Chapter 11

Existence Problems in Graph Theory

Topics in graph theory underlie much of combinatorics. This chapter begins a sequence of three chapters dealing mostly with graphs and networks. The modern approach to graph theory is heavily influenced by computer science, and it emphasizes algorithms for solving problems. Among other things, we give an introduction to graph algorithms in these chapters. We also talk about applications of graph theory, many of which have already been mentioned in Section 3.1. This chapter emphasizes existence questions in graph theory and Chapter 13 emphasizes optimization questions. Chapter 12 is a transitional chapter, which begins with existence questions and ends with optimization questions. Of course, it is hard to make such a rigid partition. The same techniques and concepts of graph theory that are used on existence problems are usually useful for optimization problems, and vice versa.

In this chapter we examine four basic existence questions and discuss their applications. The existence questions are:

1. Is a given graph G connected; that is, does there exist, for every pair of vertices x and y of G, a chain between x and y?

2. Does a given graph G have a strongly connected orientation?

3. Does a given graph G have an eulerian chain, a chain that uses each edge exactly once?

4. Does a given graph G have a hamiltonian chain, a chain that uses each vertex exactly once?

We present theorems that help us to answer these questions. With practical use in mind, we shall be concerned with describing good procedures or algorithms for answering them. The results will have applications to such subjects as traffic flow, RNA codes, street sweeping, and telecommunications.

11.1 DEPTH-FIRST SEARCH: A TEST FOR CONNECTEDNESS

Suppose that $G = (V, E)$ is a graph. The first question we shall ask is this: Is G connected, that is, does there exist, for every pair of vertices x and y of G, a chain between x and y?

Given a small graph, it is easy to see from the corresponding diagram whether or not it is connected. However, for large graphs, drawing such diagrams is infeasible. Moreover, diagrams are not readily amenable to use as input in a computer. Instead, one has to input a graph into a computer by, for example, listing its edges. (See Section 3.7 for a more complete discussion of ways to input a graph into a computer.) In any case, it is now not so easy to check if a graph is connected. Thus, we would like an algorithm for testing connectedness. In Section 11.1.1 we present such an algorithm. The algorithm will play a crucial role in solving the traffic flow problem we study in Section 11.2 and can also be utilized in finding eulerian chains, the problem we discuss in Section 11.3. This algorithm is also important for solving optimization problems in graph theory, not just existence problems. However, for truly massive graphs, i.e., when the edge set does not fit into a computer's available RAM (Random Access Memory), totally new algorithms are needed. We discuss this point further in Section 11.1.4.

11.1.1 Depth-First Search

Suppose that $G = (V, E)$ is any graph. The method we describe for testing if G is connected is based on the *depth-first search procedure*, a highly efficient procedure which is the basis for a number of important computer algorithms. (See Aho, Hopcroft, and Ullman [1974, Ch. 5], Baase [1992], Cormen, Leiserson, and Rivest [1999], Golumbic [1980], Hopcroft and Tarjan [1973], Reingold, Nievergelt, and Deo [1977], or Tarjan [1972] for a discussion. See Exercise 9 for a related search procedure known as breadth-first search.)

In the depth-first search procedure, we start with a graph G with n vertices and aim to label the vertices with the integers $1, 2, \ldots, n$. We choose an arbitrary vertex and label it 1. Having just labeled a given vertex x with the integer k, we search among all neighbors of x and find an unlabeled neighbor, say y. We give vertex y the next label, $k+1$. We also keep track of the edges used in the labeling procedure by *marking* the edge $\{x, y\}$ traversed from x to y. We then continue the search among neighbors of y. In this way, we progress along chains of G leading away from x. The complication arises if we have just labeled a vertex z which has no unlabeled neighbors. We then go back to the neighbor u of z from which z was labeled—u is called the *parent* of z—and continue the search from u. We can keep track of the parent of a vertex since we have marked the edges traversed from a vertex to the next one labeled. The procedure continues until all the labels $1, 2, \ldots, n$ have been used (equivalently all vertices have been labeled) or it is impossible to continue because we have returned to a labeled vertex with no parent: namely, the vertex labeled 1.

www.ebook3000.com

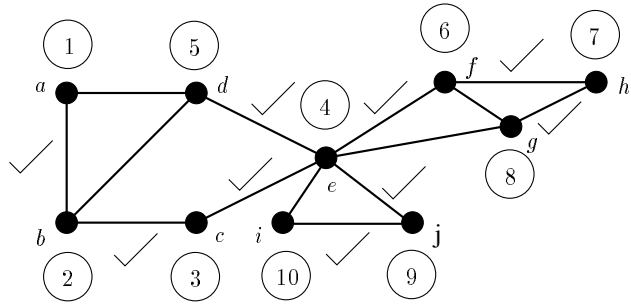


Figure 11.1: A labeling of vertices obtained using the depth-first search procedure. A check mark indicates marked edges.

We illustrate the labeling procedure in the graph of Figure 11.1. Vertex a is chosen first and labeled 1. The labels are shown in the figure by circled numbers next to the vertices. Vertex b is an unlabeled neighbor of vertex a which is chosen next and labeled 2. The edge $\{a, b\}$ is marked (we put a check mark in the figure). Next, an unlabeled neighbor of vertex b is found and labeled 3. This is vertex c. The edge $\{b, c\}$ is marked. An unlabeled neighbor of c, vertex e, is labeled 4, and the edge $\{c, e\}$ is marked. An unlabeled neighbor of e, vertex d, is labeled 5, and the edge $\{e, d\}$ is marked. Now after vertex d is labeled 5, all neighbors of d are labeled. Thus, we go back to e, the parent of d, and seek an unlabeled neighbor. We find one, vertex f, and label it 6. We also mark edge $\{e, f\}$. We now label vertex h with integer 7 and mark edge $\{f, h\}$ and label vertex g with integer 8 and mark edge $\{h, g\}$. Now all neighbors of g are labeled, so we go back to the parent of g, namely h. All neighbors of h are also labeled, so we go back to the parent of h, namely f. Finally, since all neighbors of f are labeled, we go back to the parent of f, namely e. From e, we next label vertex j with integer 9 and mark edge $\{e, j\}$. Then from j, we label vertex i with integer 10 and mark edge $\{j, i\}$. Now all vertices are labeled, so we stop.

It is easy to show that the labeling procedure can be completed if and only if the graph is connected. Thus, the depth-first search procedure provides an algorithm for testing if a graph is connected. Figure 11.2 shows the procedure on a disconnected graph. After having labeled vertex d with label 4, we find that the parent of d, namely c, has no unlabeled neighbors. Moreover, the same is true of the parent of c, namely a. Finally, a has no parent, so we cannot continue the labeling.

It should also be noted that if the labeling procedure is completed and T is the collection of marked edges, then there are $n - 1$ edges in T, for one edge is added to T each time we assign a new label. Moreover, suppose that H is the spanning subgraph of G whose edges are the edges in T. Then H has no circuits, because every edge added to T goes from a vertex labeled i to a vertex labeled j with $i < j$. Hence, H is a subgraph of G with $n = e + 1$ and no circuits. By Theorem 3.20, H is a tree. It is called the *depth-first search spanning tree*. (The reader should observe that the checked edges of Figure 11.1 form a tree.)

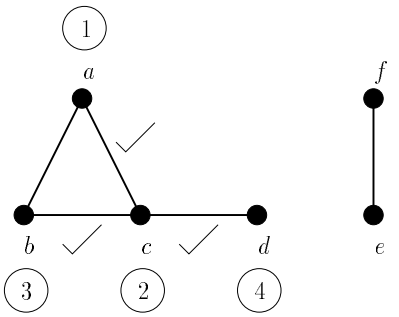


Figure 11.2: Depth-first search labeling and marking on a disconnected graph.

11.1.2 The Computational Complexity of Depth-First Search

The computational complexity of this depth-first search algorithm can be computed as follows. Suppose that the graph has n vertices and e edges. Each step of the procedure involves assigning a label or traversing an edge. The traversal is either forward from a labeled vertex to a neighbor or backward from a vertex to its parent. By marking not only edges that are *used* in a forward direction, but also edges that are *investigated* in a forward direction but lead to already labeled vertices, we can be sure that no edge is used or investigated in a forward direction more than once. Also, it is not hard to see that no edge can be used backward more than once. Thus, the entire procedure investigates at most $2e$ edges. Since at most n labels are assigned, the procedure terminates in at most $n + 2e$ steps. [In the notation of Section 2.18, this is an $O(n + e)$ algorithm.] Since $2e$ is at most

$$2\binom{n}{2} = 2\frac{n(n-1)}{2} = n^2 - n,$$

this is a polynomial bound on the complexity in terms of the number of vertices. [It is an $O(n^2)$ algorithm.[1]] The algorithm is very efficient.

It should be noted that a similar directed depth-first search procedure can be defined for digraphs. We simply label as before, but only go from a given vertex to vertices reachable from it by arcs, not edges. This method can be used to provide an efficient test for strong connectedness. For details, the reader is referred to Aho, Hopcroft, and Ullman [1974, Ch. 5], Cormen, Leiserson, and Rivest [1999], Hopcroft and Tarjan [1973], or Tarjan [1972]. (See also Exercise 8.)

11.1.3 A Formal Statement of the Algorithm[2]

For the reader who is familiar with recursive programming, we close this section with a more formal statement of the depth-first search algorithm. We first state a subroutine called DFSEARCH (v, u). In this subroutine, as well as in the main algorithm, k will represent the current value of the label being assigned and T will

[1] Some authors call the algorithm *linear* because its complexity is linear in $n + e$. Others call it *quadratic* because its complexity is quadratic in n.

[2] This subsection may be omitted.

www.ebook3000.com

be the set of marked edges. (We disregard the separate marking of edges that are investigated but lead to already labeled vertices.) The control is the vertex whose neighbors are currently being searched. Some vertices will already bear labels. The vertex v is the one that is just to be labeled and u is the parent of v.

Algorithm 11.1: DFSEARCH (v, u)

Input: A graph G with some vertices labeled and some edges in T, a vertex v that is just to be labeled, a vertex u that is the parent of v, and a current label k to be assigned.

Output: Some additional label assignments and a (possibly) new set T and a new current label k.

Step 1. Set the control equal to v, mark v labeled, assign v the label k, and replace k by $k + 1$.

Step 2. For each neighbor w of v, if w is unlabeled, add the edge $\{v, w\}$ to T, mark the edge $\{v, w\}$, call v the parent of w, and perform the algorithm DFSEARCH (w, v).

Step 3. If v has no more unlabeled neighbors and all vertices have been labeled, stop and output the labeling. If some vertex is still unlabeled, set the control equal to u and stop.

We can now summarize the entire depth-first search algorithm as follows.

Algorithm 11.2: Depth-First Search

Input: A graph G of n vertices.

Output: A labeling of the vertices of G using the integers $1, 2, \ldots, n$ and an assignment of $n - 1$ edges of G to a set T, or the message that the procedure cannot be completed.

Step 1. Set $T = \emptyset$ and $k = 1$ and let no vertex be labeled.

Step 2. Pick any vertex v, introduce a (dummy) vertex α, call α the parent of v, and perform DFSEARCH (v, α).

Step 3. If the control is ever set equal to α, output the message that the procedure cannot be completed.

Algorithm 11.2 terminates either because all vertices have been labeled or because the labeling procedure has returned to the vertex labeled 1, all neighbors of that vertex have labels, and there are some unlabeled vertices. In this case, it is easy to see that the graph is disconnected.

11.1.4 Testing for Connectedness of Truly Massive Graphs[3]

Graphs arising from modern applications involving telecommunications traffic and web data may be so massive that their edge sets do not fit into main memory

[3]This subsection may be omitted.

Table 11.1: Adjacency Structure for a Graph

Vertex x	a	b	c	d	e	f	g
Vertices adjacent to x	e, d	c, g	b, g	a, e	a, d, f	e	b, c

on modern computers. For example, massive telephone calling graphs arise when telephone numbers are vertices and a call between telephone numbers is an edge. (More generally, we view the calls as directed and we look at multidigraphs.) Such graphs can have several billion edges. When the edge set does not fit into RAM (Random Access Memory), many of the classical graph theory algorithms break down. (Even a computer with 6 gigabytes of main memory cannot hold the full telephone calling graphs that arise in practice.) This calls for the development of "external memory algorithms." Examples of such algorithms for connectedness are those in Abello, Buchsbaum, and Westbrook [2002]. Other work on massive telephone-calling graphs and related graphs of connectivity over the Internet emphasizes the evolution of such graphs over time and the changing connectivity of the graphs. For work on this theme, see, for example, Aiello, Chung, and Lu [2000]. For information on external memory algorithms, see Abello and Vitter [1999]. In particular, see the paper by Abello, Pardalos, and Resende [1999].

EXERCISES FOR SECTION 11.1

1. For each graph of Figure 11.3, perform a depth-first search beginning with the vertex labeled a. Indicate all vertex labels and all marked edges.
2. Given the adjacency structure of Table 11.1 for a graph G of vertices a, b, c, d, e, f, g, use depth-first search to determine if G is connected.
3. From each of the following adjacency matrices for a graph G, use depth-first search to determine if G is connected.

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

4. For each graph of Figure 11.3, find a depth-first search spanning tree.
5. Is every spanning tree attainable as a depth-first search spanning tree? Why?
6. Suppose that the depth-first search algorithm is modified so that if further labeling is impossible, but there is still an unlabeled vertex, some unlabeled vertex is chosen, given a new label, and the process starts again. What can you say about the subgraph determined by the marked edges (the edges in T)?

www.ebook3000.com

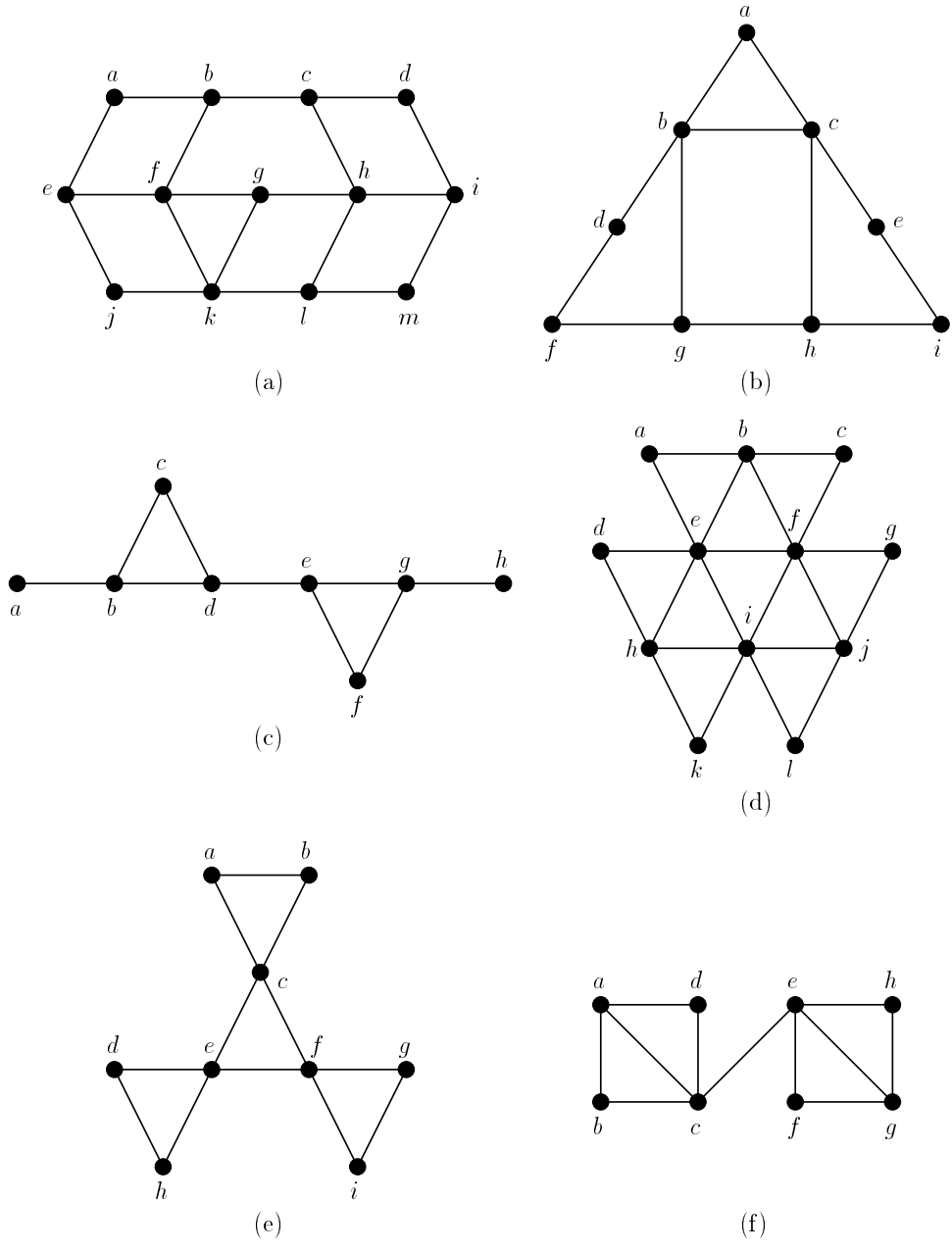


Figure 11.3: Graphs for exercises of Section 11.1.

7. How can the depth-first search algorithm as described in the text be modified to count the number of connected components of a graph G?

8. (a) Is the following test for strong connectedness of a digraph D correct? Pick any vertex of the digraph to receive label 1 and perform a directed depth-first search until no more vertices can be labeled. Then D is strongly connected if and only if all vertices have received a label.

   (b) If this is not a correct test, how can it be modified?

9. In the algorithm known as *breadth-first search*, we start with an arbitrarily chosen vertex x and place it at the head of an ordered list or *queue*. At each stage of the algorithm, we pick the vertex y at the head of the queue, delete y from the queue, and add to the tail of the queue, one at a time, all vertices adjacent to y which have never been on the queue. We continue until the queue is empty. If not all vertices have been on the queue, we start with another arbitrarily chosen vertex which has never been on the queue. Breadth-first search fans out in all directions from a starting vertex, whereas depth-first search follows one chain at a time from this starting vertex to an end. We have already employed breadth-first search in Algorithm 3.1 of Section 3.3.4. For a detailed description of breadth-first search, see, for example, Baase [1992], Even [1979], or Golumbic [1980]. Perform a breadth-first search on each graph of Figure 11.3, beginning with the vertex a. Indicate the order of vertices encountered by labeling each with an integer from 1 to n in the order they reach the head of the queue. (*Note*: In many applications, breadth-first search is inefficient or unwieldy compared to depth-first search. This is true, for example, in algorithms for testing if a given graph is planar. However, in many network optimization problems, such as most of those studied in Chapter 13, breadth-first search is the underlying procedure.)

10. Can breadth-first search (Exercise 9) be used to test a graph for connectedness? If so, how?

11. What is the computational complexity of breadth-first search?

12. Consider a telephone calling graph over the span of six weeks for the phone numbers in a given area code.

    (a) What is the size of the vertex set for this graph assuming that all possible phone numbers are available?

    (b) What is the size of the edge set if each phone number initiates an average of 12 calls per day?

11.2 THE ONE-WAY STREET PROBLEM

11.2.1 Robbins' Theorem

In this section we discuss an application of graph theory to a traffic flow problem. Imagine that a city has a number of locations, some of which are joined by two-way streets. The number of cars on the road has markedly increased, resulting in traffic jams and increased air pollution, and it has been suggested that the city should make all its streets one way. This would, presumably, cut down on traffic congestion. The

www.ebook3000.com

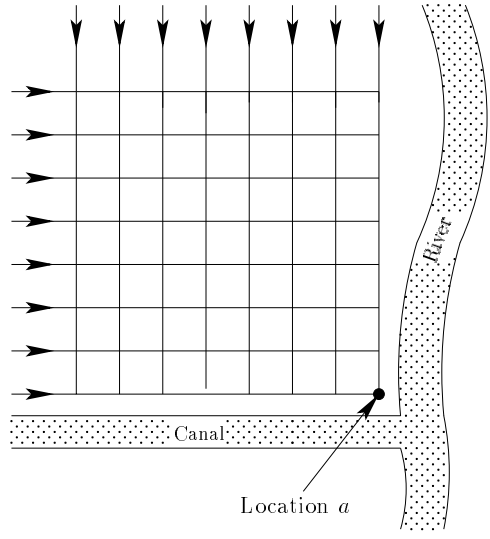


Figure 11.4: A one-way street assignment for the streets of a city which leaves travelers stuck at location a.

question is: Can this always be done? If not, when? The answer is: Of course, it can always be done. Just put a one-way sign on each street! However, it is quite possible that we will get into trouble if we make the assignment arbitrarily, for example by ending up with some places that we can get into and never leave. (See, for example, Figure 11.4, which shows an assignment of directions to the streets of a city that is satisfactory only for someone who happens to own a parking lot at location a.) We would like to make every street one-way in such a manner that for every pair of locations x and y, it is possible (legally) to reach x from y and reach y from x. Can this always be done?

To solve this problem, we assume for simplicity that all streets are two-way at the beginning. See Boesch and Tindell [1980] or Roberts [1978] for a treatment of the problem where some streets are one-way before we start (see also Exercise 13). We represent the transportation network of a city by a graph G. Let the locations in question be the vertices of G and draw an edge between two locations x and y if and only if x and y are joined by a two-way street. A simple example of such a graph is shown in Figure 11.5(a). In terms of the graph, our problem can now be restated as follows: Is it possible to put a direction or arrow (a one-way sign) on each edge of the graph G in such a way that by following arrows in the resulting figure, which is a digraph, one can always get from any point x to any other point y? If it is not always possible, when is it possible? Formally, we define an *orientation* of a graph G as an assignment of a direction to each edge of G. We seek an orientation of G which, to use the terminology of Section 3.2, is strongly connected.

In the graph of Figure 11.5(a), we can certainly assign a direction to each edge to obtain a strongly connected digraph. We have done this in Figure 11.5(b). However,

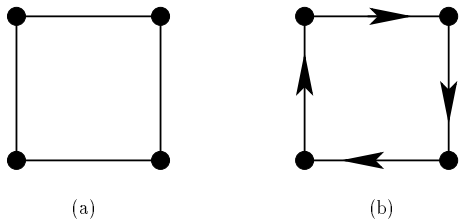


Figure 11.5: (a) A two-way street graph for a city and (b) a strongly connected orientation.

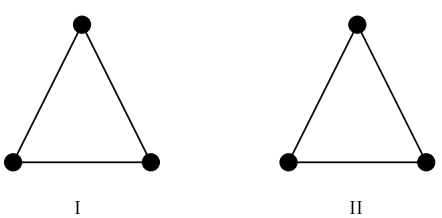


Figure 11.6: Graph representing a disconnected city.

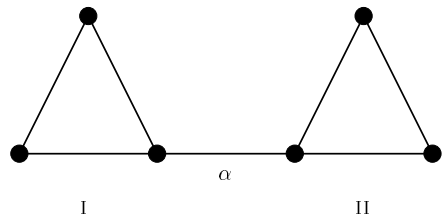


Figure 11.7: α is a bridge.

it is not always possible to obtain a strongly connected orientation. For example, if our graph has two components, as in the graph of Figure 11.6, there is no way of assigning directions to edges which will make it possible to go from vertices in section I of the graph to vertices in section II. To have a strongly connected orientation, our graph must certainly be connected. However, even a connected graph may not have a strongly connected orientation. Consider the graph of Figure 11.7. It is connected. We must put a direction on edge α. If α is directed from section I to section II, then from no vertex in II can we reach any vertex in I. If α is directed from II to I, then from no vertex in I can we reach any vertex in II. What is so special about the edge α? The answer is that it is the only edge joining two separate pieces of the graph. Put more formally, removal of α (but not the two vertices it joins) results in a disconnected graph. Such an edge in a connected graph is called a *bridge*. Figure 11.8 gives another example of a bridge α. It is clear that for a graph G to have a strongly connected orientation, G must not only be connected, but it must also have no bridges.

In Figure 11.8, suppose that we add another bridge β joining the two separate components which are joined by the bridge α, obtaining the graph of Figure 11.9. Does this graph have a strongly connected orientation? The answer is yes. A satisfactory assignment is shown in Figure 11.10. Doesn't this violate the observation we have just made, namely that if a graph G has a strongly connected orientation, it can have no bridges? The answer here is no. For we were too quick to call β a bridge. In the sense of graph theory, neither β nor α is a bridge in the graph of Figure 11.9. A graph-theoretical bridge has the nasty habit that if we have a bridge

www.ebook3000.com

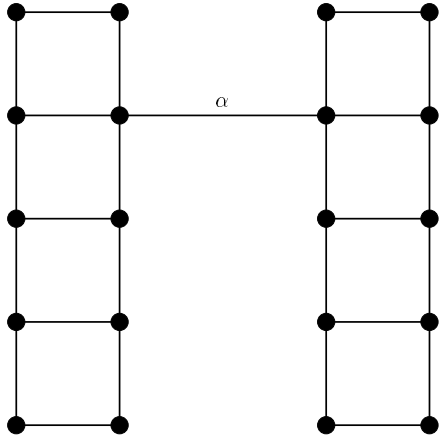


Figure 11.8: α is again a bridge.

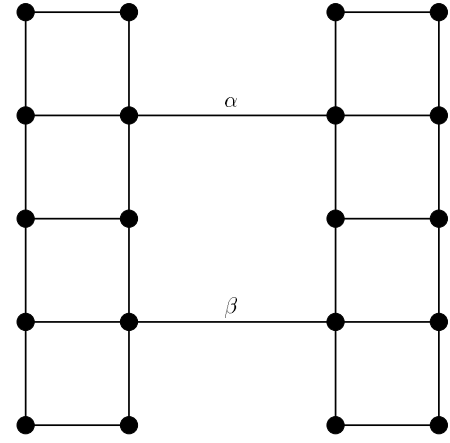


Figure 11.9: Edge α is no longer a bridge.

and build another bridge, then neither is a bridge! In applying combinatorics, if we use suggestive terminology such as the term *bridge*, we have to be careful to be consistent in our usage.

Suppose now that G has the following properties: It is a connected graph and has no bridges. Are these properties sufficient to guarantee that G has a strongly connected orientation? The answer turns out to be yes, as we summarize in the following theorem.

Theorem 11.1 (Robbins [1939].) A graph G has a strongly connected orientation if and only if G is connected and has no bridges.

We omit the proof of Theorem 11.1. For two different proofs, see Roberts [1976, 1978] and Boesch and Tindell [1980]. For a sketch of one of the proofs, see Exercise 13.

11.2.2 A Depth-First Search Algorithm

Robbins' Theorem in some sense completely solves the problem we have stated. However, it is not, by itself, a very useful result. For the theorem states that there is a one-way street assignment for a connected, bridgeless graph, but it does not tell us how to find such an assignment. In this section we present an efficient algorithm for finding such an assignment if one exists.

Suppose that $G = (V, E)$ is any graph. We shall describe a procedure for orienting G which can be completed if and only if G is connected. If G has no bridges, the resulting orientation is strongly connected. For a proof of the second assertion, see Roberts [1976]. The procedure begins by using the depth-first search procedure described in Section 11.1.1 to label the vertices and mark the edges. If the labeling procedure cannot be completed, G is disconnected and hence has no

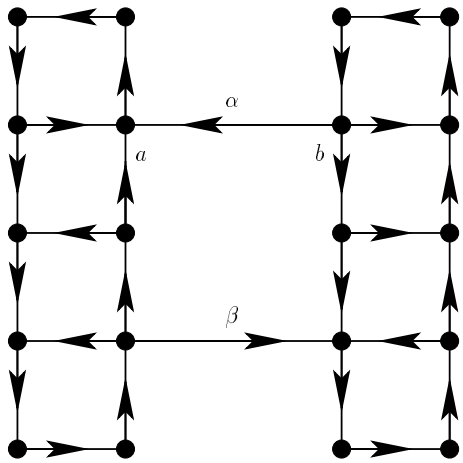


Figure 11.10: A satisfactory one-way assignment for the graph of Figure 11.9.

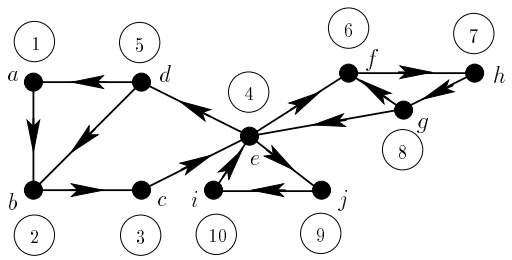


Figure 11.11: The strongly connected orientation for the graph G of Figure 11.1 obtained from the labeling and marking in that figure.

strongly connected orientation. The next step is to use the labeling and the set of marked edges T to define an orientation. Suppose that an edge is labeled with the labels i and j. If the edge is in T, that is, if it is marked, we orient from lower number to higher number, that is, from i to j if i is less than j and from j to i otherwise. If the edge is not in T, that is, if it is unmarked, we orient from higher number to lower number. We illustrate this procedure using the labeling and marking of Figure 11.1. The corresponding orientation is shown in Figure 11.11. Note that we orient from a to b because edge $\{a, b\}$ is in T and a gets the label 1, which is less than the label 2 given to b. However, we orient from d to b because edge $\{d, b\}$ is not in T and the label of d, that is, 5, is more than the label of b, that is, 2.

This algorithm always leads to an orientation of G if the labeling procedure can be completed, that is, if G is connected. However, it is guaranteed to lead to a strongly connected orientation only if G has no bridges. If we assume that the decision of how to orient an edge, given a labeling and a record of edges in T, takes about the same amount of time as each step in the depth-first search procedure which is used to label and record edges in T, then the computational complexity

www.ebook3000.com

$g(n,e)$ of this algorithm for an n vertex, e edge graph is at most the computational complexity $f(n,e)$ of the depth-first search procedure plus the number of edges in the graph G. In Section 11.1.2 we showed that $f(n,e)$ is at most $n+2e$. Thus, $g(n,e)$ is at most $n+3e$. [It is again an $O(n+e)$ algorithm.] In terms of n, the complexity is at most $\frac{3}{2}n^2 - \frac{3}{2}n + n$, since $e \leq \binom{n}{2}$. Thus, we have a polynomially bounded algorithm. [It is $O(n^2)$.] For other algorithms, see Boesch and Tindell [1980] and Chung, Garey, and Tarjan [1985]. For information on the use of parallel algorithms for strongly connected orientations, see Atallah [1984] and Vishkin [1985] and see a summary in Karp and Ramachandran [1990].

11.2.3 Efficient One-Way Street Assignments

Not every one-way street assignment for a city's streets is very efficient. Consider, for example, the one-way street assignment shown in the graph of Figure 11.10. This one-way street assignment is obtainable by the algorithm we have described (Exercise 4). If a person at location a wanted to reach location b, he or she used to be able to do it very quickly, but now has to travel a long way around. In short, this assignment meets the criterion we have set up, of being able to go from any point to any other point, but it does not give a very efficient solution to the traffic flow problem. In general, the most efficient one-way street assignment is one in which, "on the whole," distances traveled are not too great.

There are many possible notions of efficiency. We shall make some precise. These are all based on notions of distance. If G is a graph, recall that the distance $d(u,v)$ between vertices u and v is the length (number of edges) of the shortest chain between u and v in G. If D is a digraph, the *distance* $\bar{d}(u,v)$ from u to v is the length of the shortest path from u to v in D. One way to define efficiency, the only way we shall explore, is to say that an orientation is most efficient if it minimizes some "objective function" defined in terms of distances. Examples of such objective functions are given in Table 11.2. The problems of minimizing some of these objective functions are equivalent. For instance, minimizing (2) is equivalent to minimizing (3).

It turns out that we are in a "bad news-worse news" situation. The bad news is that no good algorithms are known for finding the most efficient strongly connected orientation according to any of the criteria in Table 11.2. The worse news is that some of these optimization problems are really hard. For example, Chvátal and Thomassen [1978] show that finding a strongly connected orientation that minimizes (1) is an NP-hard problem, to use the terminology of Section 2.18.

It should be remarked that efficiency is not always the goal of a one-way street assignment. In the National Park System throughout the United States, traffic congestion has become a serious problem. A solution being implemented by the U.S. National Park Service is to try to discourage people from driving during their visits to national parks. This can be done by designing very inefficient one-way street assignments, which make it hard to get from one place to another by car, and by encouraging people to use alternatives, for example bicycles or buses. Figure 11.12

Table 11.2: Some Objective Functions for Strongly Connected Orientations

Objective Function	Description
(1) $\max_{u,v} \bar{d}(u,v)$	Maximum distance traveled
(2) $\sum_{u,v} \bar{d}(u,v)$	Sum of distances traveled
(3) $\frac{1}{n^2-n} \sum_{u,v} \bar{d}(u,v)$	Average distance traveled
(4) $\max_{u,v} \left[\bar{d}(u,v) - d(u,v)\right]$	Maximum change in distance
(5) $\sum_{u,v} \left[\bar{d}(u,v) - d(u,v)\right]$	Sum of changes in distance
(6) $\frac{1}{n^2-n} \sum_{u,v} \left[\bar{d}(u,v) - d(u,v)\right]$	Average change in distance
(7) $\sum_{u} \max_{x} \bar{d}(u,x)$	Sum of maximum distance to travel
(8) $\sum_{u} \left[\max_{x} \bar{d}(u,x) - \max_{x} d(u,x)\right]$	Sum of changes in maximum distance to travel

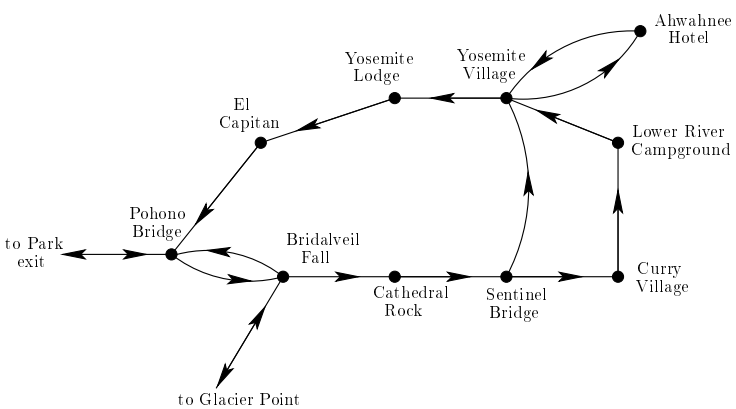


Figure 11.12: One-way street assignment for Yosemite Valley automobile traffic, summer of 1973. (*Note*: Buses may go both ways between Yosemite Lodge and Yosemite Village and between Yosemite Village and Sentinel Bridge.)

www.ebook3000.com

shows approximately the one-way street assignment instituted in the Yosemite Valley section of Yosemite National Park during the summer of 1973. (In addition to making roads one-way, the Park Service closed off others to cars entirely. Note that there is a two-way street; the idea of making *every* street one-way is obviously too simple. Also, in this situation, there is no satisfactory assignment in which every street is one-way. Why?) This is a highly inefficient system. For example, to get from Yosemite Lodge to Yosemite Village, a distance of less than 1 mile by road, it is necessary to drive all the way around via Pohono Bridge, a distance of over 8 miles. However, the Park's buses are allowed to go directly from Yosemite Lodge to Yosemite Village and many people are riding them!

From a mathematical point of view, it would be nice to find algorithms for finding an inefficient (the most inefficient) one-way street assignment, just as to find an efficient (the most efficient) such assignment. Unfortunately, no algorithms for inefficient assignments are known either.

11.2.4 Efficient One-Way Street Assignments for Grids

If a problem proves to be hard, one can explore simpler versions of the problem, solve it in special cases, and so on. This suggests the question: Are there good solutions for the problem of finding the most efficient strongly connected orientation if the graph is known to have a special structure? The structures of interest should, of course, have something to do with the application of interest. The most natural structures to explore are the grid graphs which correspond to cities with north-south streets and east-west avenues. We explore these in what follows. To fix some notation, we let G_{n_1,n_2} be the graph consisting of $n_1 + 1$ east-west avenues and $n_2 + 1$ north-south streets. Thus, for example, $G_{3,5}$ is the 4×6 grid. (The slightly unwieldy notation is borrowed from the literature of the subject, where it is useful in some of the proofs.)

One can always find an optimal strongly connected orientation by brute force: Try all orientations. However, brute force algorithms can be very impractical. Consider the brute force algorithm of trying all orientations of the grid graph $G_{n-1,n-1}$ consisting of n east-west avenues and n north-south streets. Each such orientation would be tested for strong connectedness, and if it is strongly connected, the objective function being minimized would be calculated. How many orientations of this grid graph are there?

To count the number of orientations of our grid graph $G_{n-1,n-1}$, one observes that there are $n-1$ edges in each east-west avenue and $n-1$ edges in each north-south street. Thus, the number of edges e is given by

$$e = 2n(n-1) = 2n^2 - 2n$$

(see Exercise 15). Having computed the number of edges, we note that the number of orientations of $G_{n-1,n-1}$ is given by the product rule. Since each edge has two possible orientations, the total number of orientations is

$$\overbrace{2 \times 2 \times \cdots \times 2}^{e \text{ times}} = 2^e = 2^{2n^2-2n}.$$

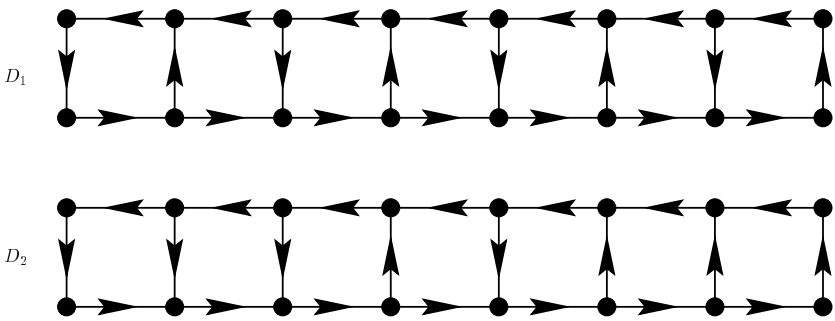


Figure 11.13: The alternating orientation D_1 and an optimal solution D_2 to the problem of minimizing measures (1) and (8) for the grid $G_{1,7}$.

Even if we could check 1 billion orientations per second, then when n is only 7, a simple calculation shows that it would take more than 500,000,000 years to check all of these orientations. The brute force method of trying all orientations is clearly impractical. This conclusion does not change even with an order-of-magnitude change in the speed of computation. (As an aside remark, we conclude that one should always count the number of steps an algorithm will take before implementing it.)

For each of the measures or objective functions defined in Table 11.2, it is a very concrete question to find the optimal orientation of the grid G_{n_1,n_2} in the sense of minimizing that measure. It is surprising that it took such a long time for it to be answered, and then only for some of the measures. The first efforts to find optimal strongly connected orientations for $G_{1,n}$, the grid of 2 east-west avenues and $n+1$ north-south streets, were carried out in 1973. Yet it was not until quite a bit later that a series of papers by Roberts and Xu [1988, 1989, 1992, 1994] and Han [1989] provided some solutions to these problems, at least for measures (1) and (8) of Table 11.2.

It is useful to describe here the optimal solutions for the case of $G_{1,n}$ with n odd. One can show that in this case, any solution that minimizes either measure (1) or (8) will have a cycle around the outside of the grid (Exercise 18). Thus, what is left is to decide on the orientations of interior north-south streets. One's intuition is to alternate these interior streets. It turns out that this alternating orientation is not optimal. Rather, it is optimal to orient the two middle streets opposite to their closest (east or west) boundaries, and to orient all other interior streets to agree with their closest boundaries (see Figure 11.13). Roberts and Xu [1989] prove that if $n \geq 5$, this orientation minimizes measures (1) and (8) among all strongly connected orientations of $G_{1,n}$, n odd. They also prove that in the case of measure (8), this orientation is the unique optimal orientation, up to the choice of direction to orient the cycle around the boundary.

It is of interest to compare the optimal solution to the alternating orientation, i.e., the orientation that alternates north-south streets. For instance, when measure (1) is used and $n = 5$, the optimal solution gets a score (measure) of 7 and the

www.ebook3000.com

alternating orientation a score of 9. When $n = 7$, the scores go up to 9 and 11, respectively. When measure (8) is used, the difference is much greater. When $n = 5$, the optimal solution gets a score of 10 and the alternating orientation a score of 26; when $n = 7$, the scores are 12 and 32, respectively. (These scores are all easy to verify.)

Similar results are described in Roberts and Xu [1988, 1989, 1992, 1994] and Han [1989] for measures (1) and (8) for other grids. Some of this work is described in the exercises. However, finding optimal solutions for grids under other measures remain open problems. We are quickly led to the frontiers of mathematical research.

It is interesting to study the natural "New York City" solution which alternates the orientation of east-west avenues and of north-south streets. Although not optimal for measures (1) and (8), the New York City solution is better for some grids under some other measures than solutions that are optimal under measures (1) and (8). For example, this is true for the grid $G_{5,7}$ under measure (3). Thus, there is much more research to be done here.

It is also interesting to study optimal strongly connected orientations for other graphs. We consider annular cities, cities with circular beltways and spokes heading out from the city center, in the next section.

There are layers of complication that our simplified mathematical models have omitted. For instance, our models do not consider volume of traffic on different streets or different distances or times needed to traverse different streets. To describe another complication, some of the optimal solutions described by Roberts and Xu [1994] contain arcs (x, y) and (z, y). Imagine the problems at an intersection like y. If we add penalties for such intersections, we get a different optimization problem.

11.2.5 Annular Cities and Communications in Interconnection Networks

The problem of finding the most efficient one-way street assignment has also been studied for annular cities, cities with circular beltways and spokes heading out from the city center. In particular, we shall discuss the graph $AN(c, s)$, where there are c concentric circles around a center and s straight-line spokes crossing all circles. Figure 11.14 shows $AN(3, 8)$. We describe some results about $AN(c, s)$ in the exercises (Exercises 23 and 24). Here, we mention one result. Bermond, *et al.* [2000] study orientations that minimize measure (8) of Table 11.2 for annular cities. In particular, they show that if $s = 4k$ for some k and $c > k + 2$, there is a strongly connected orientation for which $\bar{D} = \max_{u,v} \bar{d}(u, v) = \max_{u,v} d(u, v) = D$, i.e., the maximum distance traveled is no more than the maximum distance traveled before the orientation.

The problem of identifying those graphs that have a strongly connected orientation for which $\bar{D} = D$ or $\bar{D}$ is close to D also arises in communications in interconnection networks. In "total exchange," initially each processor in such a network has an item of information that must be distributed to every other processor. This arises in a variety of parallel processing applications. The process is accomplished by a sequence of synchronous calls between processors. During each

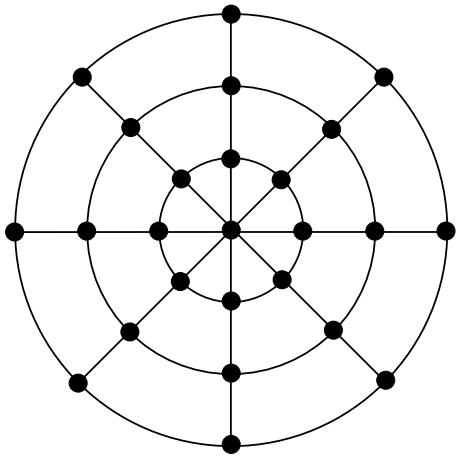


Figure 11.14: $AN(3,8)$

call, a processor can communicate with all its neighbors. In *full-duplex* Δ*-port*, communication can go both ways between neighbors. Thus, the minimum number of steps needed for all processors to get all the information is $\max_{u,v} d(u,v)$. In *half-duplex* Δ*-port*, simultaneous exchange on a given link is not authorized. Then, a given protocol corresponds to an orientation of edges. At each step, all the information known by a processor goes to all the processors to which there is an arrow in the orientation. Thus, $\bar{D}$ is an upper bound on the minimum number of steps needed for all the processors to get all the information. Since D is a lower bound, any network G for which there is a strongly connected orientation and which has $\bar{D}$ close to D has a protocol that has close to optimal performance.

The measure (1) or $\bar{D}$ has been studied for other graphs as well. For instance, Gutin [1994], Koh and Tan [1996a,b], and Plesnik [1986] found optimal orientations for the so-called complete multipartite graphs and McCanna [1988] for the n-dimensional cube, an important architecture in computer networks.

EXERCISES FOR SECTION 11.2

1. In each graph of Figure 11.3, find all bridges.
2. Apply the procedure described in the text for finding one-way street assignments to the graphs of Figure 11.3. In each case, check if the assignment obtained is strongly connected.
3. Repeat Exercise 2 for the graphs of Figure 3.23.
4. Show that the one-way street assignment shown in Figure 11.10 is attainable using the algorithm described in the text.
5. Find conditions for a graph to have a weakly connected orientation (Exercise 10, Section 3.2).

www.ebook3000.com

6. If a graph has a weakly connected orientation, how many weakly connected orientations does it have?
7. Give an example of a graph that has a weakly connected orientation but not a unilaterally connected orientation (Exercise 9, Section 3.2).
8. Let G be a connected graph. A *cut vertex* of G is a vertex with the following property: When you remove it and all edges to which it belongs, the result is a disconnected graph. In each graph of Figure 11.3, find all cut vertices.
9. Can a graph with a cut vertex (Exercise 8) have a strongly connected orientation?
10. Prove or disprove: A connected graph with no cut vertices (Exercise 8) has a strongly connected orientation.
11. Prove that an edge $\{u, v\}$ in a connected graph G is a bridge if and only if every chain from u to v in G includes edge $\{u, v\}$.
12. If u and v are two vertices in a digraph D, let $d_D(u, v) = \bar{d}(u, v)$ be the distance from u to v, i.e., the length of the shortest path from u to v in D. Distance is undefined if there is no path from u to v. Similarly, if u and v are two vertices in a graph G, the distance $d_G(u, v)$ from u to v is the length of the shortest chain from u to v in G. Again, distance is undefined if there is no chain.

    (a) Show that $d_D(u, v)$ may be different from $d_D(v, u)$.

    (b) Show that $d_G(u, v)$ always equals $d_G(v, u)$.

    (c) Show that if v is reachable from u and w from v, then $d_D(u, w) \leq d_D(u, v) + d_D(v, w)$.

13. In this exercise we consider the case of cities in which some streets are one-way to begin with. Our approach is based on Boesch and Tindell [1980]. A *mixed graph* consists of a set of vertices, some of which are joined by one-way arcs and some of which are joined by undirected edges. (The arcs correspond to one-way streets, the edges to two-way streets.) A mixed graph G can be translated into a digraph $D(G)$ by letting each edge be replaced by two arcs, one in each direction. G will be called *strongly connected* if $D(G)$ is strongly connected, and *connected* if $D(G)$ is weakly connected. An edge α in a connected mixed graph is called a *bridge* if removal of α but not its end vertices results in a mixed graph that is not connected.

    (a) Suppose that G is a strongly connected mixed graph and that $\{u, v\}$ is an edge of G. Let D' be the digraph obtained from $D(G)$ by omitting arcs (u, v) and (v, u) but not vertices u and v. Let A be the set of all vertices reachable from u by a path in D', less the vertex u. Let B be defined similarly from v. Suppose that u is not in B and v is not in A. Prove that the edge $\{u, v\}$ must be a bridge of G.

    (b) Use the result of part (a) to prove the following theorem of Boesch and Tindell [1980]: If G is a strongly connected mixed graph and $\{u, v\}$ is an edge of G that is not a bridge, there is an orientation of $\{u, v\}$ so that the resulting mixed graph is still strongly connected.

    (c) Prove from part (b) that every connected graph without bridges has a strongly connected orientation.

    (d) Translate your proof of part (c) into an algorithm for finding a strongly connected orientation of a connected, bridgeless graph.

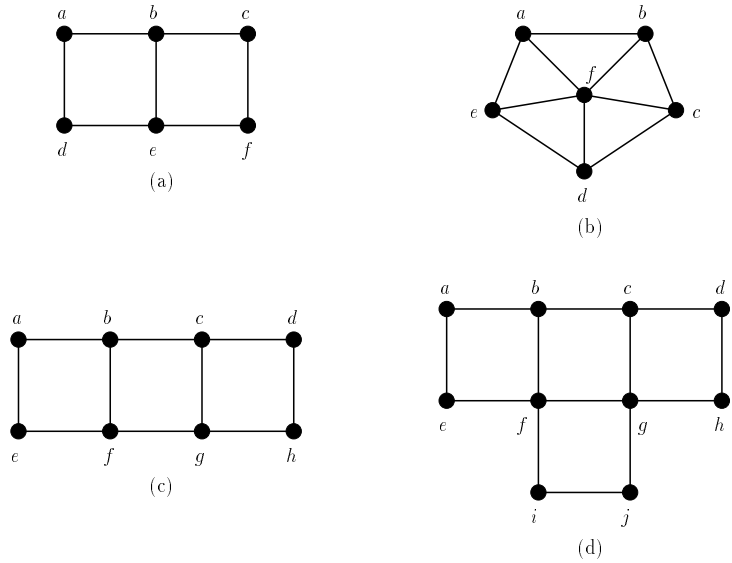


Figure 11.15: Graphs for exercises of Section 11.2.

(e) Illustrate your algorithm in part (d) on the graphs of Figure 11.3.

(f) What is the computational complexity of your algorithm in part (d)? How does this compare to the algorithm described in the text?

14. Find the number of orientations (not necessarily strongly connected) of

(a) $G_{2,n}$ (b) $G_{5,11}$ (c) $AN(4,6)$

15. Use Theorem 3.1 to calculate the number of edges in the grid graph $G_{n-1,n-1}$.

16. For each of measures (1)–(8) of Table 11.2, find a most efficient strongly connected orientation for each graph of Figure 11.15.

17. Calculate the measures (1)–(8) of Table 11.2 for the orientations of Figure 11.16. These are, respectively, the alternating orientation and optimal solution analogues to D_1 and D_2 of Figure 11.13 for $G_{1,7}$.

18. Show that in the case of $G_{1,n}$, n odd, any solution that minimizes either measure (1) or (8) of Table 11.2 will have a cycle around the outside of the grid.

19. When n is even, let orientation O_1 of $G_{1,n}$ be obtained as follows. Orient the outside of the grid with a counterclockwise cycle. Let $k+1$ be $(n/2)+1$. Orient the $(k-1)$st north-south street up, the $(k+2)$nd north-south street down, and all others to agree with the 1st or $(n+1)$st north-south street, whichever is closer. Roberts and Xu [1989] show that O_1 is optimal for measures (1) and (8) of Table 11.2 for $n \geq 10$.

(a) Draw O_1 for $G_{1,10}$.

(b) Calculate measures (1) and (8) of Table 11.2 for O_1.

www.ebook3000.com

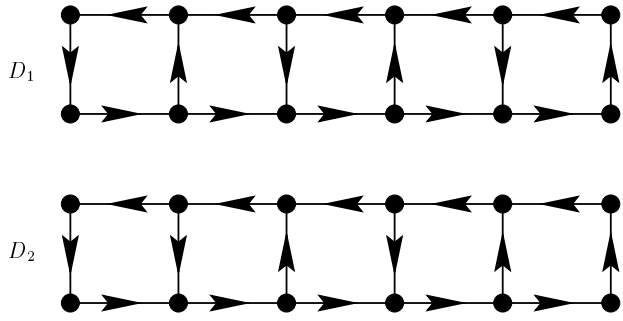


Figure 11.16: Orientations for grid $G_{1,5}$.

(c) Why is the orientation with alternating north-south streets, together with the outside counterclockwise cycle, not an optimal solution for these measures for $G_{1,n}$, n even?

20. When n is odd, let orientation O_2 of $G_{2,n}$ be obtained as follows. Orient the top and bottom east-west avenues from right to left and the middle east-west avenue from left to right. Orient the north-south streets joining the top and middle east-west avenues as follows: Let $k = (n/2) - 1$. Street 1 is down, street $n+1$ is up, the "middle streets" k and $k+1$ disagree with (have opposite orientations of) the closer of streets 1 and $n+1$, while all others have the same orientation as the closer of streets 1 and $n+1$. Orient the north-south streets joining the middle and bottom east-west avenues as follows: Street 1 is up, street $n+1$ is down, streets $k-1$ and $k+1$ disagree with the closer of streets 1 and $n+1$, while all others have the same orientation as the closer of streets 1 and $n+1$. Roberts and Xu [1989] show that O_2 is an optimal orientation under measures (1) and (8) of Table 11.2 for $n \geq 7$.

(a) Draw O_2 for $G_{2,7}$. (b) Draw O_2 for $G_{2,9}$.

(c) Calculate measures (1) and (8) of Table 11.2 for $G_{2,7}$.

(d) Calculate measures (1) and (8) of Table 11.2 for $G_{2,9}$.

(e) Suppose that you alternate east-west avenues, with top and bottom going right to left and middle going left to right; and you alternate north-south streets, with street 1 going down all the way, street 2 going up all the way, and so on. Why is this not a satisfactory orientation?

(f) Modify the "alternating" orientation of part (e) by reversing the bottom half of north-south street 1 and north-south street $n+1$. Compare measures (1) and (8) of Table 11.2 for $G_{2,7}$ under this modified alternating orientation and O_2.

(g) In O_2, do we ever find orientations with arcs (x, y) and (z, y)?

(h) Calculate measures (2)–(7) of Table 11.2 for O_2 for $G_{2,7}$ and for the modified alternating orientation of part (f). Which is better?

21. Let $\bar{d}(u) = \max_x \bar{d}(u, x)$, $d(u) = \max_x d(u, x)$, and $m(u) = \bar{d}(u) - d(u)$. Then measure (8) of Table 11.2 is $\sum_u m(u)$. A *corner point* in G_{n_1,n_2} is one of the points

at the intersection of the bounding east-west avenues and north-south streets. A *limiting path* from u to v in an orientation O of graph G is a path of length $\bar{d}(u,v)$ equal to $d(u,v)$.

(a) If w is a corner point farthest from u in G_{n_1,n_2} and there is no limiting path in O from u to w, show that $m(u) \geq 2$.

(b) Illustrate the result of part (a) with orientation O_2 of Exercise 20.

(c) Repeat part (b) with the orientation of part (f) of Exercise 20.

22. Suppose that $n_1 = 2k_1 + 1$ is odd, $i = k_1$ or $k_1 + 1$. Suppose that w is a corner point (Exercise 21) in G_{n_1,n_2} farthest from the point u on the ith east-west avenue and jth north-south street and w' is the opposite corner point on the same north-south street as w.

(a) If there is no limiting path from u to w', show that $m(u) \geq 1$ (see Exercise 21).

(b) Illustrate the result of part (a) with orientation O_2 of Exercise 20.

(c) Repeat part (b) with the orientation of part (f) of Exercise 20.

23. (Bermond, *et al.* [2000]) Let $D = \max_{u,v} d(u,v)$ and consider the graph $AN(c,s)$.

(a) Show that if $c \leq \left\lfloor \frac{s}{4} \right\rfloor$, then $D = 2c$.

(b) Show that if $c \geq \left\lfloor \frac{s}{4} \right\rfloor$, then $D = c + \left\lfloor \frac{s}{4} \right\rfloor$.

24. (Bermond, *et al.* [2000]) Let O_3 be the following orientation of $AN(c,s)$. Alternate orientations of the spokes with the first spoke oriented in from outer circle to center. If s is odd, spokes 1 and s will have the same orientation. Orient circles as follows: The first circle (starting from the center) is oriented counterclockwise, the next circle clockwise, the next counterclockwise, and so on, through to the next-to-last circle. On the outer circle, orient arcs going from spoke 2 to spoke 1 and spoke 2 to spoke 3, spoke 4 to spoke 3 and spoke 4 to spoke 5, and so on. If s is odd, the part of the outer circle from spoke 2 to spoke 1 to spoke s forms the only path of length 2.

(a) If $\bar{D} = \max_{u,v} \bar{d}(u,v)$ for O_3 and $D = \max_{u,v} d(u,v)$ for $AN(c,s)$, show that $\bar{D} \leq D + 2$ if $c \leq k$, where $s = 4k + r$, $r < 4$.

(b) By the same reasoning with O_3, show that $\bar{D} \leq D + 3$ if s is odd and $c = k + 1$.

25. One of the major concerns with transportation networks, communication networks, telephone networks, electrical networks, and so on, is to construct them so that they are not very vulnerable to disruption. Motivated by the notion of bridge, we shall be interested in Exercises 25–33 in the disruption of connectedness that arises from the removal of arcs or vertices. We say that a digraph D is in *connectedness category 3* if it is strongly connected, in *category 2* if it is unilaterally connected but not strongly connected, in *category 1* if it is weakly connected but not unilaterally connected, and in *category 0* if it is not weakly connected. Find the connectedness category of every digraph of Figure 3.7.

26. If digraph D is in connectedness category i (Exercise 25), we say that arc (u,v) is an (i,j)-*arc* if removal of (u,v) (but not vertices u and v) results in a digraph of category j. For each pair (i,j), $i = 0,1,2,3$, and $j = O,1,2,3$, either give an example of a digraph with an (i,j)-arc or prove there is no such digraph.

www.ebook3000.com

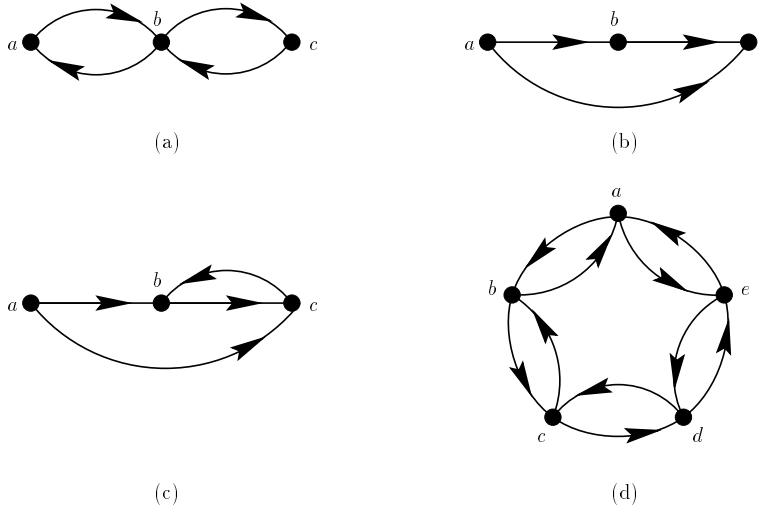


Figure 11.17: Digraphs for exercises of Section 11.2.

27. If digraph D is in connectedness category i, we say that vertex u is an (i,j)-*vertex* if the subgraph generated by vertices of D other than u is in category j. For each pair $(i,j), i = 0,1,2,3$ and $j = 0,1,2,3$, either give an example of a digraph with an (i,j)-vertex or prove that there is no such digraph.

28. Let us define the *arc vulnerability* of a digraph in connectedness categories $1, 2$, or 3 as the minimum number of arcs whose removal results in a digraph of lower connectedness category. For each digraph of Figure 11.17, determine its arc vulnerability.

29. Give an example of a digraph D with arc vulnerability equal to 4.

30. For every k, give an example of a digraph D with arc vulnerability equal to k.

31. Give an example of a digraph with n vertices and arc vulnerability equal to $n-1$. Could there be a strongly connected digraph with n vertices and arc vulnerability equal to n?

32. (a) If D is strongly connected, what is the relation between the arc vulnerability of D and the minimum *indegree* of a vertex, the minimum number of incoming arcs of a vertex?

    (b) Show that the arc vulnerability of a strongly connected digraph is at most a/n, where a is the number of arcs and n the number of vertices.

33. (Whitney [1932]) If there is a set of vertices in a digraph D whose removal results in a digraph in a lower connectedness category than D, we define the *vertex vulnerability* of D to be the size of the smallest such set of vertices. Otherwise, vertex vulnerability is undefined.

    (a) Show that if D is weakly connected and there is at least one pair $u \neq v$ such that neither (u,v) nor (v,u) is an arc of D, then the vertex vulnerability of D is less than or equal to the minimum total degree of any vertex of D. [The *total degree* of a vertex x is the sum of the *indegree* (the number of incoming arcs) of x and the *outdegree* (the number of outgoing arcs) of x.]

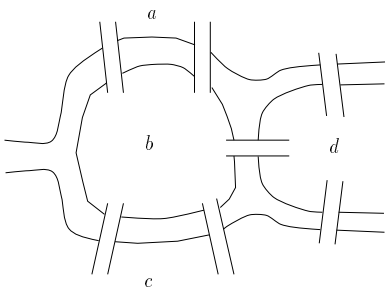


Figure 11.18: The Königsberg bridges.

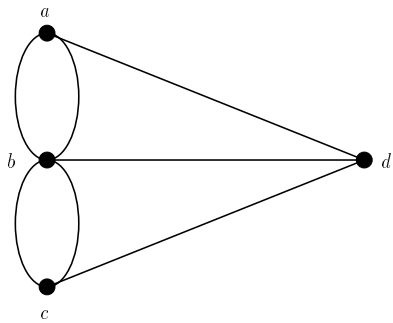


Figure 11.19: Multigraph obtained from Figure 11.18.

(b) Show that the vertex vulnerability and the minimum total degree can be different.

(c) Why do we need to assume that there is at least one pair $u \neq v$ such that neither (u, v) nor (v, u) is an arc of D?

11.3 EULERIAN CHAINS AND PATHS

11.3.1 The Königsberg Bridge Problem

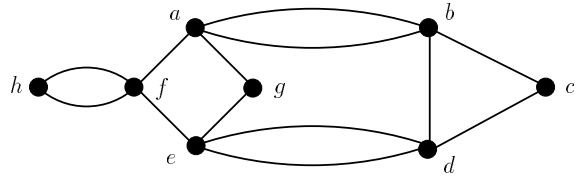
Graph theory was invented by the famous mathematician Leonhard Euler [1736] in the process of settling the famous Königsberg bridge problem.[4] Euler's techniques have found modern applications in the study of street sweeping, mechanical plotting by computer, RNA chains, coding, telecommunications, and other subjects, and we survey some of these applications in Section 11.4. Here, we present the Königsberg bridge problem and then present general techniques arising from its solution.

The city of Königsberg had seven bridges linking islands in the River Pregel to the banks and to each other, as shown in Figure 11.18. The residents wanted to know if it was possible to take a walk that starts at some point, crosses each bridge exactly once, and returns to the starting point. Euler translated this into a graph theory problem by letting the various land areas be vertices and joining two vertices by one edge for each bridge joining them. The resulting object, shown in Figure 11.19, is called a *multigraph*. We shall use the terms *multigraph* and *multidigraph* when more than one edge or arc is allowed between two vertices x and y or from vertex x to vertex y. The concepts of connectedness, such as chain, circuit, path, cycle, component, and so on, are defined just as for graphs and digraphs. (Note, however, that in a multigraph, we can have a circuit from a to b to a if there are two edges between a and b.) Loops will be allowed here.

We say that a *chain* in a multigraph G or *path* in a multidigraph D is *eulerian* if it uses every edge of G or arc of D once and only once. Euler observed that the

[4] See "The Königsberg Bridges," *Scientific American, 189* (1953), 66–70.

www.ebook3000.com



G

Figure 11.20: Multigraph with an eulerian closed chain.

citizens of Königsberg were seeking an *eulerian closed chain* in the multigraph of Figure 11.19. He asked the following general question: When does a multigraph G have an eulerian closed chain? We begin by attempting to answer this question. Clearly, if G has an eulerian closed chain, G must be *connected up to isolated*[5] *vertices*, that is, at most one component of G has an edge. Moreover, an eulerian closed chain must leave each vertex x as often as it enters x, and hence each vertex x must have even degree, where the degree of x, the reader will recall, is the number of neighbors of x (except that for loops from x to x we add 2 to the degree of x). Euler discovered the following result.

Theorem 11.2 (Euler) A multigraph G has an eulerian closed chain if and only if G is connected up to isolated vertices and every vertex of G has even degree.

Before proving the sufficiency of the condition stated in Theorem 11.2, we observe that the multigraph of Figure 11.19 does not have an eulerian closed chain. For vertex a has odd degree. Thus, the citizens of Königsberg could not complete their walk. To further illustrate the theorem, we note that the multigraph G of Figure 11.20 is connected and every vertex has even degree. An eulerian closed chain is given by $a, b, c, d, e, f, h, f, a, g, e, d, b, a$. If every vertex of a multigraph G has even degree, we say that G is *even*.

11.3.2 An Algorithm for Finding an Eulerian Closed Chain

A good way to prove sufficiency in Theorem 11.2 is to describe a procedure for finding an eulerian closed chain in an even multigraph. Start with any vertex x that has a neighbor and choose any edge joining x, say $\{x, y\}$. Next, choose any edge $\{y, z\}$ joining y, making sure not to choose an edge used previously. Continue at any vertex by choosing a previously unused edge which joins that vertex. Now each time we pass through a vertex, we use up two adjoining edges. Thus, the number of unused edges joining any vertex other than x remains even at all times. It follows that any time we enter such a vertex other than x, we can leave it. Since we started at x, the number of unused edges joining x remains odd at all times. Continue the procedure until it is impossible to continue. Now since every vertex except x can be left whenever it is entered, the only way the procedure can end is

[5] A vertex is called *isolated* if it has no neighbors.

back at x. Let us illustrate it on the multigraph G of Figure 11.20. Suppose that x is c. We can use edge $\{c, b\}$ to get to b, then edge $\{b, a\}$ to get to a, then edge $\{a, g\}$ to get to g, then edge $\{g, e\}$ to get to e, then edge $\{e, d\}$ to get to d, and then edge $\{d, c\}$ to return to c. At this point, we can go no further. Note that we have found a closed chain c, b, a, g, e, d, c starting and ending at c which uses each edge of G at most once. However, the chain does not use up all the edges of G. Let us call the procedure so far "finding a closed chain from x to x," or CL CHAIN (x, x) for short.

The algorithm can now continue. Note that at this stage, we have a chain C from x to x. Moreover, every vertex has an even number of unused joining edges. Since the graph is connected up to isolated vertices, if there is any unused edge, there must be at least one unused edge joining a vertex u on the chain C. In the multigraph of unused edges, we apply the procedure CL CHAIN (u, u) to find a closed chain D from u to u which uses each previously unused edge at most once. We can now modify our original closed chain C from x to x by inserting the "detour" D when we first hit u. We get a new closed chain C' from x to x which uses each edge of G at most once. If there are still unused edges, we repeat the procedure. We must eventually use up all the edges of the original multigraph. Continuing with our example, note that we have so far obtained the closed chain $C = c, b, a, g, e, d, c$. One unused edge joining this chain is the edge $\{a, f\}$. Since a is on C, we look for a closed chain of unused edges from a to a. Such a chain is found by our earlier procedure CL CHAIN (a, a). One example is $D = a, f, e, d, b, a$. Note that we used the second edges $\{e, d\}$ and $\{b, a\}$ here. The first ones have already been used. We insert the detour D into C, obtaining the closed chain $C' = c, b, a, f, e, d, b, a, g, e, d, c$. Since there are still unused edges, we repeat the process again. We find a vertex f on C' that joins an unused edge, and we apply CL CHAIN (f, f) to find a closed chain f, h, f from f to f. We insert this detour into C' to find $C'' = c, b, a, f, h, f, e, d, b, a, g, e, d, c$, which uses each edge once and only once.

The algorithm we have described can now be formalized. We have a subroutine called CL CHAIN (x, x) which is described as follows.

Algorithm 11.3: CL CHAIN (x, x)

Input: A multigraph G, a set U of unused edges of G, with each vertex appearing in an even number of edges in U, and a designated vertex x that appears in some edge of U.

Output: A closed chain from x to x that uses each edge of U at most once.

Step 1. Set $v = x$ and output x. (Here, v is the last vertex visited.)

Step 2. If there is an edge $\{v, y\}$ in U, set $v = y$, output y, remove $\{v, y\}$ from U, and repeat this step.

Step 3. If there is no edge $\{v, y\}$ in U, stop. The outputs in order give us the desired chain from x to x.

Using Algorithm 11.3, we can now summarize the entire procedure.

www.ebook3000.com

Algorithm 11.4: Finding an Eulerian Closed Chain

Input: An even multigraph G that is connected up to isolated vertices.
Output: An eulerian closed chain.

Step 1. Find any vertex x that has a neighbor. (If there is none, every vertex is isolated and any vertex x alone defines an eulerian closed chain. Output this and stop.) Let $U = E(G)$.

Step 2. Apply CL CHAIN (x, x) to obtain a chain C.

Step 3. Remove all edges in C from U. (Technically, this is already done in step 2.)

Step 4. If $U = \emptyset$, stop and output C. If $U \neq \emptyset$, find a vertex u on C that has a joining edge in U, and go to step 5.

Step 5. Apply CL CHAIN (u, u) to obtain a chain D.

Step 6. Redefine C by inserting the detour D at the first point u is visited, remove all edges of D from U, and go to step 4.

For a more formal statement of this algorithm, see Even [1979]. Even shows that the algorithm can be completed in the worst case in a number of steps that is a constant k times the number of edges e. Hence, if G is a graph (i.e., if there are no multiple edges), the computational complexity of the algorithm we have described is

$$ke \leq k\frac{n(n-1)}{2} = \frac{k}{2}n^2 - \frac{k}{2}n,$$

a polynomial bound in terms of n. [In the terminology of Section 2.18, this is an $O(e)$ or, for graphs, an $O(n^2)$ algorithm.] Since it is clear that the algorithm works, Theorem 11.2 is proved.

An alternative algorithm for finding an eulerian closed chain would use the depth-first search procedure of Section 11.1. We leave the description of such an algorithm to the reader (Exercise 18).

11.3.3 Further Results about Eulerian Chains and Paths

The next theorem tells when a multigraph has an eulerian chain (not necessarily closed). We leave the proof to the reader.

Theorem 11.3 (Euler) A multigraph G has an eulerian chain if and only if G is connected up to isolated vertices and the number of vertices of odd degree is either 0 or 2.

According to this theorem, the multigraph of Figure 11.19 does not have an eulerian chain. However, the multigraph of Figure 11.21 does, since there are exactly two vertices of odd degree.

We now state analogous results for multidigraphs, leaving the proofs to the reader. In these theorems, the *indegree* (*outdegree*) of a vertex is the number of

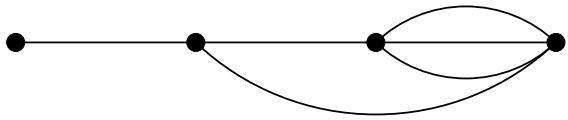
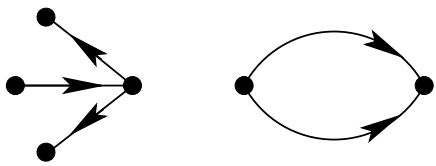


Figure 11.21: Multigraph with an eulerian chain.



(a) (b)

Figure 11.22: Two multidigraphs without eulerian paths.

incoming (outgoing) arcs. A digraph (multidigraph) is called *weakly connected* if when all directions on arcs are disregarded, the resulting graph (multigraph) is connected.

Theorem 11.4 (Good [1946]) A multidigraph D has an eulerian closed path if and only if D is weakly connected up to isolated vertices[6] and for every vertex, indegree equals outdegree.

Theorem 11.5 (Good [1946]) A multidigraph D has an eulerian path if and only if D is weakly connected up to isolated vertices and for all vertices with the possible exception of two, indegree equals outdegree, and for at most two vertices, indegree and outdegree differ by one.

Theorem 11.5 is illustrated by the two multidigraphs of Figure 11.22. Neither has an eulerian path. In the first example, there are four vertices where indegree is different from outdegree. In the second example, there are just two such vertices, but for each the indegree and outdegree differ by more than 1. Note that the hypotheses of Theorem 11.5 imply that if indegree and outdegree differ for any vertex, then exactly one vertex has an excess of one indegree and exactly one vertex has an excess of one outdegree (see Exercise 9). These vertices turn out to correspond to the first and last vertices of the eulerian path.

Note that Theorems 11.2–11.5 hold if there are loops. Note also that in a multigraph, a loop adds 2 to the degree of a vertex, while in a multidigraph, it adds 1 to indegree and 1 to outdegree. Thus, loops do not affect the existence of eulerian (closed) chains or paths.

[6] That is, when directions on arcs are disregarded, the underlying multigraph is connected up to isolated vertices.

www.ebook3000.com

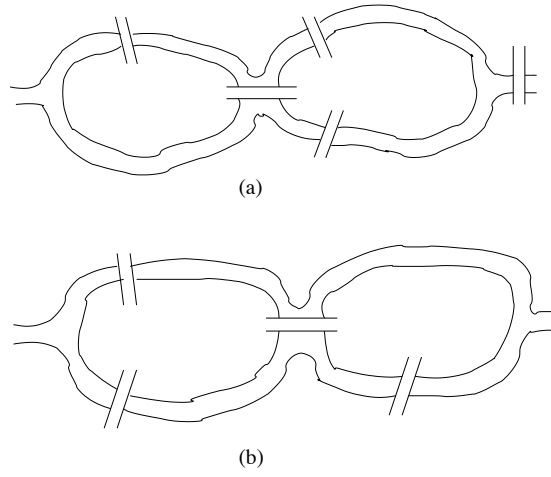


Figure 11.23: Rivers and bridges for Exercise 1, Section 11.3.

EXERCISES FOR SECTION 11.3

1. For each river with bridges shown in Figure 11.23, build a corresponding multigraph and determine if it is possible to take a walk that starts at a given location, crosses each bridge once and only once, and returns to the starting location.
2. Which of the multigraphs of Figure 11.24 have an eulerian closed chain? For those that do, find one.
3. Of the multigraphs of Figure 11.24 that do not have an eulerian closed chain, which have an eulerian chain?
4. Which of the multidigraphs of Figure 11.25 have an eulerian closed path? For those that do, find one.
5. Of the multidigraphs of Figure 11.25 that do not have an eulerian closed path, which have an eulerian path?
6. For each multigraph G of Figure 11.26, apply the subroutine CL CHAIN (a, a) to the vertex a with $U = E(G)$.
7. For each multigraph of Figure 11.26, apply Algorithm 11.4 to find an eulerian closed chain.
8. How would you modify Algorithm 11.4 to find an eulerian chain from x to y?
9. Show that in a multidigraph with an eulerian path but no eulerian closed path, exactly one vertex has an excess of one indegree and exactly one vertex has an excess of one outdegree.
10. (a) Can the drawing of Figure 11.27(a) be made without taking your pencil off the paper or retracing?
    (b) What about Figure 11.27(b)? (c) What about Figure 11.27(c)?
    (d) What about Figure 11.27(d)?

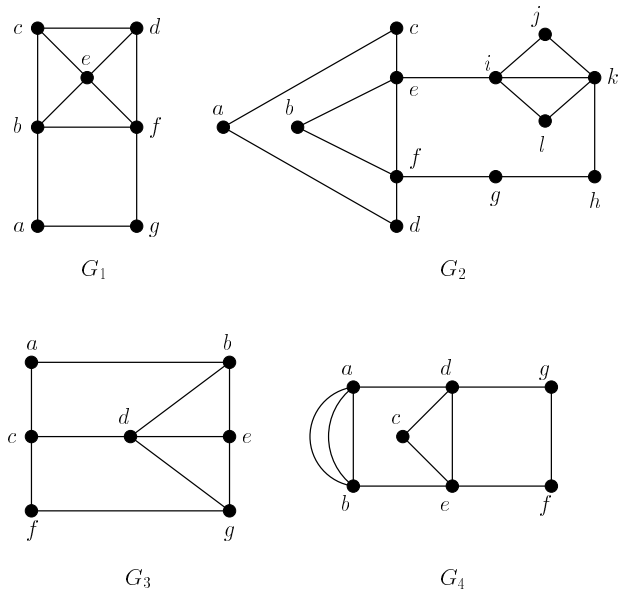


Figure 11.24: Multigraphs for exercises of Section 11.3.

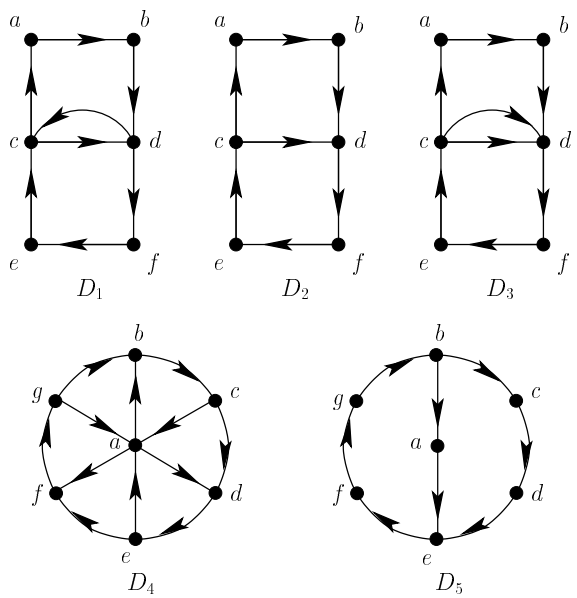


Figure 11.25: Multidigraphs for exercises of Section 11.3.

www.ebook3000.com

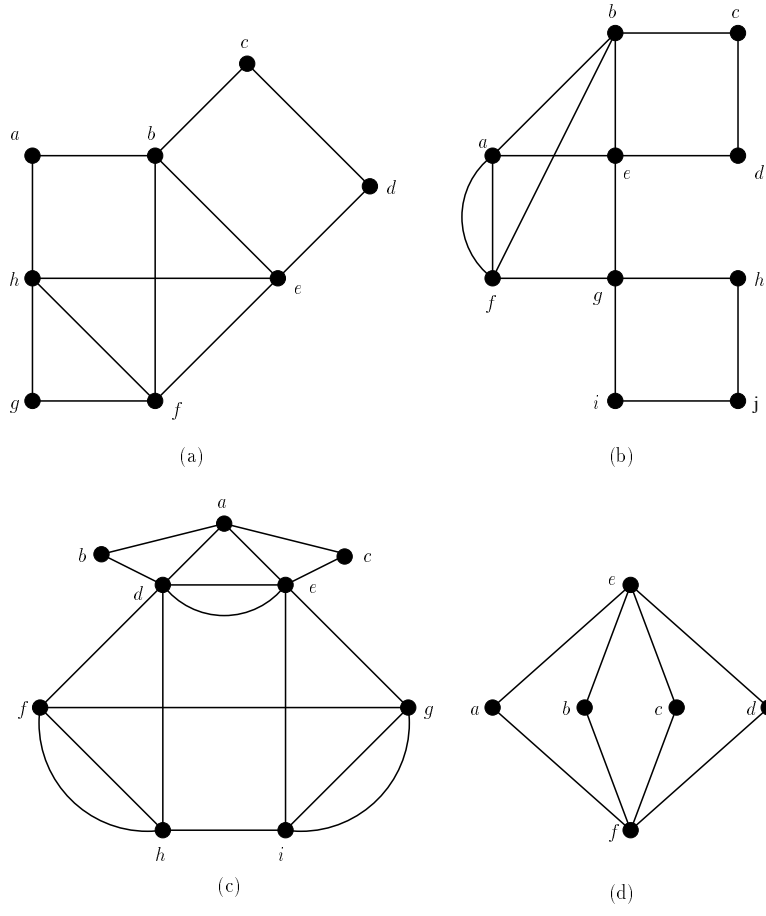


Figure 11.26: Multigraphs for exercises of Section 11.3.

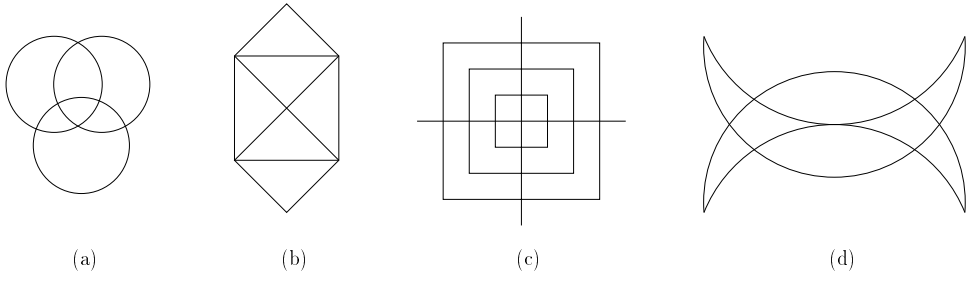


Figure 11.27: Drawings for Exercise 10, Section 11.3.

11. (Harary and Palmer [1973]) Show that the number of labeled even graphs of n vertices equals the number of labeled graphs of $n-1$ vertices. (*Hint:* To a labeled graph G of $n-1$ vertices, add a vertex labeled n and join it to the vertices of G of odd degree.)

12. (a) For each multidigraph of Figure 11.25 that has an eulerian closed path, find the number of such paths starting and ending at a.

    (b) For each multidigraph of Figure 11.25 that doesn't have an eulerian closed path but has an eulerian path, find the number of such paths.

13. Find the number of eulerian closed chains starting and ending at a for the graph of Figure 11.26(d).

14. Show that every digraph without isolated vertices and with an eulerian closed path is strongly connected.

15. Show that every digraph without isolated vertices and with an eulerian path is unilaterally connected.

16. Does every strongly connected digraph have an eulerian closed path? Why?

17. Does every unilaterally connected digraph have an eulerian path? Why?

18. Describe an algorithm for finding an eulerian closed chain that uses the depth-first search procedure.

19. Prove the necessity part of Theorem 11.3.

20. Prove the sufficiency part of Theorem 11.3.

21. Prove Theorem 11.4.

22. Prove Theorem 11.5.

11.4 APPLICATIONS OF EULERIAN CHAINS AND PATHS

In this section we present various applications of the ideas presented in Section 11.3. The subsections of this section are independent, except that Sections 11.4.2 and 11.4.3 depend on Section 11.4.1. Other than this, the subsections may be used in any order. The reader without the time to cover all of these subsections might sample Sections 11.4.1, 11.4.2, and 11.4.4 or 11.4.6.

11.4.1 The "Chinese Postman" Problem

A mail carrier starting out from a post office must deliver letters to each block in a territory and return to the post office. What is the least amount of walking the mail carrier can do? This problem was originally studied by Kwan [1962], and has traditionally been called the *"Chinese Postman" Problem*, or just the *Postman Problem*. A similar problem is faced by many delivery people, by farmers who have fields to seed, by street repair crews, and so on.

www.ebook3000.com

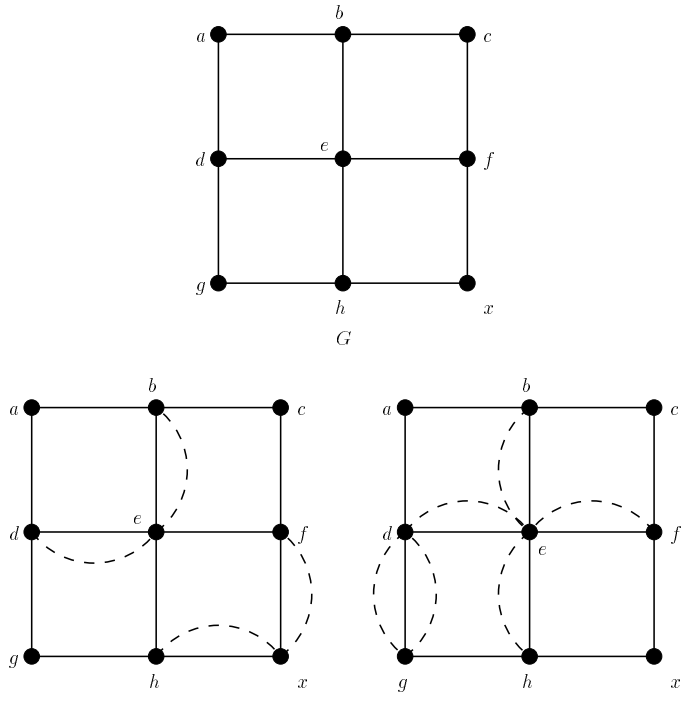


Figure 11.28: Graph representing a mail carrier's territory and two multigraphs corresponding to mail carriers' routes.

We can represent the mail carrier's problem by building a graph G with each vertex representing a street corner and each edge a street.[7] Assuming for simplicity that the post office is near a street corner, the mail carrier then seeks a closed chain in this graph, which starts and ends at the vertex x corresponding to the corner where the post office is located, and which uses each edge of the graph *at least once*.

If the graph G has an eulerian closed chain, we can pick up this chain at x and follow it back to x. No chain can give a shorter route. If there is no such eulerian closed chain, we can formulate the problem as follows. Any mail carrier's route will use all of the edges of G once and possibly some of them more than once. Suppose that we modify G by replacing each edge by as many copies of it as are in the mail carrier's route, or equivalently, by adding to G enough copies of each edge to exactly achieve the mail carrier's route. Then in the resulting multigraph, the mail carrier's route corresponds to an eulerian closed chain from x to x. For instance, consider the graph G of Figure 11.28, which represents a four-square-block area in a city. There is no eulerian closed chain in G, because for instance vertex d has degree 3. A possible mail carrier's route would be the closed chain $x, h, g, d, e, f, c, b, a, d, e, b, e, h, x, f, x$.

[7] This already simplifies the problem. A street between x and y with houses on both sides should really be represented by two edges between x and y. A mail carrier can walk up one side of the street first, and later walk up (or down) the other side.

This corresponds to the first multigraph shown in Figure 11.28. (Added edges are dashed.) An alternative route would be $x, h, e, h, g, d, e, d, g, d, a, b, e, b, c, f, e, f, x$. This corresponds to the second multigraph shown in Figure 11.28. The first of these routes is the shorter; equivalently, it requires the addition of fewer copies of edges of G. (The second route can be shortened by omitting the d, g, d part.)

The problem of trying to find the shortest mail carrier's route from x to x in G is equivalent to the problem of determining the smallest number of copies of edges of G to add to G to obtain a multigraph that has an eulerian closed chain, that is, one in which all vertices have even degree. A general method for solving this combinatorial optimization problem is to translate it into the maximum-weight matching problem of the type we discuss in Chapter 12. We discuss this translation specifically in Section 12.7.1. In our example it is easy to see that since there are four vertices of odd degree, and no two are adjacent, at least four edges must be added. Hence, the first multigraph of Figure 11.28 corresponds to a shortest mail carrier's route.

In Section 11.4.3 we generalize this problem. For a good general discussion of the "Chinese Postman" Problem (and eulerian "tours"), see Lawler [1976], Minieka [1978], and Johnson [2000].

11.4.2 Computer Graph Plotting

Reingold and Tarjan [1981] point out that the "Chinese Postman" Problem of Section 11.4.1 arises in mechanical plotting by computer[8] of a graph with prespecified vertex locations. Applications of mechanical graph plotting described by Reingold and Tarjan include shock-wave propagation problems, where meshes of thousands of vertices must be plotted; map drawing; electrical networks; and activity charts (see Reingold and Tarjan's paper for references).

Much time is wasted in mechanical graph plotting when the plotter pen moves with the pen off the paper. Thus, we seek to minimize the number of moves from vertex to vertex with the pen off the paper. This is exactly the problem of minimizing the number of edges to be added to the graph being plotted so as to obtain a multigraph with an eulerian closed chain.

11.4.3 Street Sweeping

A large area for applications of combinatorial techniques is the area of urban services. Cities spend billions of dollars a year providing such services. Combinatorics has been applied to problems involving the location and staffing of fire and police stations, design of rapid transit systems, assignments of shifts for municipal workers, routing of street-sweeping and snow-removal vehicles, and so on. See Beltrami [1977], Dror [2000], and Helly [1975] for a variety of examples. We discussed in Example 3.14 the problem of routing garbage trucks to pick up garbage. Here

[8] Although still used for CAD applications, pen plotters are quickly becoming obsolete, due to the advent of ink jet and laser printers.

www.ebook3000.com

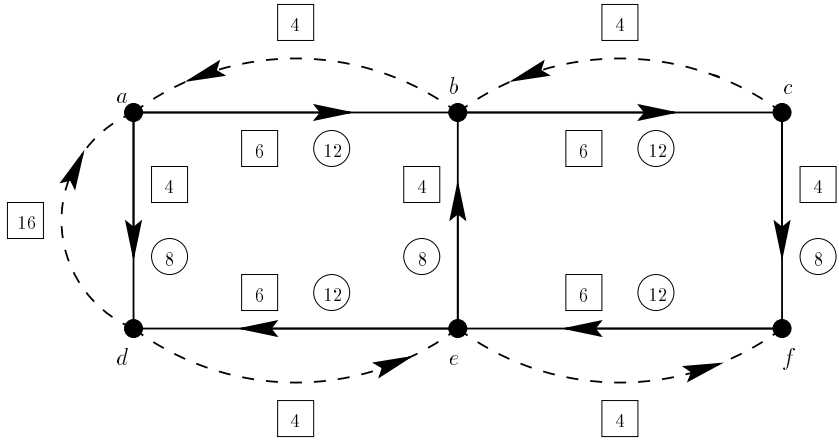


Figure 11.29: Curb multidigraph, with solid arcs defining the sweep subgraph, deadheading times in squares, and sweep times in circles.

we discuss the problem of determining optimal routes for street-sweeping or snow-removal equipment. We follow Tucker and Bodin [1983] and Roberts [1978]; see also Liebling [1970].

Consider the street-sweeping problem for concreteness. Let the street corners in the neighborhood to be swept be the vertices of a multidigraph. Include an arc from x to y if there is a curb that can be traveled from x to y. In general, a one-way street will give rise to two arcs from x to y, which is why we get a multidigraph. Call this multidigraph the *curb multidigraph*.

During a given period of time, certain curbs are to be swept. The corresponding arcs define a subgraph of the curb multidigraph called the *sweep subgraph*. (By means of parking regulations, curbs to be swept in a large city such as New York are kept free of cars during the period in question.)

Now any arc in the sweep subgraph has associated with it a number indicating the length of time required to sweep the corresponding curb. Also, any arc in the curb multidigraph has associated with it a number indicating the length of time required to follow the corresponding curb without sweeping it. This is called the *deadheading time*.

Figure 11.29 shows a curb multidigraph. Some arcs are solid—these define the sweep subgraph. Each arc has a number in a square; this is the deadheading time. The solid arcs have in addition a number in a circle; this is the sweep time.

We would like to find a way to start from a particular location (the garage), sweep all the curbs in the sweep subgraph, return to the start, and use as little time as possible. This is a generalization of the "Chinese Postman" Problem studied in Section 11.4.1, and our approach to it is similar. We seek a closed path in the curb multidigraph which includes all arcs of the sweep subgraph. The time associated with any acceptable path is the sum of the sweeping times for arcs swept plus the sum of the deadheading times for arcs in the path that are not swept. Note that

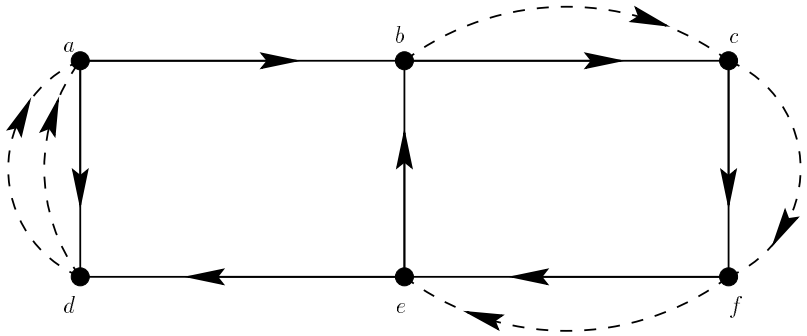


Figure 11.30: Multidigraph with an eulerian closed path (corresponding to (11.1)) obtained from the sweep subgraph of Figure 11.29 by adding the dashed arcs.

an arc can appear in the path several times. Even if it is a solid arc, we count the sweeping time only once. It is swept only the first time it is traversed. For each other appearance of the arc, we count the deadheading time.

If the sweep subgraph has an eulerian closed path, then, as in the "Chinese Postman" Problem, this path must be the minimal time street-sweeping route. In our example, Theorem 11.4 implies that there could be no such path. For there are a number of vertices in the sweep subgraph for which indegree is different from outdegree. Thus, a solution to the street-sweeping problem will be a closed path P of the curb multidigraph using some arcs not in the sweep subgraph. Suppose that we add to the sweep subgraph all the arcs used in P for deadheading, either because they are not in the sweep subgraph or because they have previously been swept in P. Add these arcs as many times as they are used. Adding these arcs gives rise to a multidigraph in which the path P is an eulerian closed path. For instance, one closed path in the curb multidigraph of Figure 11.29 that uses all arcs of the sweep subgraph is given by

$$f, e, b, c, f, e, d, a, d, a, b, c, f \tag{11.1}$$

(assume that the garage is at f). Path (11.1) corresponds to adding to the sweep subgraph the dashed arcs shown in Figure 11.30.

In general, we want the sum of the deadheading times on the added arcs to be minimized,[9] and we want to add deadheading arcs so that in the resulting multidigraph, there is an eulerian closed path; that is, every vertex has indegree equal to outdegree. Tucker and Bodin [1983] show how to solve the resulting combinatorial optimization problem by setting it up as a transportation problem of the type discussed in Section 13.4.1. The approach is similar to the approach used in the "Chinese Postman" Problem in Section 12.7.1. In our example, an optimal route

[9] We are omitting sources of delay other than deadheading, for example, delays associated with turns. These delays can be introduced by defining a system of penalties associated with different kinds of turns (see Tucker and Bodin [1983]).

www.ebook3000.com

is the path $f, e, d, e, b, a, b, a, d, e, b, c, f$. The total time required for this route is 72 for sweeping plus 20 for deadheading. The total time required for the route (11.1) is 72 for sweeping plus 48 for deadheading, which is considerably worse.

11.4.4 Finding Unknown RNA/DNA Chains

In this section we consider the problem of finding an unknown RNA or DNA string from a set of fragments (substrings) of the unknown string.

Example 11.1 Complete Digest by Enzymes In Section 2.12 we discussed the problem of finding an RNA chain given knowledge of its fragments after applying two enzymes, one of which breaks up the chain after each G link and the other of which breaks up the chain after each U or C link. Here, we show how to use the notion of eulerian closed path to find all RNA chains with given G and U, C fragments.

If there is only one G fragment or only one U, C fragment, the chain is determined. Hence, we shall assume that there are at least two G fragments and at least two U, C fragments.

We shall illustrate the procedure with the following G and U, C fragments:

G fragments: CCG, G, UCACG, AAAG, AA
U, C fragments: C, C, GGU, C, AC, GAAAGAA.

We first break down each fragment after each G, U, or C: for instance, breaking fragment GAAAGAA into G · AAAG · AA, GGU into G · G · U, and UCACG into U · C · AC · G. Each piece is called an *extended base*, and all extended bases in a fragment except the first and last are called *interior extended bases*. Using this further breakup of the fragments, we first observe how to find the beginning and end of the desired RNA chain. We make two lists, one giving all interior extended bases of all fragments from both digests, and one giving all fragments consisting of one extended base. In our example, we obtain the following lists:

Interior extended bases: C, C, AC, G, AAAG
Fragments consisting of one extended base: G, AAAG, AA, C, C, C, AC.

It is not hard to show that every entry on the first list is on the second list (Exercise 17). Moreover, the first and last extended bases in the entire chain are on the second list but not the first (Exercise 17). Since we are assuming that there are at least two G fragments and at least two U, C fragments, it is not hard to show that there will always be exactly two entries on the second list which are not on the first list (Exercise 17). One of these will be the first extended base of the RNA chain and one will be the last. In our example, these are AA and C. How do we tell which is last? We do it by observing that one of these entries will be from an *abnormal fragment*, that is, it will be the last extended base of a G fragment not ending in G or a U, C fragment not ending in U or C. In our example, AA is the last extended base of two abnormal fragments AA and GAAAGAA. This implies that the chain we are seeking begins in C and ends in AA.

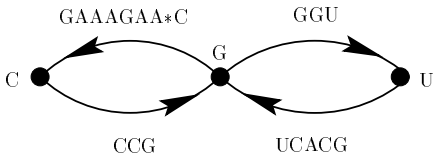


Figure 11.31: Multidigraph for reconstruction of RNA chains from complete enzyme digest.

To find all possible chains, we build a multidigraph as follows. First identify all normal fragments with more than one extended base. From each such fragment, use the first and last extended bases as vertices and draw an arc from the first to the last, labeling it with the corresponding fragment. (We shall add one more arc shortly.) Figure 11.31 illustrates this construction. For example, we have included an arc from U to G, labeled with the name of the corresponding fragment UCACG. Similarly, we add arcs from C to G and G to U. There might be several arcs from a given extended base to another, if there are several normal fragments with the same first and last extended base. Finally, we add one additional arc to our multidigraph. This is obtained by identifying the longest abnormal fragment—here this is GAAAGAA—and drawing an arc from the first (and perhaps only) extended base in this abnormal fragment to the first extended base in the chain. Here, we add an arc from G to C. We label this arc differently, by marking it $X*Y$, where X is the longest abnormal fragment, $*$ is a symbol marking this as a special arc, and Y is the first extended base in the chain. Hence, in our example, we label the arc from G to C by GAAAGAA*C. Every possible RNA chain with the given G and U, C fragments can now be identified from the multidigraph we have built. It turns out that each such chain corresponds to an eulerian closed path which ends with the special arc $X*Y$ (Exercise 20). In our example, the only such eulerian closed path goes from C to G to U to G to C. By using the corresponding arc labeling, we obtain the chain

CCGGUCACGAAAGAA.

It is easy to check that this chain has the desired G and U, C fragments. For more details on this graph-theoretic approach, see Hutchinson [1969]. ■

Example 11.2 Sequencing by Hybridization[10] A problem that arises in "sequencing" of DNA is the problem of determining a DNA string S from the list of all k-length substrings that appear in S. Known as the *sequencing by hybridization* or *SBH problem*, its solution again uses the notion of eulerian path (see Pevzner [1989]).

Consider the four bases A, C, G, T found in DNA. Let L be the list of all k-length substrings of the string S. [The data are obtained from a *DNA array* (*DNA chip*) with synthetic fragments ("probes") corresponding to all 4^k DNA sequences of length k. If a solution containing a fluorescently-labeled DNA fragment F of

[10] Our discussion follows Gusfield [1997] and Pevzner [2000].

www.ebook3000.com

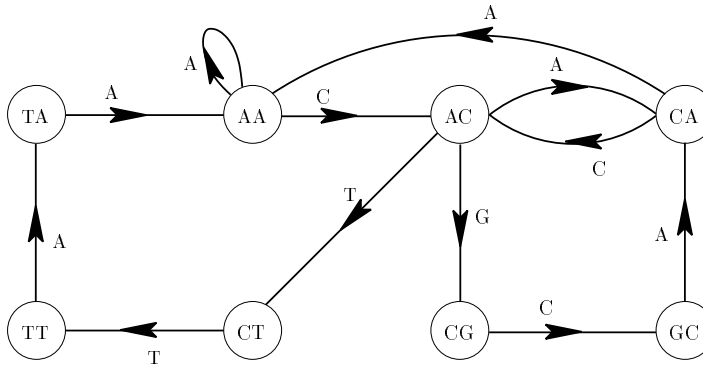


Figure 11.32: Digraph $D(L)$, where L is given by (11.2).

an unknown sequence is applied to the array, it hybridizes with those probes corresponding to substrings of length k in F, corresponding in the sense that A and T are reversed and G and C are reversed.] For example, if $S = \text{ACACGCAACTTAAA}$ and $k = 3$, then

$$L = \{\text{ACA}, \text{CAC}, \text{ACG}, \text{CGC}, \text{GCA}, \text{CAA}, \text{AAC}, \text{ACT}, \text{CTT}, \text{TTA}, \text{TAA}, \text{AAA}\}. \tag{11.2}$$

(We shall assume that no k-length substring appears more than once in S, as in this example. In practice, a value of $k \approx 10$ suffices for this purpose.) We build the digraph $D(L)$ as follows. Let each possible $(k-1)$-length DNA string be a vertex of $D(L)$. Thus, $|V(D(L))| = 4^{k-1}$. For each string $l = b_1 b_2 \cdots b_k$ in L, there is an associated arc a_l in $D(L)$. Arc a_l goes from the vertex representing the first $k-1$ bases of l to the vertex representing the last $k-1$ bases of l. So, string l corresponds to arc

$$a_l = (b_1 b_2 \cdots b_{k-1}, b_2 b_3 \cdots b_k)\,.$$

We also label arc a_l with the rightmost base b_k of l. Finally, we remove all isolated vertices of $D(L)$ (see Figure 11.32).

If P is a path in $D(L)$, there is an associated string S_P constructed from P in the following way: String S_P starts with the label of the first vertex in P and continues with the labels on the arcs of P. In Figure 11.32, for example, the path $P = \text{AC}, \text{CG}, \text{GC}, \text{CA}, \text{AC}, \text{CA}, \text{AA}, \text{AA}, \text{AC}, \text{CT}, \text{TT}, \text{TA}, \text{AA}$ yields the string $S_P = \text{ACGCACAAACTTAA}$. The path P is said to *specify* the string S_P. The digraphs $D(L)$ are actually subgraphs of the de Bruijn diagrams presented in Section 11.4.6.

Given a set L of k-length substrings, a string S is called *compatible* with L if S contains every substring in L and S contains no other substrings of length k. The goal, then, is to find those strings (and hopefully, only one) compatible with a dataset L of k-length DNA substrings. The proof of the next theorem is left for Exercise 16.

Theorem 11.6 A string S is compatible with L iff S is specified by an eulerian path in $D(L)$.

It should be clear from the construction above that there is a one-to-one correspondence between strings that are compatible with L and eulerian paths in $D(L)$. So, assuming that each substring in L occurs exactly once in a DNA string S, L uniquely determines S iff $D(L)$ has a unique eulerian path. (van Aardenne-Ehrenfest and de Bruijn [1951] give a formula for the number of eulerian paths in a digraph D.)

Finishing with our example, notice that each vertex in the digraph $D(L)$ of Figure 11.32 has indegree equal to outdegree except for two vertices, AC and AA. For these vertices, outdegree(AC) $=$ 1 $+$ indegree(AC), while indegree(AA) $=$ 1 $+$ outdegree(AA). Therefore, $D(L)$ has an eulerian path.

Although the goal of DNA sequence reconstruction from k-length substrings seems to be attainable in theory, sequencing by hybridization is not as simple as we have described. It is common that errors are made in producing the substrings in L from an unknown string S. Also, our assumption of nonrepeated substrings in L is not very reasonable, due to the nature of DNA strings. For more on sequencing by hybridization, see Drmanac, *et al.* [1989], Gusfield [1997], Pevzner [2000], or Pevzner and Lipshutz [1994]. ■

11.4.5 A Coding Application

Hutchinson and Wilf [1975] study codes on an alphabet of n letters, under the following assumption: All the information in a codeword is carried in the number of letters of each type and in the frequency of ordered pairs of letters, that is, the frequency with which one letter follows a second. For instance, Hutchinson and Wilf treat a DNA or RNA molecule as a word, with bases (not extended bases) as the letters, and make the assumption above about the genetic code. Specifically, Hutchinson and Wilf ask the following question: Given nonnegative integers $v_i, v_{ij}, i, j = 1, 2, \ldots, n$, is there a word from an alphabet of n letters so that the ith letter occurs exactly v_i times and so that the ith letter is followed by the jth exactly v_{ij} times? If so, what are all such words? We shall present Hutchinson and Wilf's solution, which uses the notion of eulerian path.

To give an example, suppose that $v_1 = 2, v_2 = v_3 = 1$ and v_{ij} is given by the following matrix:

$$(v_{ij}) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \tag{11.3}$$

Then one word that has the prescribed pattern is $ABCA$, if A corresponds to the first letter, B to the second, and C to the third. To give a second example, suppose that $v_1 = 2, v_2 = 4, v_3 = 3$, and

$$(v_{ij}) = \begin{pmatrix} 0 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 2 & 0 \end{pmatrix}. \tag{11.4}$$

www.ebook3000.com

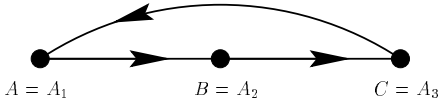


Figure 11.33: Multidigraph corresponding to (11.3).

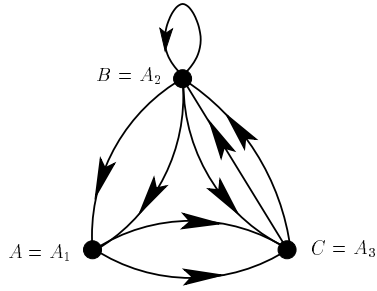


Figure 11.34: Multidigraph corresponding to (11.4).

One word that has the prescribed pattern is $BBCBACBAC$.

To analyze our problem, let us draw a multidigraph D with vertices the n letters $A_1, A_2, \ldots, A_n$, and with v_{ij} arcs from A_i to A_j. Loops are allowed. The multidigraphs corresponding to the matrices of (11.3) and (11.4) are shown in Figures 11.33 and 11.34, respectively. Let us suppose that $w = A_{i_1}, A_{i_2}, \ldots, A_{i_q}$ is a solution word. Then it is clear that w corresponds in the multidigraph D to an eulerian path that begins at A_{i_1} and ends at A_{i_q}. It is easy to see this for the two solution words we have given for our two examples. In what follows, we use the observation that a solution word corresponds to an eulerian path to learn more about solution words. The reader who wishes to may skip the rest of this subsection.

Since a solution word corresponds to an eulerian path, it follows that if there is a solution word, then D must be weakly connected up to isolated vertices. We consider first the case where $i_1 \neq i_q$. For every $i \neq i_1, i_q$, we have indegree at A_i equal to outdegree. For $i = i_1$, we have outdegree one higher than indegree, and for $i = i_q$ we have indegree one higher than outdegree. Thus, we have

$$\sum_{k=1}^{n} v_{ik} = \begin{cases} \sum_{k=1}^{n} v_{ki}, & i \neq i_1, i_q \\ \sum_{k=1}^{n} v_{ki} + 1, & i = i_1 \\ \sum_{k=1}^{n} v_{ki} - 1, & i = i_q. \end{cases} \tag{11.5}$$

This condition says that in the matrix (v_{ij}), the row sums equal the corresponding column sums, except in two places where they are off by one in the manner indicated. We also have a consistency condition, which relates the v_i to the v_{ij}:

$$v_i = \begin{cases} \sum_{k=1}^{n} v_{ki}, & i \neq i_1 \\ \sum_{k=1}^{n} v_{ki} + 1, & i = i_1. \end{cases} \tag{11.6}$$

It is easy to see, using Theorem 11.5, that if conditions (11.5) and (11.6) hold for some i_1 and $i_q, i_1 \neq i_q$, and if D is weakly connected up to isolated vertices, then there is a solution word, and every solution word corresponds to an eulerian path that begins in A_{i_1} and ends in A_{i_q}. In our second example, conditions (11.5) and (11.6) hold with $i_1 = 2, i_q = 3$. There are a number of eulerian paths from

$B = A_{i_1}$, to $C = A_{i_q}$, each giving rise to a solution word. A second example is $BACBBCBAC$.

What if the solution word begins and ends in the same letter, that is, if $i_1 = i_q$? Then there is an eulerian closed path, and we have

$$\sum_{k=1}^{n} v_{ik} = \sum_{k=1}^{n} v_{ki}, \quad i = 1, 2, \ldots, n. \tag{11.7}$$

Also, given i_1, (11.6) holds (for all i). Condition (11.7) says that in (v_{ij}), every row sum equals its corresponding column sum. Conversely, if (11.7) holds, and for i_1 (11.6) holds (for all i), and if D is weakly connected up to isolated vertices, then there is a solution and every solution word corresponds to an eulerian closed path in the multidigraph D, beginning and ending at A_{i_1}. This is the situation in our first example with $i_1 = 1$.

In sum, if there is a solution word, D is weakly connected up to isolated vertices and (11.6) holds for some i_1. Moreover, either (11.7) holds, or for i_1 and some i_q, $i_1 \neq i_q$, (11.5) holds. Conversely, suppose that D is weakly connected up to isolated vertices. If (11.6) holds for some i_1 and (11.5) holds for i_1 and some $i_q, i_1 \neq i_q$, there is a solution and all solution words correspond to eulerian paths beginning at A_{i_1} and ending at A_{i_q}. If (11.6) holds for some i_1 and (11.7) holds, there is a solution and all solution words correspond to eulerian closed paths beginning and ending at A_{i_1}.

11.4.6 De Bruijn Sequences and Telecommunications

In this subsection we consider another coding problem and its application in telecommunications. Let

$$\Sigma = \{0, 1, \ldots, p-1\}$$

be an alphabet of p letters and consider all words of length n from Σ. A (p, n) *de Bruijn sequence* (named after N. G. de Bruijn) is a sequence

$$a_0 a_1 \cdots a_{L-1} \tag{11.8}$$

with each a_i in Σ such that every word w of length n from Σ is realized as

$$a_i a_{i+1} \cdots a_{i+n-1} \tag{11.9}$$

for exactly one i, where addition in the subscripts of (11.9) is modulo L.[11] For instance,

$$01110100$$

is a $(2, 3)$ de Bruijn sequence over the alphabet $\Sigma = \{0, 1\}$ if $n = 3$. For, starting with the beginning, the three-letter words $a_i a_{i+1} a_{i+2}$ obtained are, respectively,

$$011, \quad 111, \quad 110, \quad 101, \quad 010, \quad 100, \quad 000, \quad 001.$$

[11] For a discussion of modular arithmetic, see Section 9.3.1.

www.ebook3000.com

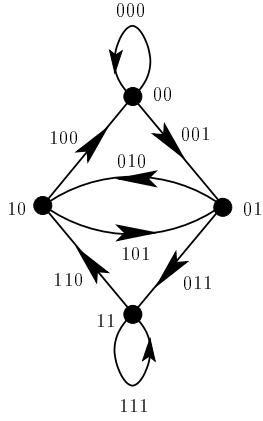


Figure 11.35: The digraph $D_{2,3}$.

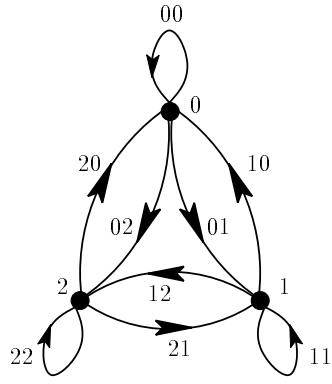


Figure 11.36: The digraph $D_{3,2}$.

The latter two are the words $a_6a_7a_0$ and $a_7a_0a_1$. De Bruijn sequences are of great significance in coding theory. They are implemented by shift registers and are sometimes called *shift register sequences*. For a detailed treatment of this topic, see Golomb [1982].

Clearly, if there is a (p, n) de Bruijn sequence (11.8), L must be p^n, where $p = |\Sigma|$. We shall show that for every positive p and n, there is a (p, n) de Bruijn sequence (11.8). Given p and n, build a digraph $D_{p,n}$, called a *de Bruijn diagram*, as follows. Let $V(D_{p,n})$ consist of all words of length $n-1$ from alphabet Σ, and include an arc from word $b_1b_2\cdots b_{n-1}$ to every word of the form $b_2b_3\cdots b_n$. Label such an arc with the word $b_1b_2\cdots b_n$. Figures 11.35 and 11.36 show the digraph $D_{p,n}$ for cases $p = 2, n = 3$ and $p = 3, n = 2$. Suppose that the de Bruijn diagram $D_{p,n}$ has an eulerian closed path. We then use successively the first letter from each arc in this path to obtain a de Bruijn sequence (11.8), where $L = p^n$. To see that this is a de Bruijn sequence, note that each word w of length n from Σ is realized. For if $w = b_1b_2\cdots b_n$, we know that the eulerian path covers the arc from $b_1b_2\cdots b_{n-1}$ to $b_2b_3\cdots b_n$. Thus, the eulerian path must go from $b_1b_2\cdots b_{n-1}$ to $b_2b_3\cdots b_n$. From there it must go to $b_3b_4\cdots$ to $b_4b_5\cdots$ to $b_n\cdots$. Thus, the first letters of the arcs in this path are $b_1b_2\cdots b_n$. It is easy to see, using Theorem 11.4, that the de Bruijn diagram of Figure 11.35 has an eulerian closed path, for this digraph is weakly connected and every vertex has indegree equal to outdegree. The de Bruijn sequence 01110100 corresponds to the eulerian closed path that goes from 01 to 11 to 11 to 10 to 01 to 10 to 00 to 00 to 01.

We shall prove the following theorem, which implies that for every pair of positive integers p and n, there is a (p, n) de Bruijn sequence. This theorem was discovered for the case $p = 2$ by de Bruijn [1946], and it was discovered for arbitrary p by Good [1946].

Theorem 11.7 For all positive integers p and n, the de Bruijn diagram $D_{p,n}$ has an eulerian closed path.

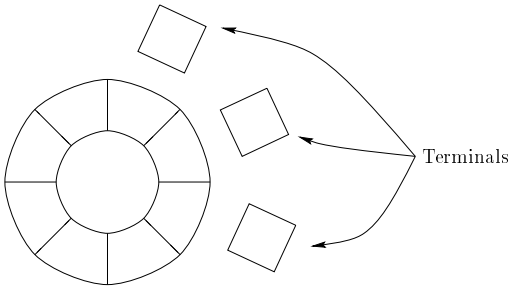


Figure 11.37: Rotating drum with eight sectors and three adjacent terminals.

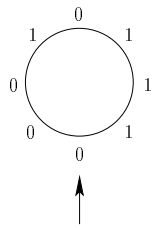


Figure 11.38: Arrangement of 0's and 1's that solves the rotating drum problem.

Proof. We first show that $D_{p,n}$ is weakly connected, indeed, strongly connected. Let $b_1b_2\cdots b_{n-1}$ and $c_1c_2\cdots c_{n-1}$ be any two vertices of $D_{p,n}$ Since we have a path

$$b_1b_2\cdots b_{n-1},\quad b_2b_3\cdots b_{n-1}c_1,\quad b_3b_4\cdots b_{n-1}c_1c_2,\quad \cdots,\quad c_1c_2\cdots c_{n-1},$$

$D_{p,n}$ is strongly connected. Next, note that every vertex has indegree and outdegree equal to p. The result follows by Theorem 11.4. Q.E.D.

We close this section by applying our results to a problem in telecommunications. We follow Liu [1968]. A rotating drum has eight different sectors. The question is: Can we tell the position of the drum without looking at it? One approach is by putting conducting material in some of the sectors and nonconducting material in other sectors. Place three terminals adjacent to the drum so that in any position of the drum, the terminals adjoin three consecutive sectors, as shown in Figure 11.37. A terminal will be activated if it adjoins a sector with conducting material. If we are clever, the pattern of conducting and nonconducting material will be so chosen that the pattern of activated and nonactivated terminals will tell us the position of the drum.

We can reformulate this as follows. Let each sector receive a 1 or a 0. We wish to arrange eight 0's and 1's around a circle so that every sequence of three consecutive digits is different. This is accomplished by finding a $(2,3)$ de Bruijn sequence, in particular the sequence 01110100. If we arrange these digits around a circle as shown in Figure 11.38, the following sequences of consecutive digits occur going counterclockwise beginning from the arrow: $011, 111, 110, 101, 010, 100, 000, 001$. These are all distinct, as desired. Thus, each position of the drum can be encoded uniquely.

Related problems concerned with teleprinting and cryptography are described in Exercises 22 and 23.

www.ebook3000.com

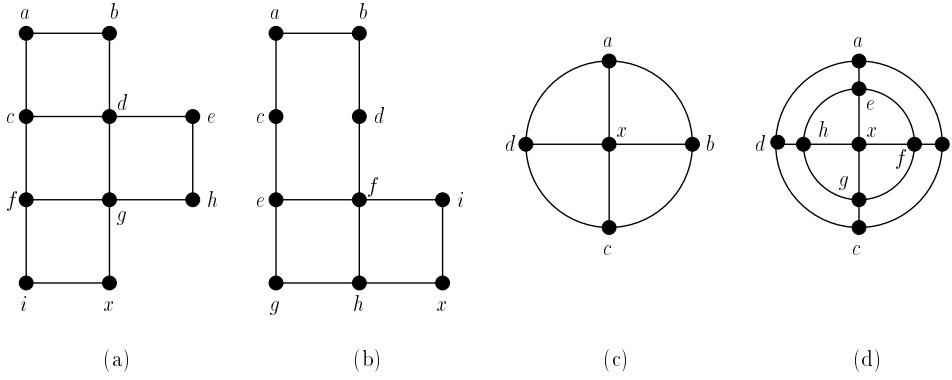


Figure 11.39: Graphs for Exercise 3 of Section 11.4.

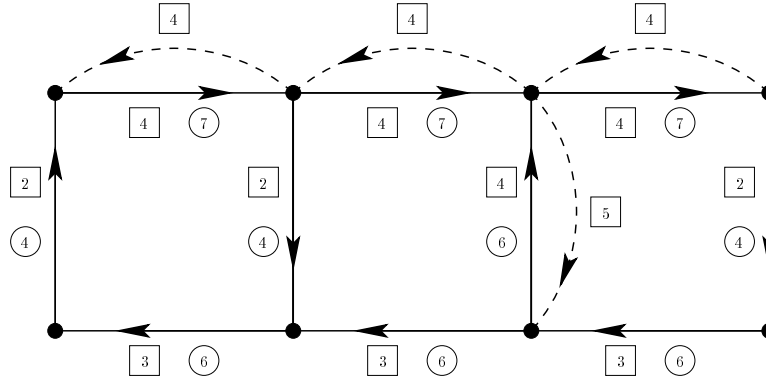


Figure 11.40: Curb multidigraph with sweep subgraph and sweeping and deadheading times shown as in Figure 11.29.

EXERCISES FOR SECTION 11.4

1. In graph G of Figure 11.28, find another way to add four copies of edges of G to obtain a multigraph with an eulerian closed chain.
2. Enumerate all eulerian paths for digraph $D(L)$ in Figure 11.32.
3. In each graph of Figure 11.39, find a shortest mail carrier's route from x to x by finding the smallest number of copies of edges of G that give rise to a multigraph with an eulerian closed chain.
4. Given the sweep subgraph and curb multidigraph of Figure 11.40, determine a set of deadheading arcs to add to the sweep subgraph to obtain a multidigraph with an eulerian closed path. Determine the total time required to traverse this path, including sweeping and deadheading.
5. Draw the de Bruijn diagram $D_{p,n}$ and find a (p, n) de Bruijn sequence for:

   (a) $p = 2, n = 4$ (b) $p = 3, n = 3$

6. For the G and U, C fragments given in the following exercises of Section 2.12, find an RNA chain using the method of Section 11.4.4.

   (a) Exercise 1 (b) Exercise 3 (c) Exercise 5

7. (a) Draw digraph $D(L)$ of Example 11.2 for the list $L = \{$CCGA, CGCG, CGTG, CGAA, GCCG, GCGT, GTGC, GAAC, TGCC, AACC, ACCC$\}$.

   (b) Find the number of eulerian paths in $D(L)$.

   (c) Find the compatible string(s) of L.

8. (a) Draw digraph $D(L)$ of Example 11.2 for the list $L = \{$AA, AC, AG, AT, CA, CT, GA, GT, TA, TC, TG, TT$\}$.

   (b) Find the number of eulerian paths in $D(L)$.

   (c) Is $S =$ ACTTCATGAGTAA compatible with L?

   (d) Is $S =$ AATGAGTACTTCA compatible with L?

   (e) Is $S =$ CTAGTACATGATA compatible with L?

9. In each of the following cases, determine if there is a codeword with the ith letter occurring exactly v_i times and the ith followed by the jth exactly v_{ij} times, and find such a word if there is one.

   (a) $v_1 = 3, v_2 = 4, v_3 = 5,$

   $$(v_{ij}) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 4 \end{pmatrix}.$$

   (b) $v_1 = 3, v_2 = 4, v_3 = 5,$

   $$(v_{ij}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix}.$$

   (c) $v_1 = 3, v_2 = 4, v_3 = 5,$

   $$(v_{ij}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}.$$

   (d) $v_1 = 3, v_2 = 3, v_3 = 3, v_4 = 3,$

   $$(v_{ij}) = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

   (e) $v_1 = 3, v_2 = 3, v_3 = 2, v_4 = 3$ and (v_{ij}) as in part (d).

10. In the "Chinese Postman" Problem, show that if x and y are any two vertices of G, the shortest mail carrier's route from x to x has the same length as the shortest mail carrier's route from y to y.

11. Notice that in the second multigraph of Figure 11.28, the dashed edges can be divided into two chains, each joining two vertices of odd degree in G. These are the chains d, e, b and h, x, f.

www.ebook3000.com

(a) Can two such chains be found in the third multigraph of Figure 11.28?

(b) In general, if H is a multigraph obtained from G by using edges in a shortest mail carrier's route, show that the dashed edges in H can be grouped into several (not necessarily two) chains, each joining two vertices of odd degree in G. Moreover, every vertex of odd degree in G is an end vertex of exactly one of these chains. (This result will be important in discussing the last part of the solution to the "Chinese Postman" Problem in Section 12.7.1.)

12. In a shortest mail carrier's route, can any edge be used more than twice? Why?

13. Suppose that we have an optimal solution to the street-sweeping problem, that is, a closed path in the curb multidigraph which includes all arcs of the sweep subgraph and uses as little total time as possible. Is it possible that some arc is traversed more than twice?

14. Given a weakly connected digraph D, does there always exist a path that uses each arc of D at least once?

15. Prove that a digraph D is strongly connected if and only if D has a circuit that uses each arc of D at least once.

16. Prove Theorem 11.6.

17. Show the following about the list of interior extended bases of the G and U, C fragments and the list of fragments consisting of one extended base.

    (a) Every entry on the first list is on the second list.

    (b) The first and last extended bases in the entire RNA chain are on the second list but not on the first.

    (c) If there are at least two G fragments and at least two U, C fragments, there will always be exactly two entries on the second list that are not on the first.

18. Under what circumstances does an RNA chain have two abnormal fragments?

19. Find the length of the smallest possible ambiguous RNA chain, that is, one whose G and U, C fragments are the same as that of another RNA chain.

20. This exercise sketches a proof[12] of the claim S: Suppose that we are given G and U, C fragments from an RNA chain with at least two G fragments and at least two U, C fragments. Then every RNA chain with given G and U, C fragments corresponds to an eulerian closed path that ends with the arc $X * Y$ in the multidigraph D constructed in Section 11.4.4. Let us say that an extended base has *type* G if it ends in G and *type* U, C if it ends in U or C. Note that any RNA chain can be written as

$$A_0 A_1 \cdots A_k A_{k+1} B,$$

where each A_i consists of extended bases of one type, the A_i alternate in type, and B is the last extended base if the chain ends in A and is the empty chain otherwise. For $i = 0, 1, \ldots, k$, let $\bar{A}_i$ be $A_i a_{i+1}$, where a_j is the first extended base in A_j. Let $\bar{A}_{k+1}$, be $A_{k+1}B$. Finally, say a fragment of one extended base is *trivial*. Show the following.

[12] The authors thank Michael Vivino for the idea behind this proof.

(a) $\bar{A}_0, \bar{A}_1, \ldots, \bar{A}_k$ are all nontrivial normal fragments and there are no other nontrivial normal fragments.

(b) $\bar{A}_{k+1}$ is the longest abnormal fragment.

(c) $a_0, a_1, \ldots, a_{k+1}$ are the vertices of the multidigraph D, $\bar{A}_i$ is the label on the arc from a_i to a_{i+1}, and $\bar{A}_{k+1} * a_0$ is the label on the arc from a_{k+1} to a_0.

(d) The arcs labeled $\bar{A}_0, \bar{A}_1, \ldots, \bar{A}_k, \bar{A}_{k+1} * a_0$ in this order define an eulerian closed path.

(e) Any other RNA chain that produces the same set of nontrivial normal fragments and the same longest abnormal fragment produces the same multidigraph D.

(f) Any RNA chain built from D produces the same set of nontrivial normal fragments and the same longest abnormal fragment.

(g) The nontrivial normal fragments and the longest abnormal fragment uniquely determine all of the fragments.

(h) Statement S holds.

21. In the problem discussed in Section 11.4.5, if there is a solution word, every eulerian path determines exactly one solution word. However, how many different eulerian paths are determined by each solution word?

22. The following problem arises in cryptography: Find a word from a given m-letter alphabet in which each arrangement of r letters appears exactly once. Find the solution to this problem if $r = 3$ and $m = 4$.

23. An important problem in communications known as the teleprinter's problem is the following: How long is the longest circular sequence of 0's and 1's such that no sequence of r consecutive bits appears more than once in the sequence? (The sequence of r bits is considered to appear if it starts near the end and finishes at the beginning.) Solve the teleprinter's problem. (It was first solved by Good [1946].)

11.5 HAMILTONIAN CHAINS AND PATHS

11.5.1 Definitions

Analogous to an eulerian chain or path in a graph or digraph is a *hamiltonian chain* or *path*, a chain or path that uses each vertex once and only once. The notion of hamiltonian chain goes back to Sir William Rowan Hamilton, who in 1857 described a puzzle that led to this concept (see below). In this section we discuss the question of existence of hamiltonian chains and paths. We discuss some applications here and others in Section 11.6.

Note that a hamiltonian chain or path is automatically simple, and hence by our conventions cannot be closed. However, we shall call a *circuit* or a *cycle* $u_1, u_2, \ldots, u_t, u_1$ *hamiltonian* if $u_1, u_2, \ldots, u_t$ is, respectively, a hamiltonian chain or path.

Although the notions of hamiltonian chain, path, and so on, are analogous to the comparable eulerian notions, it is very hard to tell if a graph or digraph has

www.ebook3000.com

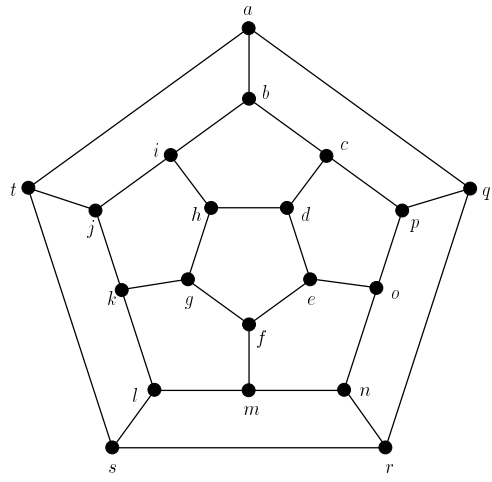


Figure 11.41: A graph representing the vertices and edges of a dodecahedron.

such an object. Indeed, it is an NP-complete problem to determine if a graph or digraph has a hamiltonian chain (path, circuit, cycle). Some results are known about existence of these objects, and we present some here. First we mention some applications of hamiltonian chains, paths, circuits, or cycles.

Example 11.3 Following the Edges of a Dodecahedron We begin with an example. Hamilton's puzzle was to determine a way to follow the edges of a dodecahedron that visited each corner exactly once and returned to the start. The vertices and edges of a dodecahedron can be drawn as a graph as in Figure 11.41. The problem becomes: Find a hamiltonian circuit in this graph. There is one: $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, a$. ■

Example 11.4 The Traveling Salesman Problem (Example 2.10 Revisited) The notion of hamiltonian cycle arises in the traveling salesman problem, which we discussed in Section 2.4. A salesman wishes to visit n different cities, each exactly once, and return to his starting point, in such a way as to minimize cost. (In Section 2.4 we mentioned other applications of the traveling salesman problem, for example to finding optimal routes for a bank courier or a robot in an automated warehouse.) Suppose that we let the cities be the vertices of a complete symmetric digraph, a digraph with all pairs of vertices joined by two arcs, and suppose that we put a weight c_{ij} on the arc (i, j) if c_{ij} is the cost of traveling from city i to city j. Since the complete symmetric digraph has a hamiltonian cycle, the existence question is not of interest. Rather, we seek a hamiltonian cycle in this digraph that has a minimum sum of weights. As we have pointed out previously, this traveling salesman problem is hard; that is, it is NP-complete. ■

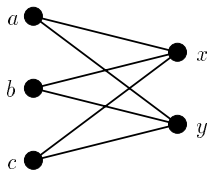


Figure 11.42: A graph without a hamiltonian circuit.

Example 11.5 Scheduling Industrial Processes A manufacturing plant has a single processing facility. A number of items are to be processed there. If item j is processed immediately after item i, there is a cost c_{ij} of resetting the processing facility from its configuration for processing item i to its configuration for processing item j. If no resetting is necessary, the cost is, of course, zero. Assuming that the costs of processing items are independent of the order in which items are processed, the problem is to choose the order so as to minimize the sum of the resetting costs c_{ij}. This problem arises in scheduling computer runs, as we observed in Example 2.17. It also arises in the chemical and pharmaceutical industries, where the processing facility might be a reaction vessel and resetting means cleaning. Costs are, of course, minimized if we can find a production schedule in which no resetting is necessary. To see if such a production schedule exists, we build a digraph D whose vertices are the items to be processed and which has an arc from i to j if j can follow i without resetting. If there is a hamiltonian path in D, such a production schedule exists. For more information on this problem, and for a treatment of the case where there is no hamiltonian path, see Christofides [1975]. See also the related discussion in Section 11.6.3. ■

11.5.2 Sufficient Conditions for the Existence of a Hamiltonian Circuit in a Graph

Not every graph has a hamiltonian circuit. Consider the graph of Figure 11.42. Note that every edge joins one of the vertices in $A = \{a, b, c\}$ to one of the vertices in $B = \{x, y\}$. Thus, a hamiltonian circuit would have to pass alternately from one of the vertices in A to one of the vertices in B. This could happen only if $|A| = |B|$.

In this section we present sufficient conditions for the existence of a hamiltonian circuit.

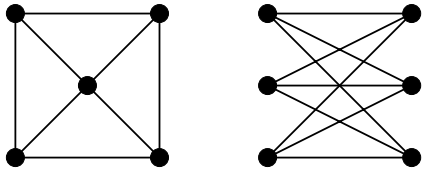
Theorem 11.8 (Ore [1960]) Suppose that G is a graph with $n \geq 3$ vertices and whenever vertices $x \neq y$ are not joined by an edge, the degree of x plus the degree of y is at least n. Then G has a hamiltonian circuit.

Proof.[13] Suppose that G has no hamiltonian circuit. We shall show that for some nonadjacent x, y in $V(G)$,

$$\deg_G(x) + \deg_G(y) < n, \tag{11.10}$$

[13] The proof may be omitted.

www.ebook3000.com



(a) (b)

Figure 11.43: Two graphs with hamiltonian circuits.

where $\deg_G(a)$ means degree of a in G. If we add edges to G, we eventually obtain a complete graph, which has a hamiltonian circuit. Thus, in the process of adding edges, we must eventually hit a graph H with the property that H has no hamiltonian circuit, but adding any edge to H gives us a graph with a hamiltonian circuit. We shall show that in H, there are nonadjacent x and y so that

$$\deg_H(x) + \deg_H(y) < n. \tag{11.11}$$

But $\deg_G(a) < \deg_H(a)$ for all a, so (11.11) implies (11.10).

Pick any nonadjacent x and y in H. Then H plus the edge $\{x, y\}$ has a hamiltonian circuit. Since H does not, this circuit must use the edge $\{x, y\}$. Hence, it can be written as

$$x, y, a_1, a_2, \ldots, a_{n-2}, x.$$

Now $V(H) = \{x, y, a_1, a_2, \ldots, a_{n-2}\}$. Moreover, we note that for $i > 1$,

$$\{y, a_i\} \in E(H) \Rightarrow \{x, a_{i-1}\} \notin E(H). \tag{11.12}$$

For if not, then

$$y, a_i, a_{i+1}, \ldots, a_{n-2}, x, a_{i-1}, a_{i-2}, \ldots, a_1, y$$

is a hamiltonian circuit in H, which is a contradiction. Now (11.12) and $\{x, y\} \notin E(H)$ imply (11.11). Q.E.D.

To illustrate Ore's Theorem, note that the graph (a) of Figure 11.43 has a hamiltonian circuit, for any two vertices have a sum of degrees at least 5. Note that Ore's Theorem does not give necessary conditions. For consider the circuit of length 5, Z_5. There is a hamiltonian circuit, but any two vertices have a sum of degrees equal to 4, which is less than n.

The next result, originally proved independently, follows immediately from Theorem 11.8.

Corollary 11.8.1 (Dirac [1952]) Suppose that G is a graph with $n \geq 3$ vertices and each vertex has degree at least $n/2$. Then G has a hamiltonian circuit.

To illustrate Corollary 11.8.1, note that the graph (b) of Figure 11.43 has a hamiltonian circuit because every vertex has degree 3 and there are six vertices.

Corollary 11.8.2 (Bondy and Chvátal [1976]) Suppose that G is a graph with $n \geq 3$ vertices and that x and y are nonadjacent vertices in G so that

$$\deg(x) + \deg(y) \geq n.$$

Then G has a hamiltonian circuit if and only if G plus the edge $\{x, y\}$ has a hamiltonian circuit.

Proof. If G has a hamiltonian circuit, then certainly G plus edge $\{x, y\}$ does. The converse follows from the proof of Theorem 11.8. Q.E.D.

Let us see what happens if we try to repeat the construction in Corollary 11.8.2. That is, we start with a graph $G = G_1$. We find a pair of nonadjacent vertices x_1 and y_1 in G_1 so that in G_1, the degrees of x_1 and y_1 sum to at least n. We let G_2 be G_1 plus edge $\{x_1, y_1\}$. We now find a pair of nonadjacent vertices x_2 and y_2 in G_2 so that in G_2, the degrees of x_2 and y_2 sum to at least n. We let G_3 be G_2 plus edge $\{x_2, y_2\}$. We continue this procedure until we obtain a graph $H = G_i$ with no nonadjacent x_i and y_i whose degrees in G_i sum to at least n. It is not hard to show (Exercise 17) that no matter in what order we perform this construction, we always obtain the same graph H. We call this graph H the *closure* of G, and denote it by $c(G)$. We illustrate the construction of $c(G)$ in Figure 11.44. The next result follows from Corollary 11.8.2.

Corollary 11.8.3 (Bondy and Chvátal [1976]) G has a hamiltonian circuit if and only if $c(G)$ has a hamiltonian circuit.

Note that in Figure 11.44, $c(G)$ is a complete graph in parts (a) and (c) but not in part (b). Parts (a) and (c) illustrate the following theorem. [Note that part (c) could not be handled by Ore's Theorem.]

Theorem 11.9 (Bondy and Chvátal [1976]) Suppose that G is a graph with at least three vertices. If $c(G)$ is complete, G has a hamiltonian circuit.

Proof. A complete graph with at least three vertices has a hamiltonian circuit. Q.E.D.

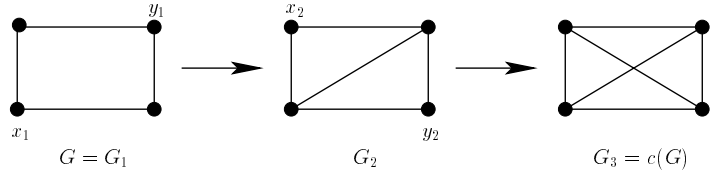
Note that Ore's Theorem is a corollary of Theorem 11.9.

We have given conditions sufficient for the existence of a hamiltonian circuit but not for a hamiltonian chain. For some conditions sufficient for the latter, see, for example, the text and exercises of Chartrand and Lesniak [1996].

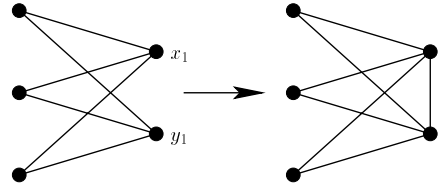
11.5.3 Sufficient Conditions for the Existence of a Hamiltonian Cycle in a Digraph

Ore's and Dirac's results (Theorem 11.8 and Corollary 11.8.1) have the following analogues for digraphs. Here, $\text{od}(u)$ is the outdegree of u and $\text{id}(u)$ is the indegree of u.

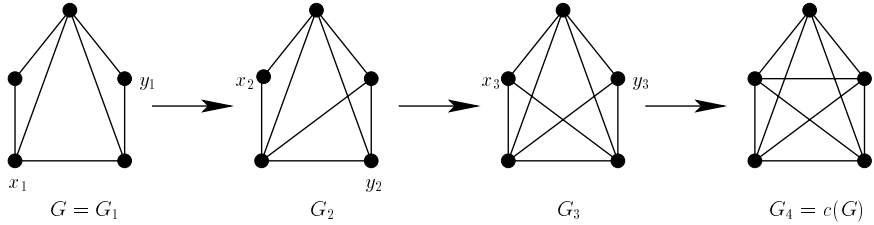
www.ebook3000.com



(a)



(b)



(c)

Figure 11.44: Three constructions of the closure.

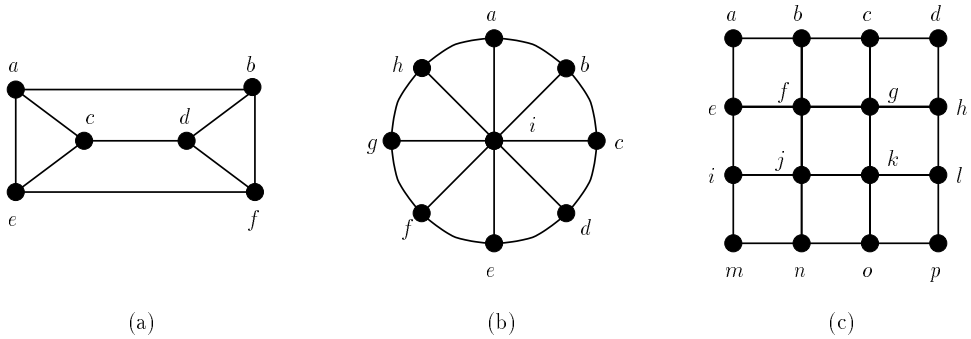


Figure 11.45: Graphs for exercises of Section 11.5.

Theorem 11.10 (Woodall [1972]) Suppose that D is a digraph with $n \geq 3$ vertices and whenever $x \neq y$ and there is no arc from x to y, then

$$\mathrm{od}(x) + \mathrm{id}(y) \geq n. \tag{11.13}$$

Then D has a hamiltonian cycle.

Theorem 11.11 (Ghouila-Houri [1960]) Suppose that D is a strongly connected digraph with n vertices and for every vertex x,

$$\mathrm{od}(x) + \mathrm{id}(x) \geq n. \tag{11.14}$$

Then D has a hamiltonian cycle.

Corollary 11.11.1 Suppose that D is a digraph with n vertices and for every vertex x,

$$\mathrm{od}(x) \geq \frac{n}{2} \quad \text{and} \quad \mathrm{id}(x) \geq \frac{n}{2}. \tag{11.15}$$

Then D has a hamiltonian cycle.

Proof. One shows that (11.15) implies that D is strongly connected. The proof is left to the reader (Exercise 18). Q.E.D.

We have given conditions for the existence of a hamiltonian cycle, but not for a hamiltonian path. For a summary of conditions sufficient for the latter, see, for example, Chartrand and Lesniak [1996].

EXERCISES FOR SECTION 11.5

1. In each graph of Figure 11.45, find a hamiltonian circuit.
2. In each graph of Figure 11.46, find a hamiltonian chain.

www.ebook3000.com

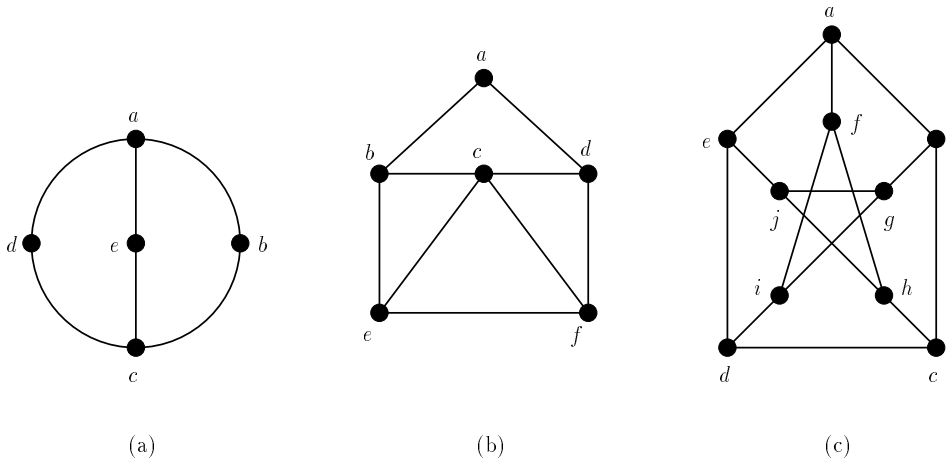


Figure 11.46: Graphs for exercises of Section 11.5.

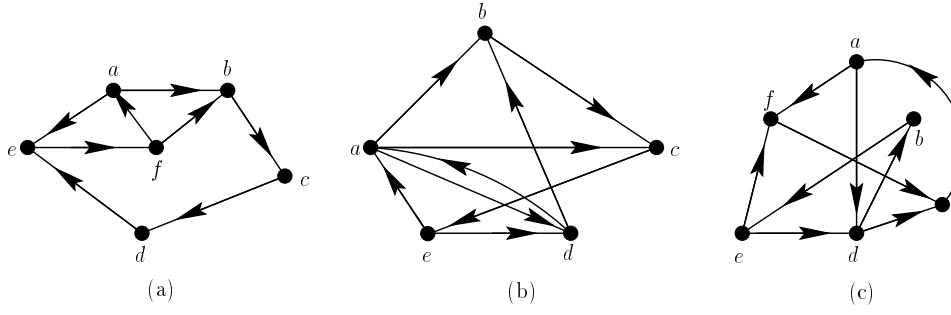


Figure 11.47: Digraphs for exercises of Section 11.5.

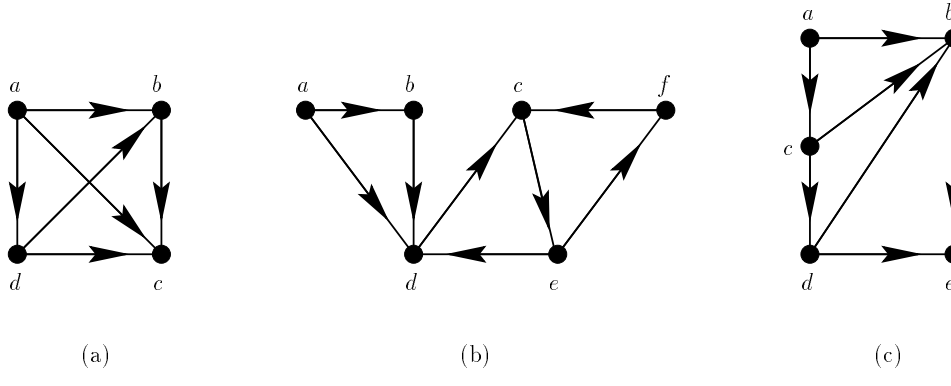


Figure 11.48: Digraphs for exercises of Section 11.5.

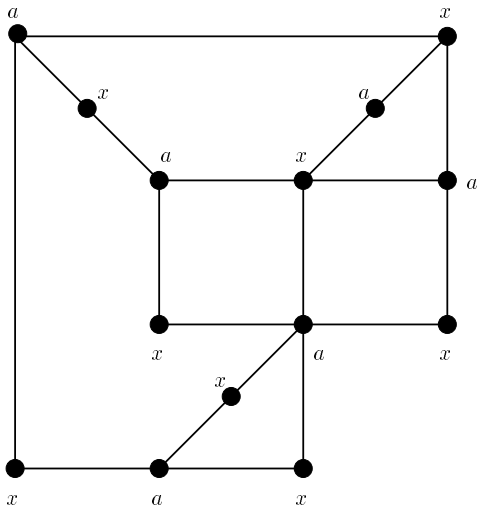


Figure 11.49: A graph with no hamiltonian circuit.

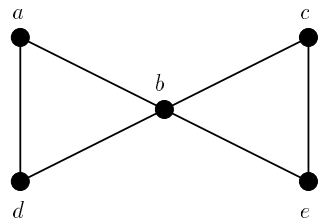


Figure 11.50: Graph for Exercise 6, Section 11.5.

3. In each digraph of Figure 11.47, find a hamiltonian cycle.
4. In each digraph of Figure 11.48, find a hamiltonian path.
5. Show that the graph of Figure 11.49 can have no hamiltonian circuit. (*Hint:* The a's and x's should tell you something.)
6. Can the graph of Figure 11.50 have a hamiltonian circuit? Why?
7. Give examples of graphs that:
   (a) Have both eulerian and hamiltonian circuits
   (b) Have a hamiltonian but not an eulerian circuit
   (c) Have an eulerian but not a hamiltonian circuit
   (d) Have neither an eulerian nor a hamiltonian circuit
8. Give an example of:
   (a) A graph with no hamiltonian chain
   (b) A digraph with no hamiltonian path
9. Suppose that G is a graph in which there are 10 vertices, a, b, c, d, e and x, y, u, v, w, and each of the first five vertices is joined to each of the last five.
   (a) Does Ore's Theorem (Theorem 11.8) imply that G has a hamiltonian circuit? Why?
   (b) Does Dirac's Theorem (Corollary 11.8.1) imply that G has a hamiltonian circuit? Why?
   (c) Does G have a hamiltonian circuit?

www.ebook3000.com

10. (a) Find the closure $c(G)$ of each graph of Figures 11.45 and 11.46.

    (b) For which graphs in Figures 11.45 and 11.46 does Theorem 11.9 imply that there is a hamiltonian circuit?

11. Give an example of a graph whose closure does not have a hamiltonian circuit.

12. Show that if G has a hamiltonian circuit, $c(G)$ is not necessarily complete.

13. For which digraphs of Figures 11.47 and 11.48 does Theorem 11.10 imply that there is a hamiltonian cycle?

14. For which digraphs of Figures 11.47 and 11.48 does Theorem 11.11 imply that there is a hamiltonian cycle?

15. A graph is *regular of degree* k if every vertex has the same degree, k. Show that G has a hamiltonian circuit if

    (a) G has 11 vertices and is regular of degree 6;

    (b) G has 13 vertices and is regular of degree 6.

16. Suppose that $n \geq 4$ and that of n people, any two of them together know all the remaining $n-2$. Show that the people can be seated around a circular table so that everyone is seated between two friends.

17. Suppose that by successively adding edges joining nonadjacent vertices whose degrees sum to at least n, we eventually obtain a graph H and then are unable to continue. Suppose that by doing the construction in a different order, we eventually obtain a graph K and then are unable to continue. Show that $H = K$. (*Hint:* If not, find the first edge added to G in the first construction which is not an edge of K.)

18. Show that Equation (11.15) implies that D is strongly connected.

19. Suppose that the hypotheses of Theorem 11.10 hold with n in Equation (11.13) replaced by $n-1$. Use Theorem 11.10 to show that D has a hamiltonian path.

20. Show that Corollary 11.8.1 is false if "each vertex has degree at least $n/2$" is replaced by "each vertex has degree at least $(n-1)/2$."

21. Suppose that the resetting costs as in Example 11.5 are given in the following matrix C, where c_{ij} is the cost of resetting the production facility from its configuration for processing item i to its configuration for processing item j.

$$C = \begin{pmatrix} 0 & 0 & 0 & 6 & 5 & 5 & 3 & 8 & 4 & 9 \\ 6 & 0 & 1 & 7 & 3 & 9 & 8 & 0 & 6 & 8 \\ 11 & 4 & 0 & 8 & 0 & 7 & 10 & 5 & 3 & 5 \\ 8 & 8 & 0 & 0 & 11 & 8 & 7 & 3 & 9 & 8 \\ 4 & 4 & 7 & 4 & 0 & 0 & 0 & 11 & 10 & 8 \\ 6 & 5 & 6 & 2 & 0 & 0 & 8 & 8 & 2 & 5 \\ 6 & 7 & 6 & 5 & 3 & 11 & 0 & 9 & 0 & 15 \\ 4 & 3 & 9 & 8 & 8 & 7 & 0 & 0 & 3 & 5 \\ 0 & 3 & 4 & 11 & 10 & 6 & 8 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 7 & 6 & 5 & 3 & 1 & 0 \end{pmatrix}.$$

Is there a production schedule in which no resetting costs are incurred?

22. (Pósa [1962]). Use Theorem 11.9 to show that if G is a graph of at least three vertices and if for every integer j with $1 \leq j < n/2$, the number of vertices of degree not exceeding j is less than j, then G has a hamiltonian circuit.

23. Prove or disprove the following variant on Ore's Theorem (Theorem 11.8). Suppose that G is a graph with $n \geq 3$ vertices and

$$\frac{\sum (\deg_G(x) + \deg_G(y))}{q} \geq n,$$

where the sum is taken over all nonadjacent pairs x and y and q equals the number of such x, y pairs; then G is hamiltonian.

11.6 APPLICATIONS OF HAMILTONIAN CHAINS AND PATHS

In this section we present several applications of the ideas of hamiltonian chains and paths. Sections 11.6.2 and 11.6.3 depend in small ways on Section 11.6.1. Otherwise, these sections are independent and can be read in any order. In particular, a quick reading could include just a glance at Section 11.6.1, followed by Section 11.6.3, or it could consist of just Section 11.6.4. Section 11.6.5 should be read as a natural follow-up to Section 11.4.4.

11.6.1 Tournaments

Let (V, A) be a digraph and assume that for all $u \neq v$ in V, (u, v) is in A or (v, u) is in A, but not both. Such a digraph is called a *tournament*. Tournaments arise in many different situations. There are the obvious ones, the round-robin competitions in tennis, basketball, and so on.[14] In a round-robin competition, every pair of players (pair of teams) competes, and one and only one member of each pair beats the other. (We assume that no ties are allowed.) Tournaments also arise from *pair comparison experiments*, where a subject or a decisionmaker is presented with each pair of alternatives from a set and is asked to say which of the two he or she prefers, which is more important, which is more qualified, and so on. (The reader might wish to consider the discussion of preference in Chapter 4, in Example 4.1 and elsewhere, for comparison.) Tournaments also arise in nature, where certain individuals in a given species develop dominance over others of the same species. In such situations, for every pair of animals of the species, one and only one is dominant over the other. The dominance relation defines what is called a *pecking order* among the individuals concerned.

Sometimes we want to *rank* the players of a tournament. This might be true if we are giving out second prize, third prize, and so on. It might also be true, for example, if the "players" are alternative candidates for a job and the tournament represents preferences among candidates. Then our first-choice job candidate might

[14] The reader should distinguish a round-robin competition from the elimination-type competition, which is more common.

www.ebook3000.com

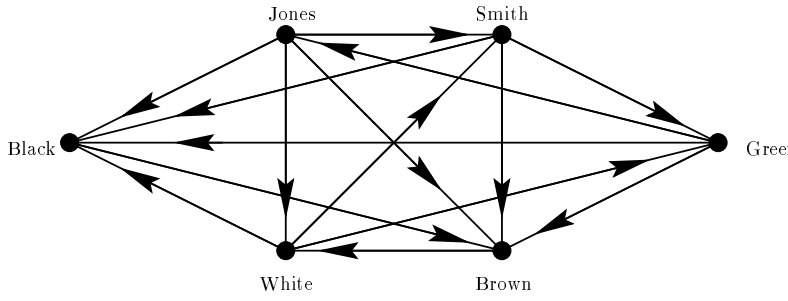


Figure 11.51: A round-robin Ping-Pong tournament.

not accept and we might want to have a second choice, third choice, and so on, chosen ahead of time. One way to find a ranking of the players is to observe that every tournament has a hamiltonian path. This result, which we shall prove shortly, implies that we can label the n players as $u_1, u_2, \ldots, u_n$ in such a way that u_1 beats u_2, u_2 beats u_3, $\ldots$, and u_{n-1} beats u_n. Such a labeling gives us a ranking of the players: u_1 is ranked first, u_2 second, and so on. To illustrate, consider the tournament of Figure 11.51. Here, a hamiltonian path and hence a ranking is given by: Jones, Smith, Green, Brown, White, Black. Another way to find a ranking of the players is to use the score sequence defined in Exercise 20 (see also Exercises 23 and 28).

Theorem 11.12 (Rédei [1934]) Every tournament (V, A) has a hamiltonian path.

Proof.[15] We proceed by induction on the number n of vertices. The result is trivial if $n = 2$. Assuming the result for tournaments of n vertices, let us consider a tournament (V, A) with $n + 1$ vertices. Let u be an arbitrary vertex and consider the subgraph generated by $V - \{u\}$. It is easy to verify that this subgraph is still a tournament. Hence, by inductive assumption, it has a hamiltonian path $u_1, u_2, \ldots, u_n$. If there is an arc from u to u_1, then $u, u_1, u_2, \ldots, u_n$ is a hamiltonian path of (V, A). If there is no arc from u to u_1, then, since (V, A) is a tournament, there is an arc from u_1 to u. Let i be the largest integer such that there is an arc from u_i to u. If i is n, there is an arc from u_n to u, and we conclude that $u_1, u_2, \ldots, u_n, u$ is a hamiltonian path of (V, A). If $i < n$, there is an arc from u_i to u, and moreover, by definition of i, there is no arc from u_{i+1} to u. Since (V, A) is a tournament, there is an arc from u to u_{i+1}. Thus, $u_1, u_2, \ldots, u_i, u, u_{i+1}, \ldots, u_n$ is a hamiltonian path of (V, A). Q.E.D.

If $u_1, u_2, \ldots, u_n$ is a hamiltonian path in a tournament (V, A), then as we have already observed, we can use it to define a ranking of the players. Unfortunately, there can be other hamiltonian paths in the tournament. In our example, Smith, Brown, White, Green, Jones, Black is another. In this situation, then, there are

[15] The proof may be omitted.

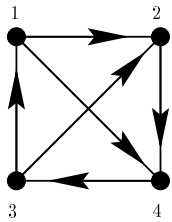


Figure 11.52: For each vertex a and rank r, there is some hamiltonian path for which a has rank r.

many possible rankings of the players. The situation can get even worse. In the tournament of Figure 11.52, in fact, for each vertex a and each possible rank r, there is some hamiltonian path for which a has rank r.

In general, given a set of possible rankings, we might want to try to choose a "consensus" ranking. The problem of finding a consensus among alternative possible rankings is a rather difficult one, that deserves a longer discussion than we can give it here (see Roberts [1976, Ch. 7]).

We next ask whether there are any circumstances where the problem of having many different rankings does not occur. That is, are there circumstances when a tournament has a unique hamiltonian path? The answer is given by saying that a tournament is *transitive* if whenever (u, v) is an arc and (v, w) is an arc, then (u, w) is an arc.[16] The tournament of Figure 11.51 is not transitive, since, for example, Jones beats Smith and Smith beats Green, but Jones does not beat Green.

Theorem 11.13 A tournament has a unique hamiltonian path if and only if the tournament is transitive.

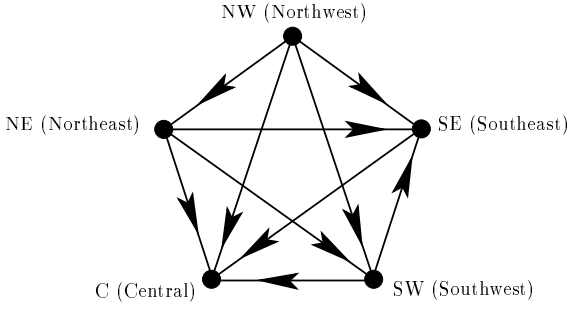
Proof. See Exercise 27. Q.E.D.

At this point, let us consider applications of Theorem 11.13 to a decisionmaker's preferences among alternatives, or to his or her ratings of relative importance of alternatives, and so on. The results are illustrated by pair comparison data for preference: for example, that shown in Figure 11.53. Here, transitivity holds, so there is a unique hamiltonian path. The path is Northwest, Northeast, Southwest, Southeast, Central.

In studying preference, it is often reasonable to assume (or demand) that the decisionmaker's preferences define a transitive tournament, that is, that if he or she prefers u to v and v to w, then he or she prefers u to w. This turns out to be equivalent to assuming that the decisionmaker can uniquely rank the alternatives among which he or she is expressing preferences. The reader might wish to compare the discussion of preference in Example 4.1 and of various order relations in Section 4.2. In the language of that section, a tournament is a strict linear order (see Exercise 6).

[16] Transitivity played an important role in Chapter 4. Sometimes, transitivity is defined differently for digraphs, requiring that the existence of arcs (u, v) and (v, w) implies the existence of arc (u, w) only if $u \neq w$. The condition $u \neq w$ holds automatically for tournaments if (u, v) and (v, w) are arcs. Why?

www.ebook3000.com



	NW	SW	C	SE	NE
NW	0	1	1	1	1
SW	0	0	1	1	0
C	0	0	0	0	0
SE	0	0	1	0	0
NE	0	1	1	1	0

Figure 11.53: Results of a pair comparison experiment for preference among geographical areas. The x, y entry in the matrix is 1 iff x is preferred to y. In the corresponding digraph, an arc from x to y indicates that x is preferred to y.

Performing a pair comparison experiment to elicit preferences can be a tedious job. If there are n alternatives, $\binom{n}{2} = n(n-1)/2$ comparisons are required, and this can get to be a large number even for moderate n. However, if we believe that a subject is transitive, we know that a pair comparison experiment amounts to a transitive tournament and hence to a unique ranking of the alternatives. Thus, we are trying to order a set of n items. How many comparisons are really required to recover this ranking and hence the original tournament? This is the problem of sorting that we discussed in Section 3.6. There, we pointed out that good sorting algorithms can take on the order of $n \log_2 n$ steps, which is a smaller number than $n(n-1)/2$.

11.6.2 Topological Sorting

It turns out that a tournament is transitive if and only if it is *acyclic*, that is, has no cycles (Exercise 26). Finding the unique hamiltonian path in a transitive or acyclic tournament is a special case of the following more general problem. Suppose that we are given a digraph D of n vertices. Label the vertices of D with the integers $1, 2, \ldots, n$ so that every arc of D leads from a vertex with a smaller label to a vertex with a larger label. Such a labeling is called a *topological order* for D, and the process of finding such an order is called *topological sorting*.

Theorem 11.14 A digraph D has a topological order if and only if D is acyclic.

Proof. Clearly, if there is a topological order $1, 2, \ldots, n$, then D could not have a cycle. Conversely, suppose that D is acyclic. By Exercise 31, there is a vertex x_1 with no incoming arcs. Label x_1 with the label 1, and delete it from D. Now the resulting digraph is still acyclic and hence has a vertex x_2 with no incoming arcs. Label x_2 with label 2; and so on. This clearly gives rise to a topological order. Q.E.D.

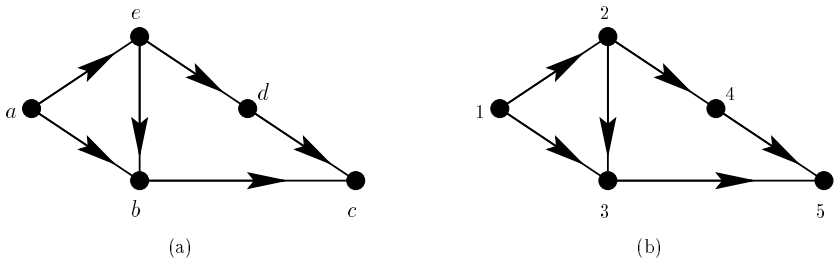


Figure 11.54: Part (b) shows a topological order for the digraph of part (a).

We illustrate the construction in the proof of Theorem 11.14 by finding a topological order in the digraph D shown in Figure 11.54(a). We first choose vertex a. After eliminating a, we choose vertex e. After eliminating e, we have a choice of either b or d; say, for concreteness, that we choose b. Then after eliminating b, we choose d. Finally, we are left with c. This gives the labeling shown in Figure 11.54(b). One of the problems with the procedure described is that at each stage, we must search through the entire remaining digraph for the next vertex to choose. For a description of a more efficient algorithm, based on the depth-first search procedure described in Section 11.1, see Cormen, Leiserson, and Rivest [1999], Golumbic [1980], or Reingold, Nievergelt, and Deo [1977].

Topological sorting has a variety of applications. For instance, it arises in the analysis of *activity networks* in project planning (Deo [1974]). A large project is divided into individual tasks called *activities*. Some activities must be completed before others can be started—for example, an item must be sanded before it can be painted. Build a digraph D as follows. The vertices are the activities, and there is an arc from activity x to activity y if x must precede y. We seek to find an order in which to perform the activities so that if there is an arc from x to y in D, then x comes before y. This requires a topological sorting of D. It can be done if and only if D has no cycles. A similar situation arises, for example, if we wish to arrange words in a glossary and guarantee that no word is used before it is defined. This, too, requires a topological sorting.

11.6.3 Scheduling Problems in Operations Research[17]

Many scheduling problems in operations research, for instance those involving activity networks discussed in Section 11.6.2, involve finding an order in which to perform a certain number of operations. We often seek an optimal order. Sometimes such problems can be solved by finding hamiltonian paths. We have already seen an example of this in Example 11.5. Here we present another example. Related problems are discussed in Section 13.2.3.

Suppose that a printer has n different books to publish. He has two machines, a printing machine and a binding machine. A book must be printed before it can be

[17] This subsection is based on Berge [1962], Johnson [1954], and Liu [1972].

www.ebook3000.com

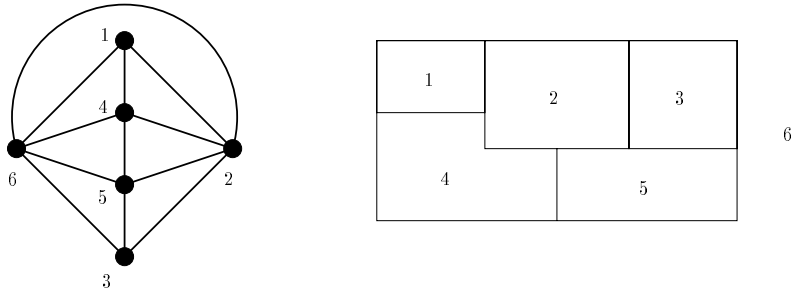


Figure 11.55: A relationship graph G and a corresponding layout plan.

bound. The binding machine operator earns much more money than the printing machine operator, and must be paid from the time the binding machine is first started until all the books have been bound. In what order should the books be printed so that the total pay of the binding machine operator is minimized?

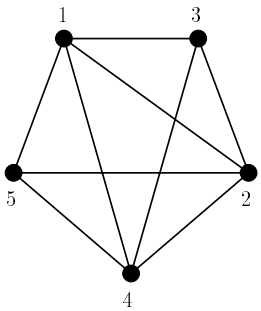
Let p_k be the time required to print the kth book and b_k the time required to bind the kth book. Let us make the special assumption that for all i and j, either $p_i \leq b_j$ or $p_j \leq b_i$. Note that it is now possible to find an ordering of books so that if books are printed and bound in that order, the binding machine will be kept busy without idle time after the first book is printed. This clearly will minimize the pay of the binding machine operator. To find the desired ordering, draw a digraph with an arc from i to j if and only if $b_i \geq p_j$. Then this digraph contains a tournament and so, by Theorem 11.12, has a hamiltonian path. This path provides the desired ordering. More general treatment of this problem, without the special assumption, can be found in Johnson [1954] (see also Exercise 38).

11.6.4 Facilities Design[18]

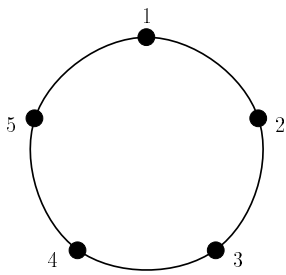
Graph theory is finding a variety of applications in the design of such physical facilities as manufacturing plants, hospitals, schools, golf courses, and so on. In such design problems, we have a number of areas that need to be located and it is desired that certain of them be next to each other. We draw a graph G, the *relationship graph*, whose vertices are the areas in question and that has an edge between two areas if they should be next to each other. If G is planar, it comes from a map (see Example 3.19), and the corresponding map represents a layout plan where two areas (countries) joined by an edge in G in fact share a common wall (boundary). Figure 11.55 shows a planar graph and the corresponding facilities layout.

If the relationship graph G is not planar, we seek to eliminate some requirements, i.e., eliminate some edges of G, to find a spanning subgraph G' of G that is planar.

[18] This subsection is based on Chachra, Ghare, and Moore [1979].



G



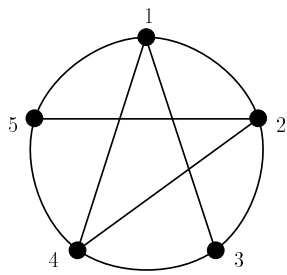


Figure 11.56: A relationship graph G, the vertices of a hamiltonian circuit arranged around a circle, and the remaining edges of G represented as chords of the circle.

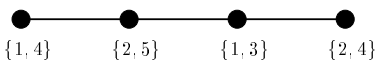


Figure 11.57: Graph H obtained from Figure 11.56.

Then G' can be used to build the layout plan. One method for finding G' that is used in facilities planning is the following (due to Demourcron, Malgrance, and Pertuiset [1964]). Determine if G has a hamiltonian circuit. If so, find such a circuit $C = u_1, u_2, \ldots, u_n$. Locate the vertices u_i around a circle in this order. For instance, consider graph G of Figure 11.56. Then $C = 1, 2, 3, 4, 5$ is a hamiltonian circuit and it is represented around a circle in Figure 11.56. Build a new graph H as follows. The vertices of H are the edges in G which are not edges of C. Two edges of H are adjacent if and only if when the corresponding chords are drawn in the circle determined by C, these chords intersect. See Figure 11.56 for the chords in our example, and see Figure 11.57 for the graph H. Suppose that H is 2-colorable, say using the colors red and blue. Then G is planar. To see this, simply note that all red chords can be drawn in the interior of the circle determined by C and all blue chords in the exterior, with no edges crossing. For instance, in graph H of Figure 11.57, we can color vertices $\{1,3\}$ and $\{1,4\}$ red and vertices $\{2,4\}$ and $\{2,5\}$ blue. Then we obtain the planar drawing of G shown in Figure 11.58. If H is not 2-colorable, we seek a maximal subgraph K of H that is 2-colorable. The edges of K can be added to C to get a planar graph G' which is a spanning subgraph of G. G' is used for finding a layout plan. For instance, suppose that G is obtained from G of Figure 11.56 by adding edge $\{3,5\}$. Then, using the same hamiltonian circuit C, we see that H is as shown in Figure 11.59. This H is not 2-colorable. We have to eliminate some vertex to get K, for instance vertex $\{3,5\}$. The resulting graph G' is the graph G of Figure 11.56, which is planar.

www.ebook3000.com

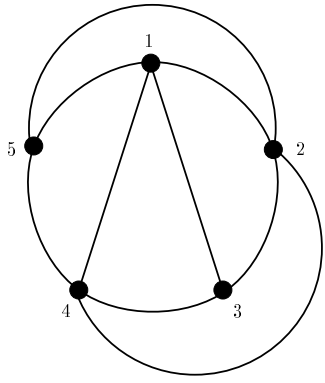


Figure 11.58: Planar drawing of graph G of Figure 11.56.

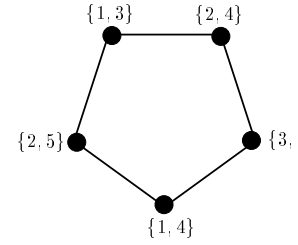


Figure 11.59: Graph H obtained if edge $\{3, 5\}$ is added to G of Figure 11.56.

11.6.5 Sequencing by Hybridization[19]

In Example 11.2 we described the problem of determining a DNA string S from the list L of all k-length substrings, the problem of sequencing by hybridization. An alternative approach to this problem is to build a digraph $E(L)$ whose vertices are strings in L and which has an arc from $a_1a_2 \cdots a_k$ in L to $b_1b_2 \cdots b_k$ in L if $a_2 = b_1$, $a_3 = b_2$, ..., $a_k = b_{k-1}$. The digraph $E(L)$ corresponding to L of (11.2) is shown in Figure 11.60. Assuming, as in Example 11.2, that no k-length substring appears more than once in L, it is easy to see that there is a one-to-one correspondence between paths that visit each vertex of $E(L)$ exactly once and DNA strings whose k-length substrings correspond to strings in L. We use the first letter of each vertex in such a path and end with the last $k-1$ letters of the last vertex in the path. Thus, the DNA strings arising from L correspond to hamiltonian paths in $E(L)$. In Figure 11.60, the hamiltonian path ACA, CAC, ACG, CGC, GCA, CAA, AAC, ACT, CTT, TTA, TAA, AAA corresponds to the DNA string ACACGCAACTTAAA. However, there are other hamiltonian paths. For instance, the path ACG, CGC, GCA, CAA, AAA, AAC, ACA, CAC, ACT, CTT, TTA, TAA. Since the problem of finding hamiltonian paths is NP-complete, this method of finding an unknown DNA fragment is not as efficient as the eulerian path method described in Example 11.2.

EXERCISES FOR SECTION 11.6

1. In each of the following situations, determine if the digraph corresponding to the binary relation is a tournament.

[19] This subsection follows Pevzner [2000].

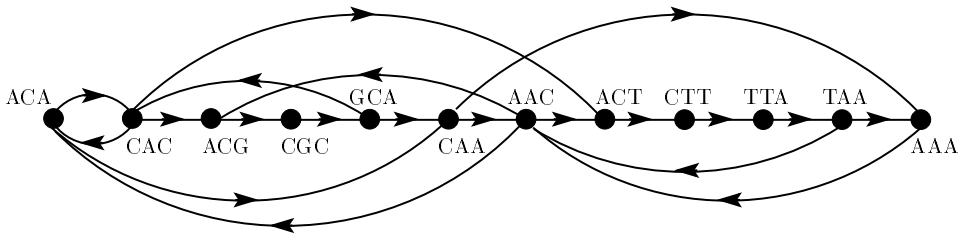


Figure 11.60: Digraph $E(L)$ obtained from list L of (11.2).

Table 11.3: Results of a Pair Comparison Experiment for Preference Among Cities as a Place to Live (Entry i, j is 1 iff i is Preferred to j)

	New York	Boston	San Francisco	Los Angeles	Houston
New York	0	0	0	0	1
Boston	1	0	0	1	1
San Francisco	1	1	0	1	1
Los Angeles	1	0	0	0	0
Houston	0	0	0	1	0

(a) The binary relation (X, R), where $X = \{1, 2, 3, 4\}$, R is defined by Equation (4.1).

(b) The binary relation (X, S), where $X = \{1, 2, 3, 4\}$, S is defined by Equation (4.2).

2. For each binary relation of Exercise 1, Section 4.1, determine if the corresponding digraph is a tournament.
3. For each binary relation of Exercise 1, determine if the digraph corresponding to the binary relation has a topological order and find such an order if there is one.
4. For each binary relation of Exercise 1, Section 4.1, determine if the digraph corresponding to the binary relation has a topological order.
5. For each property of a binary relation defined in Table 4.2, determine if it holds for a tournament (i) always, (ii) sometimes, or (iii) never.
6. Show that a transitive tournament is a strict linear order as defined in Section 4.2.
7. In each tournament of Figure 11.61, find all hamiltonian paths.
8. For the tournament defined by the preference data of Table 11.3, find all hamiltonian paths.
9. Which of the tournaments with four or fewer vertices is transitive?
10. Draw a digraph of a transitive tournament on five players.
11. Find a topological order for each of the digraphs of Figure 11.62.
12. In the book printing problem of Section 11.6.3, suppose that $b_1 = 3, b_2 = 5, b_3 = 8, p_1 = 6, p_2 = 2, p_3 = 9$. Find an optimal order in which to produce books.
13. Repeat Exercise 12 if $b_1 = 10, b_2 = 7, b_3 = 5, p_1 = 11, p_2 = 4, p_3 = 8$.

www.ebook3000.com

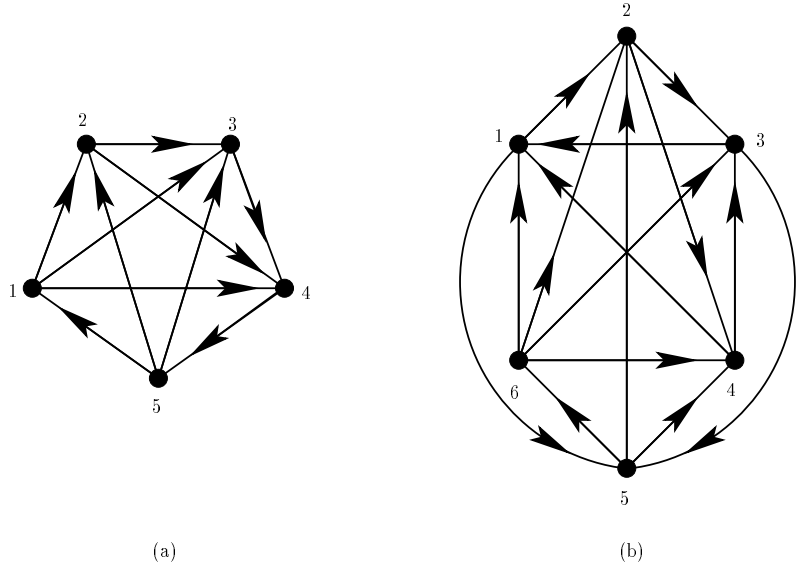


Figure 11.61: Tournaments for exercises of Section 11.6.

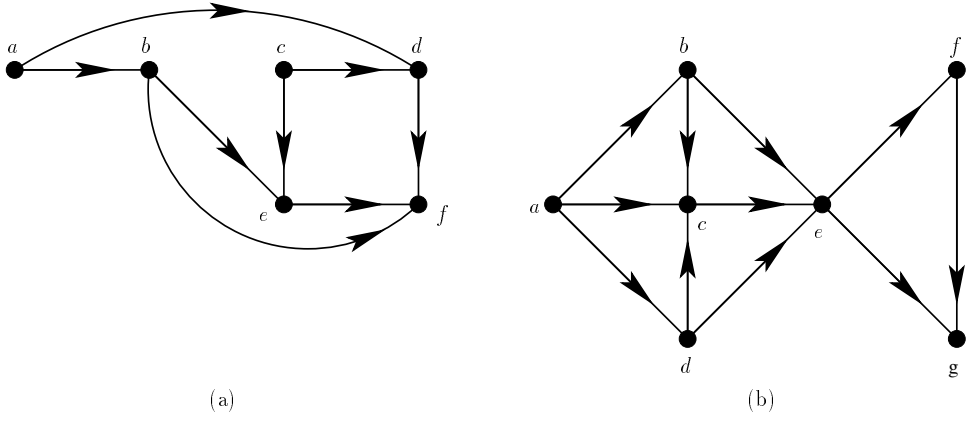


Figure 11.62: Digraphs for Exercise 11 of Section 11.6.

14. Design a layout plan for the planar graphs (a) and (b) of Figure 11.46.
15. If G is graph (a) of Figure 11.45, find a hamiltonian circuit C and construct H as in Section 11.6.4. Use H to determine a planar diagram for G, if one exists. Otherwise, find a planar G'.
16. Repeat Exercise 15 if G is graph (b) of Figure 11.45.
17. Repeat Exercise 15 if G is graph (b) of Figure 11.43.
18. (a) Draw digraph $E(L)$ of Section 11.6.5 for the list $L = \{$CCGA, CGCG, CGTG, CGAA, GCCG, GCGT, GTGC, GAAC, TGCC, AACC, ACCC$\}$.
    (b) Find the number of hamiltonian paths in $E(L)$.
    (c) Find the compatible DNA string(s) of L.
19. (a) Draw digraph $E(L)$ of Section 11.6.5 for the list $L = \{$AA, AC, AG, AT, CA, CT, GA, GT, TA, TC, TG, TT$\}$.
    (b) Is $S =$ ACTTCATGAGTAA compatible with L?
    (c) Is $S =$ AATGAGTACTTCA compatible with L?
    (d) Is $S =$ CTAGTACATGATA compatible with L?
20. In a tournament, the *score* of vertex u is the outdegree of u. (It is the number of players u beats.) If the vertices are labeled $1, 2, \ldots, n$, with $s(1) \leq s(2) \leq \cdots \leq s(n)$, the sequence $(s(1), s(2), \ldots, s(n))$ is called the *score sequence* of the tournament. Exercises 20–24 and 28 investigate the score sequence. Find the score sequence of the tournaments of:

    (a) Figure 11.51 (b) Figure 11.53

21. Show that if $(s(1), s(2), \ldots, s(n))$ is the score sequence of a tournament, then $\sum_{i=1}^{n} s(i) = \binom{n}{2}$.
22. Could each of the following be the score sequence of a tournament? Why?

    (a) $(1, 1, 2, 3)$ (b) $(0, 0, 0, 2, 7)$ (c) $(0, 1, 1, 4, 4)$
    (d) $(0, 0, 3, 3)$ (e) $(1, 2, 2, 3, 3, 4)$

23. Show that in a tournament, the ranking of players using the score sequence can be different from the ranking of players using a hamiltonian path.
24. Draw the digraph of a tournament with score sequence:

    (a) $(0, 1, 2, 3)$ (b) $(2, 2, 2, 2, 2)$ (c) $(1, 1, 1, 4, 4, 4)$

25. Show that a tournament is transitive if and only if it has no cycles of length 3.
26. Show that a tournament is transitive if and only if it is acyclic.
27. Prove Theorem 11.13 as follows.
    (a) Show that if D is transitive, it has a unique hamiltonian path, by observing that if there are two such paths, there must be u and v with u following v in one, but v following u in the other.
    (b) Prove the converse by observing that in the unique hamiltonian path u_1, u_2, $\ldots$, u_n, we have $(u_i, u_j) \in A$ iff $i < j$.

www.ebook3000.com

28. (a) Use the result of Exercise 27(b) to find the score sequence of a transitive tournament of n vertices.

    (b) Show that for a transitive tournament, the ranking obtained by the score sequence is the same as the ranking obtained by the unique hamiltonian path.

29. Suppose that the vertices of a tournament can be listed as $u_1, u_2, \ldots, u_n$ so that the score $s(u_i) = n - i$. Does it follow that the tournament has a hamiltonian path? (Give proof or counterexample.)

30. Use the result of Exercise 28(a) to determine if the preference data of Table 11.3 are transitive.

31. Show that every acyclic digraph has a vertex with no incoming arcs.

32. In a tournament, a triple $\{x, y, z\}$ of vertices is called *transitive* if the subgraph generated by x, y, and z is transitive, equivalently if one of these three vertices beats the other two. This one vertex is called the *transmitter* of the transitive triple.

    (a) How would you find the number of transitive triples of which a given vertex u is the transmitter?

    (b) Show that in a tournament, if $s(x)$ is the score of x, there are $\sum_x \binom{s(x)}{2}$ transitive triples.

    (c) Show that every tournament of at least four vertices has to have a transitive triple.

33. If you have not already done so, do Exercise 40, Section 3.6.

34. (Harary, Norman, and Cartwright [1965]) If D has a level assignment (Exercise 40, Section 3.6) and r is the length of the longest simple path of D, show that $r + 1$ is the smallest number of distinct levels (values L_i) in a level assignment for D.

35. If you have not already done so, do Exercise 41, Section 3.6.

36. In a transitive tournament, if vertex u has maximum score, u beats every other player.

    (a) Show that in an arbitrary tournament, if u has maximum score, then for every other player v, either u beats v or u beats a player who beats v. (This result was discovered by Landau [1955] during a study of pecking orders among chickens. For generalizations of this result, see Maurer [1980].)

    (b) Show that the necessary condition of part (a) for u to be a winner is not sufficient.

37. (Camion [1959] and Foulkes [1960]) Prove that every strongly connected tournament has a hamiltonian cycle. (*Hint:* Show that there is a cycle of length k for $k = 3, 4, \ldots, n$, where n is the number of vertices.)

38. An alternative criterion for optimality in the book production problem of Section 11.6.3 is to finish printing and binding all the books in as short a time as possible. Show that even under the special assumption of Section 11.6.3, there can be an optimal solution that does not correspond to a hamiltonian path in the digraph constructed in that section.

REFERENCES FOR CHAPTER 11

Abello, J., Buchsbaum, A., and Westbrook, J., "A Functional Approach to External Graph Algorithms," *Algorithmica, 32* (2002), 437–458.

Abello, J., Pardalos, P. M., and Resende, M. G. C., "On Maximum Clique Problems in Very Large Graphs," in J. Abello and J. Vitter (eds.), *External Memory Algorithms*, DIMACS Series, Vol. 50, American Mathematical Society, Providence, RI, 1999, 119–130.

Abello, J., and Vitter, J. (eds.), *External Memory Algorithms*, DIMACS Series, Vol. 50, American Mathematical Society, Providence, RI, 1999.

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

Aiello, W., Chung, F. R. K., and Lu, L., "A Random Graph Model for Massive Graphs," *Proc. 32nd Annual ACM Symposium on Theory of Computing*, (2000), 171–180.

Atallah, M. J., "Parallel Strong Orientation of an Undirected Graph," *Inform. Process. Lett., 18* (1984), 37–39.

Baase, S., *Computer Algorithms*, 2nd ed., Addison-Wesley Longman, Reading, MA, 1992.

Beltrami, E. J., *Models for Public Systems Analysis*, Academic Press, New York, 1977.

Berge, C., *The Theory of Graphs and Its Applications*, Wiley, New York, 1962.

Bermond, J.-C., Bond, J., Martin, C., Pekeč, A., and Roberts, F. S., "Optimal Orientations of Annular Networks," *J. Interconn. Networks, 1* (2000), 21–46.

Boesch, F., and Tindell, R., "Robbins' Theorem for Mixed Graphs," *Amer. Math. Monthly, 87* (1980), 716–719.

Bondy, J. A., and Chvátal, V., "A Method in Graph Theory," *Discrete Math., 15* (1976), 111–136.

Camion, P., "Chemins et Circuits Hamiltoniens des Graphes Complets," *C. R. Acad. Sci. Paris, 249* (1959), 2151–2152.

Chachra, V., Ghare, P. M., and Moore, J. M., *Applications of Graph Theory Algorithms*, Elsevier North Holland, New York, 1979.

Chartrand, G., and Lesniak, L., *Graphs and Digraphs*, 3rd ed., CRC Press, Boca Raton, 1996.

Christofides, N., *Graph Theory: An Algorithmic Approach*, Academic Press, New York, 1975.

Chung, F. R. K., Garey, M. R., and Tarjan, R. E., "Strongly Connected Orientations of Mixed Multigraphs," *Networks, 15* (1985), 477–484.

Chvátal, V., and Thomassen, C., "Distances in Orientations of Graphs," *J. Comb. Theory B, 24* (1978), 61–75.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L., *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1999.

de Bruijn, N. G., "A Combinatorial Problem," *Nedl. Akad. Wet., Proc., 49* (1946), 758–764; *Indag. Math., 8* (1946), 461–467.

Demourcron, G., Malgrance, V., and Pertuiset, R., "Graphes Planaires: Reconnaissance et Construction de Representations Planaires Topologiques," *Recherche Operationelle, 30* (1964), 33.

Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1974.

Dirac, G. A., "Some Theorems on Abstract Graphs," *Proc. Lond. Math. Soc., 2* (1952),

www.ebook3000.com

69–81.

Drmanac, R., Labat, I., Brukner, I., and Crkvenjakov, R., "Sequencing of Megabase Plus DNA by Hybridization: Theory of the Method," *Genomics, 4* (1989), 114–128.

Dror, M., *Arc Routing: Theory, Solutions and Applications*, Kluwer, Boston, 2000.

Euler, L., "Solutio Problematis ad Geometriam Situs Pertinentis," *Comment. Acad. Sci. I. Petropolitanae, 8* (1736), 128–140. [Reprinted in *Opera Omnia*, Series 1–7 (1766), 1–10.]

Even, S., *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.

Foulkes, J. D., "Directed Graphs and Assembly Schedules," *Proc. Symp. Appl. Math, Amer. Math. Soc., 10* (1960), 281–289.

Ghouila-Houri, A., "Une Condition Suffisante D'existence d'un Circuit Hamiltonien," *C. R. Acad. Sci. Paris, 156* (1960), 495–497.

Golomb, S. W., *Shift Register Sequences*, Aegean Park Press, Laguna Hills, CA, 1982.

Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

Good, I. J., "Normal Recurring Decimals," *J. Lond. Math. Soc., 21* (1946), 167–169.

Gusfield, D., *Algorithms on Strings, Trees and Sequences; Computer Science and Computational Biology*, Cambridge University Press, New York, 1997.

Gutin, Z. G., "Minimizing and Maximizing the Diameter in Orientations of Graphs," *Graphs Combin., 10* (1994), 225–230.

Han, X., "On the Optimal Strongly Connected Orientations of City Street Graphs: Over Twenty East-West Avenues or North-South Streets," mimeographed, Shanghai Computer Software Laboratory, Shanghai, China, 1989.

Harary, F., Norman, R. Z., and Cartwright, D., *Structural Models: An Introduction to the Theory of Directed Graphs*, Wiley, New York, 1965.

Harary, F., and Palmer, E. M., *Graphical Enumeration*, Academic Press, New York, 1973.

Helly, W., *Urban Systems Models*, Academic Press, New York, 1975.

Hopcroft, J. E., and Tarjan, R. E., "Algorithm 447: Efficient Algorithms for Graph Manipulation," *Commun. ACM, 16* (1973), 372–378.

Hutchinson, G., "Evaluation of Polymer Sequence Fragment Data Using Graph Theory," *Bull. Math. Biophys., 31* (1969), 541–562.

Hutchinson, J. P., and Wilf, H. S., "On Eulerian Circuits and Words with Prescribed Adjacency Patterns," *J. Comb. Theory A, 18* (1975), 80–87.

Johnson, E. L., "Chinese Postman and Euler Tour Problems in Bi-Directed Graphs," in M. Dror (ed.), *Arc Routing: Theory, Solutions and Applications*, Kluwer, Boston, 2000, 171–196.

Johnson, S. M., "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Naval Res. Logist. Quart., 1* (1954), 61–68.

Karp, R. M., and Ramachandran, V., "Parallel Algorithms for Shared-Memory Machines," in J. Van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Vol. A, Elsevier, Amsterdam, 1990, 869–941.

Koh, K. M., and Tan, B. P., "The Diameter of an Orientation of a Complete Multipartite Graph," *Discr. Math., 149* (1996), 331–356. (a) [See also addendum *Discr. Math., 173* (1997), 297–298.]

Koh, K. M., and Tan, B. P., "The Minimum Diameter of Orientations of a Complete Multipartite Graph," *Graphs Combin., 12* (1996), 333–339. (b)

Kwan, M. K., "Graphic Programming Using Odd or Even Points," *Chin. Math., 1*

(1962), 273–277.

Landau, H. G., "On Dominance Relations and the Structure of Animal Societies III. The Condition for a Score Sequence," *Bull. Math. Biophys., 15* (1955), 143–148.

Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

Liebling, T. M., *Graphentheorie in Planungs- und Tourenproblemen*, Lecture Notes in Operations Research and Mathematical Systems No. 21, Springer-Verlag, New York, 1970.

Liu, C. L., *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York, 1968.

Liu, C. L., *Topics in Combinatorial Mathematics*, Mathematical Association of America, Washington, DC, 1972.

Maurer, S. B., "The King Chicken Theorems," *Math. Magazine, 53* (1980), 67–80.

McCanna, J. E., "Orientations of the n-Cube with Minimum Diameter," *Disc. Math., 68* (1988), 309–310.

Minieka, E., *Optimization Algorithms for Networks and Graphs*, Dekker, New York, 1978.

Ore, O., "Note on Hamilton Circuits," *Amer. Math. Monthly, 67* (1960), 55.

Pevzner, P. A., "1-Tuple DNA Sequencing: Computer Analysis," *J. Biomol. Structure Dynamics, 7* (1989), 63–73.

Pevzner, P. A., *Computational Molecular Biology*, MIT Press, Cambridge, MA, 2000.

Pevzner, P. A., and Lipshutz, R., "Towards DNA Sequencing Chips," in *Proc. 19th Symp. Math. Found. Comp. Sci.*, LNCS 684, Springer, New York, 1994, 143–158.

Plesnik, J., "Remarks on Diameters of Orientations of Graphs," *Math. Univ. Comeniance, 46/47* (1986), 225–236.

Pósa, L., "A Theorem Concerning Hamiltonian Lines," *Magyar Tud. Akad. Mat. Kutató Int. Közl., 7* (1962), 225–226.

Rédei, L., "Ein kombinatorischer Satz," *Acta Litt. Sci. (Sect. Sci. Math.), Szeged, 7* (1934), 39–43.

Reingold, E. M., Nievergelt, J., and Deo, N., *Combinatorial Algorithms: Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1977.

Reingold, E. M., and Tarjan, R. E., "On a Greedy Heuristic for Complete Matching," *SIAM J. Comput., 10* (1981), 676–681.

Robbins, H. E., "A Theorem on Graphs, with an Application to a Problem of Traffic Control," *Amer. Math. Monthly, 46* (1939), 281–283.

Roberts, F. S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice Hall, Englewood Cliffs, NJ, 1976.

Roberts, F. S., *Graph Theory and Its Applications to Problems of Society*, NSF-CBMS Monograph No. 29, SIAM, Philadelphia, 1978.

Roberts, F. S., and Xu, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. I: Large Grids," *SIAM J. Discr. Math., 1* (1988), 199–222.

Roberts, F. S., and Xu, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. II: Two East-West Avenues or North-South Streets," *Networks, 19* (1989), 221–233.

Roberts, F. S., and Xu, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. III: Three East-West Avenues or North-South Streets," *Networks, 22* (1992), 109–143.

Roberts, F. S., and Xu, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. IV: Four East-West Avenues or North-South Streets," *Discr. Appl. Math., 49* (1994), 331–356.

www.ebook3000.com

TARJAN, R. E., "Depth-First Search and Linear Graph Algorithms," *SIAM J. Comput., 1* (1972), 146–160.

TUCKER, A. C., and BODIN, L. D., "A Model for Municipal Street-Sweeping Operations," in W. F. Lucas, F. S. Roberts, and R. M. Thrall (eds.), *Discrete and System Models*, Vol. 3 of *Modules in Applied Mathematics*, Springer-Verlag, New York, 1983, 76–111.

VAN AARDENNE-EHRENFEST, T., and DE BRUIJN, N. G., "Circuits and Tress in Oriented Linear Graphs," *Simon Stevin, 28* (1951), 203–217.

VISHKIN, U., "On Efficient Parallel Strong Orientation," *Inform. Process. Lett., 20* (1985), 235–240.

WHITNEY, H., "Congruent Graphs and the Connectivity of Graphs," *Amer. J. Math., 54* (1932), 150–168.

WOODALL, D. R., "Sufficient Conditions for Circuits in Graphs," *Proc. Lond. Math. Soc., 24* (1972), 739–755.