & f(5) = 5; \\
f(6) = 15; & f(7) = 8; & f(8) = 9; & f(9) = 26; & f(10) = 23; \\
f(11) = 21; & f(12) = 30; & f(13) = 27; & f(14) = 13; & f(15) = 28; \\
f(16) = 16; & f(17) = 13; & f(18) = 23; & f(19) = 25; & f(20) = 11; \\
f(21) = 5; & f(22) = 19; & f(23) = 25; & f(24) = 30; & f(25) = 18; \\
f(26) = 9; & f(27) = 16; & f(28) = 15; & f(29) = 7; & f(30) = 30.
\end{array}$$

3.161. Compute the function associated to the rooted tree with edge set

$$\{(1,1),(2,12),(3,1),(4,3),(5,10),(6,17),(7,15),(8,7),(9,3),\\ (10,3),(11,12),(12,1),(13,4),(14,10),(15,1),(16,4),(17,4)\}$$

by the map ϕ^{-1} in the proof of 3.47.

3.162. Formulate a theorem for rooted trees similar to 3.75, and prove it by analyzing the bijection in 3.47.

3.163. Let G be the digraph in Figure 3.2. Use the algorithm in 3.52 to convert the walk

$$W = (1, b, 1, b, 1, a, 3, f, 5, m, 2, n, 5, h, 4, c, 3, f, 5, j, 4, g, 5, m, 2, k, 4)$$

to a path in G from 1 to 4.

3.164. What are the strong components of a functional digraph?

3.165. Show that a connected graph G with n vertices has n edges iff G has exactly one cycle.

3.166. Prove that a graph G is not connected iff there exists an ordering of the vertices of G for which the adjacency matrix of G is block-diagonal with at least two diagonal blocks.

3.167. Prove 3.60 using 3.58, and again without using 3.58.

3.168. How many connected simple graphs have vertex set $\{1,2,3,4\}$?

3.169. How many connected simple graphs on the vertex set $\{1,2,3,4,5\}$ have exactly five edges?

3.170. Bipartite Graphs. A graph G is called *bipartite* iff there exist two sets A and B (called *partite sets for* G) such that $A \cap B = \emptyset$, $A \cup B = V(G)$, and every edge of G has one endpoint in A and one endpoint in B. (a) Prove that a bipartite graph G has no cycle of odd length. (b) Prove that a graph G with no odd-length cycles is bipartite by considering, for each component C of G, the length of the shortest path from a fixed vertex $v_0 \in C$ to the other vertices in C (cf. 3.134). (c) Prove that a graph G with no odd-length cycles is bipartite by induction on the number of edges in G.

3.171. How many bipartite simple graphs have partite sets $A = \{1,2,\ldots,m\}$ and $B = \{m+1,\ldots,m+n\}$?

3.172. Suppose G is a k-regular graph with n vertices. (a) How many edges are in G? (b) If $k > 0$ and G is bipartite with partite sets A and B, prove that $|A| = |B|$.

3.173. Fix $k \geq 2$. Prove or disprove: there exists a k-regular bipartite graph G such that G has a cut-edge.

3.174. Prove that an n-vertex graph G in which every vertex has degree at least $(n-1)/2$ must be connected.

3.175. Let G be a forest with n vertices and k connected components. Compute $\sum_{v \in V(G)} \deg_G(v)$ in terms of n and k.

3.176. The *arboricity* of a simple graph G, denoted $\mathrm{arb}(G)$, is the least n such that there exist n forests F_i with $V(G) = \bigcup_{i=1}^{n} V(F_i)$ and $E(G) = \bigcup_{i=1}^{n} E(F_i)$. Prove that

$$\mathrm{arb}(G) \geq \max_{H} \left\lceil \frac{|E(H)|}{|V(H)|-1} \right\rceil,$$

where H ranges over all induced subgraphs of G with more than one vertex. (It can be shown that equality holds [99].)

3.177. Show that any tree not isomorphic to a path graph P_n (see 3.124(e)) must have at least three leaves.

3.178. Let T be a tree. Show that $\deg_T(v)$ is odd for all $v \in V(T)$ iff for all $e \in E(T)$, both connected components of $(V(T), E(T) \sim \{e\})$ have an odd number of vertices.

3.179. Helly Property of Trees. Suppose T, T_1, ..., T_k are trees, each T_i is a subgraph of T, and $V(T_i) \cap V(T_j) \neq \emptyset$ for all $i, j \leq k$. Show that $\bigcap_{i=1}^{k} V(T_i) \neq \emptyset$.

3.180. Let G be a tree with leaves $\{v_1, \ldots, v_m\}$. Let H be a tree with leaves $\{w_1, \ldots, w_m\}$. Suppose that, for each i and j, the length of the unique path in G from v_i to v_j equals the length of the unique path in H from w_i to w_j. Prove $G \cong H$.

3.181. For $1 \leq n \leq 7$, count the number of isomorphism classes of trees with n vertices.

3.182. (a) How many isomorphism classes of n-vertex trees have exactly 3 leaves? (b) How many trees with vertex set $\{1, 2, \ldots, n\}$ have exactly 3 leaves?

3.183. How many trees with vertex set $\{1, 2, \ldots, n\}$ have exactly k leaves?

3.184. Let K_n be the complete graph on n vertices (see 3.124). (a) Give a bijective or probabilistic proof that every edge of K_n appears in the same number of spanning trees of K_n. (b) Use Cayley's theorem to count the spanning trees of K_n that do not use the edge $\{1, 2\}$.

3.185. Use 3.75 to find the number of trees T with $V(T) = \{1, 2, \ldots, 8\}$ and $\deg(T) = [3, 3, 3, 1, 1, 1, 1, 1]$.

3.186. Let t_n be the number of trees on a given n-element vertex set. Without using Cayley's theorem, prove the recursion

$$t_n = \sum_{k=1}^{n-1} k \binom{n-2}{k-1} t_k t_{n-k}.$$

3.187. (a) Use the pruning bijection to find the word associated to the tree

$$T = (\{0, 1, \ldots, 8\}, \{\{1, 5\}, \{2, 8\}, \{3, 7\}, \{7, 0\}, \{6, 2\}, \{4, 7\}, \{5, 4\}, \{2, 4\}\}).$$

(b) Use the inverse of the pruning bijection to find the tree with vertex set $\{0, 1, \ldots, 8\}$ associated to the word 1355173.

3.188. Use the inverse of the pruning bijection to find all trees with vertex set $\{1, 2, \ldots, 7\}$ associated to the words in $\mathcal{R}(11334)$.

3.189. Let G be the graph with vertex set $\{\pm 1, \pm 2, \ldots, \pm n\}$ and with an edge between i and $-j$ for all $i, j \in \{1, 2, \ldots, n\}$. (a) Show that any spanning tree in G has at least one positive leaf and at least one negative leaf. (b) Develop an analogue of the pruning map that sets up a bijection between the set of spanning trees of G and pairs of words (u, v), where $u \in \{1, \ldots, n\}^{n-1}$ and $v \in \{-1, \ldots, -n\}^{n-1}$. Conclude that G has n^{2n-2} spanning trees.

3.190. (a) How many words in $\mathcal{R}(0^9 1^1 2^1 3^2 4^1)$ are terms? (b) How many words in $\mathcal{R}(0^8 1^1 2^3 3^1)$ are lists of n terms? What is n?

3.191. Given $w = 00220000201030$, use the proof of the cycle lemma 3.90 to find all i such that the cyclic rotation $R_i(w)$ is a list of 4 terms.

3.192. Consider a product $x_1 \times x_2 \times \cdots \times x_m$ where the binary operation $\times$ is not necessarily associative. Define a bijection from the set of complete parenthesizations of this product to the set of terms in $\mathcal{R}(0^m 2^{m-1})$. Then use 3.91 to show that the number of such parenthesizations is given by a Catalan number.

3.193. Let $\chi_n(x)$ be the chromatic polynomial for the graph C_n consisting of n vertices joined in a cycle. Prove that

$$\chi_n(x) = (x-1)^n + (-1)^n(x-1) \qquad (n \geq 2).$$

3.194. Find the chromatic polynomials for the graphs in 3.124(a),(b),(c),(d).

3.195. Find the chromatic polynomial and chromatic number for the graph G_2 in Figure 3.1.

3.196. Find two non-isomorphic simple graphs with the same chromatic polynomial.

3.197. A certain department wishes to schedule meetings for a number of committees, whose members are listed in the following table.

Committee	Members
Advisory	Driscoll, Loomis, Lasker
Alumni	Sheffield, Loomis
Colloquium	Johnston, Tchaikovsky, Zorn
Computer	Loomis, Clark, Spade
Graduate	Kennedy, Loomis, Trotter
Merit	Lee, Rotman, Fowler, Sheffield
Personnel	Lasker, Schreier, Tchaikovsky, Trotter
Undergraduate	Jensen, Lasker, Schreier, Trotter, Perkins

(a) What is the minimum number of time slots needed so that all committees could meet with no time conflicts? (b) How many non-conflicting schedules are possible if there are six (distinguishable) time slots available? (c) Repeat (a) and (b), assuming that Zorn becomes a member of the merit committee (and remains a member of the colloquium committee).

3.198. Let K_n be the complete graph on n vertices (see 3.124). (a) How many subgraphs does K_n have? (b) How many induced subgraphs does K_n have?

3.199. Prove that a graph G has at least one spanning tree iff G is connected.

3.200. Fill in the details of the proof of 3.111.

3.201. Use the spanning tree recursion 3.108 to find $\tau(G_1)$ for the graph G_1 in Figure 3.1.

3.202. Let T_1 and T_2 be spanning trees of a graph G.
(a) If $e_1 \in E(T_1) \sim E(T_2)$, prove there exists $e_2 \in E(T_2) \sim E(T_1)$ such that

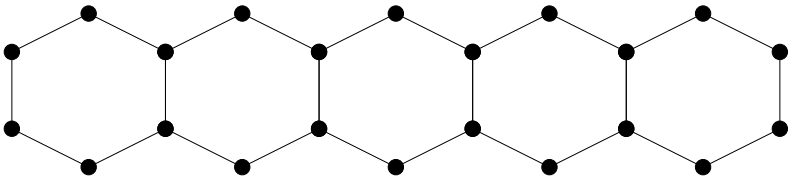
$$T_3 = (V(G), (E(T_1) \sim \{e_1\}) \cup \{e_2\})$$

is a spanning tree of G.
(b) If $e_1 \in E(T_1) \sim E(T_2)$, prove there exists $e_2 \in E(T_2) \sim E(T_1)$ such that

$$T_4 = (V(G), (E(T_2) \cup \{e_1\}) \sim \{e_2\})$$

is a spanning tree of G.

3.203. Fix $k \geq 3$. For each $n \geq 1$, let G_n be a graph obtained by gluing together n regular k-gons in a row along shared edges. The picture below illustrates the case $k = 6$, $n = 5$.



Let G_0 consist of a single edge. Prove the recursion

$$\tau(G_n) = k\tau(G_{n-1}) - \tau(G_{n-2}) \qquad (n \geq 2).$$

What are the initial conditions?

3.204. Given a simple graph G, let $G \sim v$ be the induced subgraph with vertex set $V(G) \sim \{v\}$. Assume $|V(G)| = n \geq 3$. (a) Prove that $|E(G)| = (n-2)^{-1} \sum_{v \in V(G)} |E(G \sim v)|$. (b) Prove that, for $v_0 \in V(G)$, $\deg_G(v_0) = (n-2)^{-1} \sum_{v \in V(G)} |E(G \sim v)| - |E(G \sim v_0)|$.

3.205. For each graph in 3.124(a) through (f), count the number of spanning trees by direct enumeration, and again by the matrix-tree theorem.

3.206. Confirm by direct enumeration that the digraph in Figure 3.25 has 16 spanning trees rooted at 0.

3.207. Let G be the graph with vertex set $\{0,1\}^3$ such that there is an edge between $v, w \in V(G)$ iff the words v and w differ in exactly one position. Find the number of spanning trees of G.

3.208. Let I be the $m \times m$ identity matrix, let J be the $m \times m$ matrix all of whose entries are 1, and let t, u be scalars. Show that $\det(tI - uJ) = t^m - mt^{m-1}u$.

3.209. Deduce Cayley's theorem 3.72 from the matrix-tree theorem 3.114.

3.210. Let A and B be disjoint sets of size m and n, respectively. Let G be the simple graph with vertex set $A \cup B$ and edge set $\{\{a, b\} : a \in A, b \in B\}$. Show that $\tau(G) = m^{n-1}n^{m-1}$.

3.211. How many closed Eulerian tours starting at vertex 5 does the digraph in Figure 3.26 have?

3.212. An *Eulerian tour* of a graph G is a walk in G that uses every edge exactly once and visits every vertex. (a) Find necessary and sufficient conditions for a graph to have a closed Eulerian tour. (b) Find necessary and sufficient conditions for a graph to have an Eulerian tour.

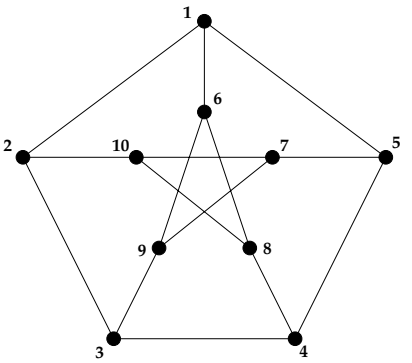
3.213. Consider a "digraph with indistinguishable edges" consisting of a vertex set V and a *multiset* of directed edges $(u, v) \in V \times V$. Formulate the notion of a closed Eulerian tour for such a digraph, and prove an analogue of 3.122.

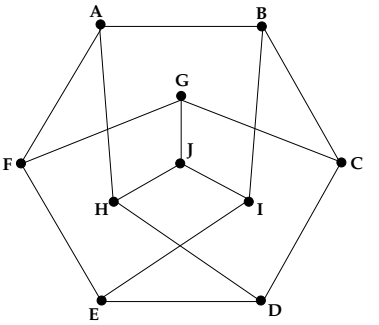
3.214. de Bruijn Sequences. Let $A = \{x_1, \ldots, x_n\}$ be an n-letter alphabet. For each $k \geq 2$, show that there exists a word $w = w_0 w_1 \cdots w_{n^k-1}$ such that the n^k words

$$w_i w_{i+1} \cdots w_{i+k-1} \qquad (0 \leq i < n^k)$$

(where subscripts are reduced mod n^k) consist of all possible k-letter words over A.

3.215. The *Petersen graph* is the graph G with vertex set consisting of all two-element subsets of $\{1, 2, 3, 4, 5\}$, and with edge set $\{\{A, B\} : A \cap B = \emptyset\}$. (a) Compute the number of vertices and edges in G. (b) Show that G is isomorphic to each of the graphs shown here.





(c) Show that G is 3-regular. (d) Is G bipartite? (e) Show that any two non-adjacent vertices in G have exactly one common neighbor.

3.216. Find (with proof) all k such that the Petersen graph has a cycle of length k.

3.217. Given any edge e in the Petersen graph G, count the number of cycles of length 5 in G that contain e. Use this to count the total number of cycles of length 5 in G.

3.218. (a) Prove that the Petersen graph G has exactly ten cycles of length 6. (b) How many claws (see 3.124) appear as induced subgraphs of G?

3.219. How many spanning trees does the Petersen graph have?

Notes

Our coverage of graph theory in this chapter has been limited to a few enumerative topics. Systematic expositions of graph theory may be found in [14, 17, 18, 27, 59, 67, 136, 143]; the text by West is especially recommended. Roberts [114] gives a treatment of graph theory that emphasizes applications.

The bijection used to enumerate rooted trees in 3.47 is due to Eğecioğlu and Remmel [32]. The original proof of Cayley's formula 3.75 appears in Cayley [24]. The pruning bijection described in §3.12 is due to Prüfer [105]; the image of a tree under this map is often called the *Prüfer code* of the tree. For more on the enumeration of trees, see Moon [96].

A version of the cycle lemma 3.90 occurs in the work of Dvoretsky and Motzkin [31]. This lemma and other equivalent results have been independently rediscovered (in various guises) by many authors. Our discussion of the enumeration of lists of terms in §3.14 closely follows Raney's classic paper on Lagrange inversion [107].

The matrix-tree theorem for undirected graphs is usually attributed to Kirchhoff [76]; Tutte extended the theorem to digraphs [132]. The enumeration of Eulerian tours in 3.122 was proved by van Aardenne-Ehrenfest and de Bruijn [133].