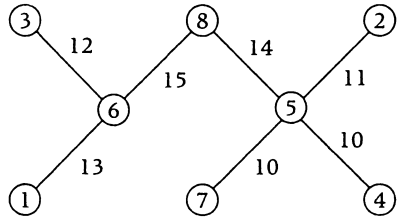
FIGURE 27.1

Cost of the tour generated by the preorder traversal of the minimum spanning tree T is less than twice the optimal tour for the TSP.

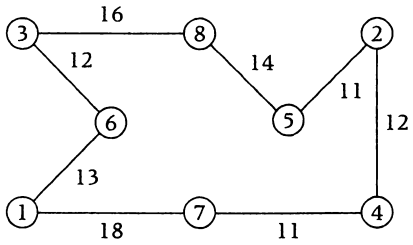
$$\begin{pmatrix} 0 & 19 & 14 & 16 & 17 & 13 & 18 & 20 \\ 19 & 0 & 17 & 12 & 11 & 19 & 15 & 15 \\ 14 & 17 & 0 & 18 & 16 & 12 & 16 & 16 \\ 16 & 12 & 18 & 0 & 10 & 20 & 11 & 20 \\ 17 & 11 & 16 & 10 & 0 & 16 & 10 & 14 \\ 13 & 19 & 12 & 20 & 16 & 0 & 18 & 15 \\ 18 & 15 & 16 & 11 & 10 & 18 & 0 & 17 \\ 20 & 15 & 16 & 20 & 14 & 15 & 17 & 0 \end{pmatrix}$$

Cost matrix $C=(c_{ij})$ satisfying triangle inequality, where c_{ij} is the cost of edge $\{i,j\}$ in K_8



A minimum spanning tree T in weighted graph K_8 having preorder traversal

1 6 3 8 5 2 4 7



Tour generated by preorder traversal of T

27.2 Bin Packing

Suppose we have a set of n objects $a_1, a_2, \ldots, a_n$ of sizes $f_1, f_2, \ldots, f_n$, respectively, where $0 < f_i < 1$, $i = 1, \ldots, n$. The *bin-packing* optimization problem is to place (pack) the objects into bins $B_1, \ldots, B_m$ having unit size using the minimum number $m \leq n$ of bins. Each bin can contain any subset of objects whose total size does not exceed 1. If m^* denotes the minimum number of bins in an optimal solution, then it is easy to see that $m^* \geq \lceil F \rceil$, where $F = \sum_{i=1}^{n} f_i$ (see Exercise 27.6). The decision version asks, for a given integer k, whether we can pack the objects

using no more than k bins. It can be seen that that bin-packing problem is NP-complete even for $k = 2$ by showing that sum of subsets reduces to bin packing (see Exercise 26.17 in Chapter 26). In this section, we discuss two strategies, next fit and decreasing first fit, that yield a 2-approximation and (an almost) 1.5-approximation, respectively.

27.2.1 Next Fit

The *next-fit* strategy for solving the bin-packing problem proceeds by successively placing the objects $a_1, a_2, \ldots, a_n$ into the bins as follows. A bin is considered *open* if the next object can be placed in the bin; otherwise, it is considered *closed*. We begin by placing a_1 into B_1. In general, when placing the next object a_i, if B_j is the last bin used and a_i fits into B_j, we place it there and continue; otherwise, we close B_j and place a_i into B_{j+1}.

We now consider now many bins are used by the next-fit strategy. Let m denote the number of bins used by the next-fit algorithm, and let s_i denote the sum of the sizes over all the objects that have been placed in the i^{th} bin B_i by the algorithm. Clearly, $F = \sum_{i=1}^{m} s_i$. It follows from the design of the next-fit algorithm that no object placed in bin $B_i, i = 2, \ldots, m$ could be added to the objects that were placed in bin B_{i-1}, without exceeding the capacity of B_{i-1}, (otherwise, the object should have been added to B_{i-1} before it was closed). Thus, we have

$$\begin{aligned} s_2 &> 1 - s_1, \\ s_3 &> 1 - s_2, \\ &\ldots \\ s_m &> 1 - s_{m-1}. \end{aligned} \qquad \textbf{(27.2.1)}$$

Observing that the values on the left side of Formula (27.2.1) sum to $F - s_1$ and the values on the right side sum to $m - 1 - (F - s_m)$, we have that $F - s_1 > m - 1 - (F - s_m)$, or, equivalently,

$$m < 2F + (1 - s_1 - s_m). \qquad \textbf{(27.2.2)}$$

Because m is an integer, it follows from Formula (27.2.2) that

$$m \leq \lceil 2F \rceil \leq 2\lceil F \rceil \leq 2m^*. \qquad \textbf{(27.2.3)}$$

The inequalities in Formula (27.2.3) show that the number m of bins used by the next-fit algorithm is at most double the number m^* of bins used by an optimal algorithm, so that the next-fit algorithm is a 2-approximation.

27.2.2 First Fit

The *first-fit* heuristic is similar to next fit except that the next object a_i is placed in the first available bin that has enough remaining capacity to fit in the object. Unlike next fit, we assume that all the bins are open. It can be shown (see Exercise 27.10) that if m denotes the number of bins used by a first-fit solution, then

$$m \leq 1.7\, m^* + 2. \qquad \textbf{(27.2.4)}$$

However, in the *first-fit decreasing* version of the algorithm, where the objects are first sorted in decreasing order of size—that is, $f_1 \geq f_2 \geq \cdots \geq f_n$—we do better than Formula (27.2.4), obtaining

$$m \leq 1.5\, m^* + 1. \qquad \textbf{(27.2.5)}$$

To verify inequality (27.2.5), we partition the objects into the following four sets:

$$A = \left\{a_i : f_i > \frac{2}{3}\right\}$$
$$B = \left\{a_i : \frac{1}{2} < f_i \leq \frac{2}{3}\right\}$$
$$C = \left\{a_i : \frac{1}{3} < f_i \leq \frac{1}{2}\right\}$$
$$D = \left\{a_i : f_i \leq \frac{1}{3}\right\}$$

There are two cases to consider.

Case 1. *D is nonempty, and one bin contains all the objects in D.* It follows from the design of the algorithm that all the items must be contained in the last bin B_m. Further, all bins except B_m must have used more than two-thirds of their capacities—that is $s_i > 2/3$, $i = 1, \ldots, m - 1$—otherwise, all the objects in bin B_m would fit in whichever of these bins had two-thirds or less capacity, and the algorithm would never have put the objects into B_m. It follows that

$$F = \sum_{i=1}^{m} s_i \geq \sum_{i=1}^{m-1} s_i \geq \frac{2}{3}(m - 1).$$

Therefore, we have

$$m \leq 1.5F + 1 \leq 1.5\lceil F \rceil + 1 \leq 1.5m^* + 1.$$

Case 2. *No bin contains all the objects in D, or D is empty.* In this case, we can remove all the items from D without changing the number of bins. Thus, case 2 can be

reduced to the special case of the bin-packing problem in which all the objects belong to A, B, or C—that is, have size of at least one-third. However, for this special case, it is not difficult to show that the first-fit decreasing algorithm actually produces an optimal solution (see Exercise 27.11).

27.3 The Steiner Tree Problem

The Steiner tree problem generalizes both the minimum spanning tree and the shortest-path problems for undirected graphs, as discussed in Chapter 12. The Steiner tree problem has important applications to establishing efficient network communication between a group S of nodes. Let $G = (V, E)$ be a graph on node (vertex) set V and edge set E representing a network, and suppose W is a positive real weighting of the edges. Given a subset S of k nodes in the graph G, a *Steiner tree* for S is a minimum-weight subtree of G that contains all the nodes in S (see Figure 27.2). Note that a Steiner tree for $S = V$ is a minimum spanning tree, and a Steiner tree for $S = \{u, v\}$ is a shortest path joining u and v. Given S and an integer k, the problem of determining whether there exists a tree having k edges that contains the nodes of S is NP-complete (see Exercise 27.15), so that the Steiner tree problem is NP-hard. However, in this section, we describe a 2-approximation algorithm for the Steiner tree problem. In fact, the algorithm we describe is slightly better than a 2-approximation, yielding a tree whose weight is at most $2 - 2/k$ times the weight of a Steiner tree.

Given S, in the first stage of constructing a 2-approximation for a Steiner tree T^* for S, we compute the matrix D of weighted distances between every pair of vertices u and v in S. The matrix D can be computed in time $O(kn^2)$ by applying Dijkstra's shortest-path algorithm (see Chapter 12) k times, once for each node in S as the root. Now consider the complete graph $H = (S, E_H)$ on S, where each

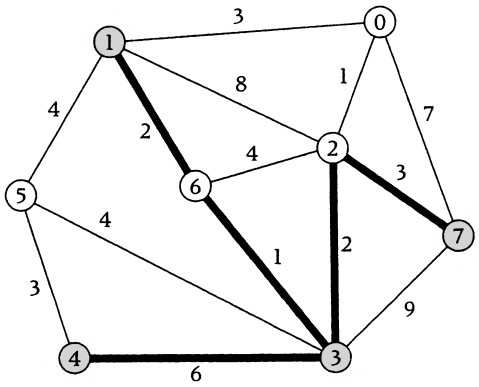


FIGURE 27.2
A Steiner tree for S = {1, 3, 4, 7}.

and hence

$$\sum_{i\in\{1,\ldots,k\}-\{j\}}\omega(W_i) = (2 - 2/k)\omega(T^*). \qquad \textbf{(27.3.3)}$$

Because the weight of W_i is at least as great as distance $D(s_i, s_{i+1})$ (length of a shortest path) from s_i to s_{i+1}, it follows that

$$\sum_{i\in\{1,\ldots,k\}-\{j\}}D(s_i, s_{i+1}) = (2 - 2/k)\omega(T^*). \qquad \textbf{(27.3.4)}$$

Since the set of edges $\{\{s_i, s_{i+1}\} | \ i \in \{1, \ldots k\} - \{j\}\}$ determines a spanning tree in H, it follows that

$$\sum_{i\in\{1,\ldots,k\}-\{j\}}D(s_i, s_{i=1}) \geq D(T_H) \geq \omega(T_G). \qquad \textbf{(27.3.5)}$$

Combining Formulas (27.3.4) and (27.3.5) yields

$$\omega(T_G) \leq (2 - 2/k)\omega(T^*). \qquad \blacksquare$$

27.4 The Facility Location Problem

Consider a network represented by a graph $G = (V, E)$, together with a real-valued distance function $d(u, v)$ defined for each pair of vertices $u, v \in V$, satisfying the *triangle inequality;* that is, $d(x, y) + d(y, z) \leq d(x, y)$ for every triple of vertices x, y, z. For example, $d(u, v)$ could be the weight of a shortest path from u to v with respect to some real weighting of the edges of G, where there are no negative cycles.

The *facility* or *server* location problem is the problem of locating k facilities (servers) at k vertices of the graph, so that the farthest distance r from any vertex of the graph to the nearest facility (server) is minimized. We call the set of vertices C that satisfies this minimization condition a *k-center* of the graph G, and we call the associated minimum value r_k^* of r the k-radius of G. More formally, we define the distance $d(U, v)$ from a subset of vertices U to a vertex v to be the minimum distance from a vertex $u \in U$ to v, and define $r(U)$ by

$$r(U) = \max \{d(U, v) \mid v \in V\}$$

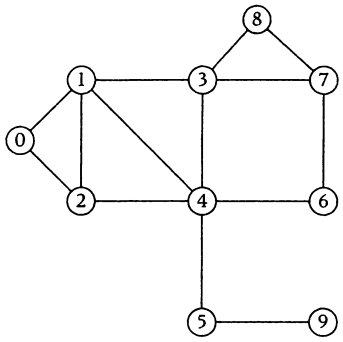
In Figure 27.5, $d(x, y)$, $d(v, y)$ and $r(U)$ are shown for a sample graph G and sample set U of size 3, where the distance $d(x, y)$ is taken to be the length of a shortest path from x to y in G. The *k-radius* of G is given by

$$r_k^* = \min \{r(U) \mid U \subseteq V, \mid U \mid = k\},$$

and a *k-center* is a subset C of vertices of cardinality k such that $r(C) = r_k^*$.

FIGURE 27.5

(a) Sample graph G and distances $d(x, y)$; (b) $d(U, v)$ and $r(U)$ for $U = \{0, 1, 8\}$.



$d(i,j)$

i / j	0	1	2	3	4	5	6	7	8	9
0	0	1	1	2	2	3	3	3	3	4
1	1	0	1	1	1	2	2	2	2	3
2	1	1	0	2	1	2	2	3	3	3
3	2	1	2	0	1	2	2	1	1	3
4	2	1	1	1	0	1	1	2	2	2
5	3	2	2	2	1	0	2	3	3	1
6	3	2	2	2	1	2	0	1	2	3
7	3	2	2	1	2	3	1	0	1	4
8	3	2	3	1	2	3	2	1	0	4
9	4	3	3	3	2	1	3	4	4	0

(a)

$U=\{0,1,8\}$

i	0	1	2	3	4	5	6	7	8	9
$D(U,i)$	0	0	1	1	1	2	2	1	0	3

$r(U) = 3$

(b)

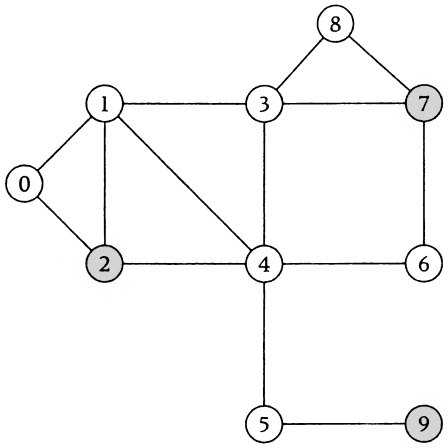


FIGURE 27.6

A 3-center C with radius $r(C) = r_3^* = 1$ for the same graph as in Figure 27.5.

In Figure 27.6, a 3-center with radius $r_3^* = 1$ is shown for the same graph as in Figure 27.5. In the case where the k-radius $r_3^* = 1$, a k-center C is a *dominating set*—that is, a subset of vertices of a given size k such that every vertex not in the set is adjacent at least one vertex in the set. The decision version of the dominating-set problem—Does there exist a dominating set of size k?—is a well-known NP-complete problem (see Exercise 27.17). Thus, the problem of determining r_3^* is NP-hard. It is interesting that the problem of obtaining an approximation that is better than a 2-approximation for the facility location problem is also NP-hard; that is, the problem of finding a k-approximation to r_3^* for $k < 2$ is NP-hard (see Exercise 27.18).

The following theorem shows how we can use a greedy heuristic to obtain a 2-approximation to the k-center problem.

Theorem 27.4.1 Let the set A be constructed by starting with any vertex a_1 and iterating as follows. For $1 < i \leq k$, assuming that $\{a_1, \ldots, a_{i-1}\}$ has been chosen, then add to A a vertex a_i such that that $d(A, a_i)$ is maximized. Then, A is a 2-approximation to a k-center.

PROOF

Let $C = \{c_1, \ldots, c_k\}$ be a k-center. For each vertex c_i in the center C, let R_i denote the subset of all vertices whose distance from c_i is at most r_k^*, and let $R = \{R_1, \ldots, R_k\}$. Because C is a k-center, every node v must lie in at least one subset from R. First suppose that each node of A lies in a unique subset from R, and consider any node v of G. Let R_j denote the subset of R that v belongs to and let a_p denote the node from A that lies in R_j. Then, applying the triangle inequality, we have

$$d(A, v) \leq d(a_p, v) \leq d(a_p, c_j) + d(c_j, v) \leq 2r_k^*.$$

On the other hand, suppose that two nodes of A, say a_p and a_q, where $p < q$, lie in the same subset of R, say R_j. Choose the smallest index q with this property; that is, $a_1, a_2, \ldots, a_{p-1}$ each lie in different subsets of R, but a_p lies in the same subset as a_q. Now consider any vertex v. Since a_q was chosen using the greedy method, we have

$$d(\{a_1, \ldots, a_{q-1}\}, v) \leq d(\{a_1, \ldots, a_{q-1}\}, a_q). \tag{27.4.1}$$

Because $a_p \in \{a_1, \ldots, a_{q-1}\}$, it follows from the definition of $d(U, v)$ that

$$d(\{a_1, \ldots, a_{q-1}\}, a_q) \leq d(a_p, a_q) \tag{27.4.2}$$

Again, applying the triangle inequality we have

$$d(a_p, a_q) \leq d(a_p, c_j) + d(c_j, a_q) \leq 2r_k^*. \tag{27.4.3}$$

Combining Formulas (27.4.1), (27.4.2) and (27.4.3), we have

$$d(A, v) \leq d(\{a_1, \ldots, a_{q-1}\}, v) \leq d(\{a_1, \ldots, a_{q-1}\}, a_q) \leq d(a_p, a_q) \leq 2r_k^*. \quad \blacksquare$$

Theorem 27.4.1 yields a greedy 2-approximation algorithm for the facility location problem. The approximation 3-center A generated by this greedy algorithm for the same graph as in Figure 27.6 is shown in Figure 27.7. Note that $r(A) = 2$, which is twice the actual 3-radius $r_3^* = 1$.

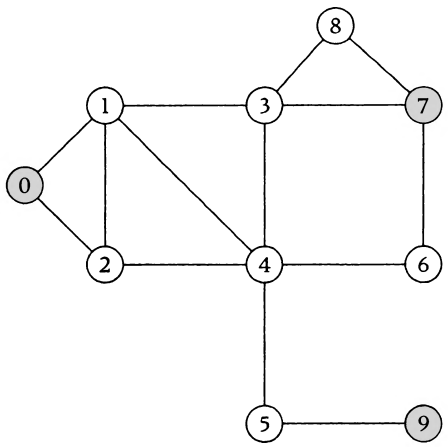


FIGURE 27.7

Approximation 3-center $A = \{0, 7, 9\}$ generated by a greedy algorithm based on Theorem 27.4.1 in the order 0, 9, 7 for same graph as in Figure 27.6. Note that $r(A) = 2$, which is twice the actual 3-radius $r_3^* = 1$.

27.5 Closing Remarks

Although we have seen that an $O(n^2)$ 2-approximation algorithm for the TSP exists under the assumption that the triangle inequality holds, the problem of finding a k-approximation algorithm for the TSP, for any constant k, is NP-hard if we do not make this assumption (see Exercise 27.3). In this chapter, we discussed a 2-approximation algorithm for the Steiner tree problem. Using more-sophisticated arguments, Robins and Zelikovski designed a 1.55-approximate algorithm to the Steiner tree problem.

Because NP-hard problems arise in so many practical situations, much research has been devoted to finding approximation algorithms for these problems. We refer the reader to the references for further discussion of this important research area.

References and Suggestions for Further Reading

Three books on approximation algorithms:

Ausiello, G., P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. New York: Springer-Verlag, 1999.

Hochbaum, D. (ed.). *Approximation Algorithms for NP-Hard Problems*. Boston: PWS, 1997.

Vazirani, V. V. *Approximation Algorithms*. Berlin: Springer-Verlag, 2001.

Coffman, E.G., M. R. Garey, and D.S. Johnson. "Approximation Algorithms for Bin Packing—An Updated Survey." In *Algorithm Design for Computer System Design*, edited by G. Ausiello, M. Lucertini, and P. Serafini, 46—93. New York: Springer-Verlag, 1984. A survey paper on approximation algorithms for bin packing.

EXERCISES

Section 27.1 The Traveling Salesman Problem

27.1 Find an optimal tour for the instance of the TSP given in Figure 27.1, and compare your tour with the tour shown in Figure 27.1 generated by the 2-approximation algorithm.

27.2 Show that the 2-approximation algorithm provided by the minimum spanning tree argument does not yield a ρ-approximation for any $\rho < 2$; that is, for every n, give a weighting of K_n satisfying the triangle inequal-

ity such that the tour generated by the approximation algorithm is strictly greater than ρ times the optimal tour.

27.3 Show that if a polynomial algorithm A exists yielding a ρ-approximation for the general TSP problem, then a polynomial algorithm exists for the Hamiltonian cycle decision problem. [*Hints*: Let $G = (V, E)$ be any instance of the Hamiltonian cycle decision problem. Consider the instance of the TSP on the complete graph $G' = (V, E')$ with the following weighting: $\omega(u, v) = 1$ if $uv \in E$, otherwise $\omega(u, v) = \rho|V| + 1$. Show that G' has a tour of cost $\leq \rho|V|$ if, and only if, G has a Hamiltonian cycle.]

27.4 Show the action of the 2-approximation algorithm for the TSP given in Section 27.1 for ten nodes, where the weighted adjacency matrix of K_{10} is given by the matrix in Figure 27.5(a).

27.5 Write a program implementing the 2-approximation algorithm for the TSP given in Section 27.1.

Section 27.2 Bin Packing

27.6 Let m^* denote the minimum number of bins in an optimal solution to the bin-packing problem. Show that $m^* \geq \lceil F \rceil$, where $F = \sum_{i=1}^{n} f_i$.

27.7 Show the action of the next-fit heuristic for the following instances of the bin-packing problem:

a. $(f_1, f_2, \ldots, f_9) = (.1, .6, .4, .1, .7, .4, .15, .35, .9)$.

b. $(f_1, f_2, \ldots, f_{11}) = (.5, .3, .4, .9, .7, .4, .15, .35, .05, .95, .1)$.

27.8 Repeat Exercise 27.7 for first fit.

27.9 Repeat Exercise 27.7 for first fit decreasing.

27.10 Verify Formula (27.2.4).

27.11 Complete the argument for case 2 of the verification of inequality (27.2.5) as follows:

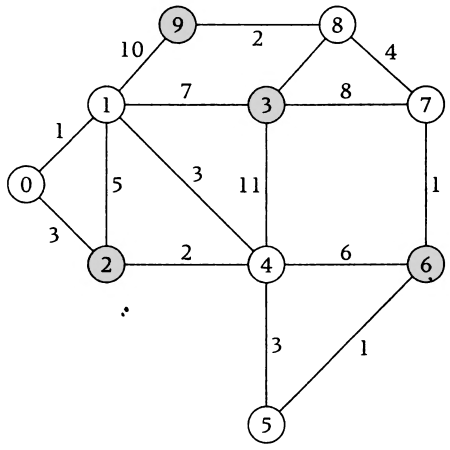
a. Show that case 2 can be reduced to the special case of the bin-packing problem in which all the objects belong to A, B, or C—that is, have size of at least one-third.

b. Show that if all the objects have size of at least one-third, then the first-fit decreasing algorithm produces an optimal solution.

27.12 Write a program implementing and comparing the next-fit, first-fit, and first-fit decreasing heuristics.

Section 27.3 The Steiner Tree Problem

27.13 Show the action of the 2-approximation algorithm for finding an approximation to the Steiner tree problem given in Section 27.3 for the set S in the following weighted graph, where the nodes of S are shaded.



27.14 Write a program implementing the 2-approximation algorithm for the Steiner tree problem given in Section 27.3.

27.15 Given a graph $G = (V, E)$, a subset S of V, and a positive real number k, show that the problem of determining whether there exists a tree with k edges containing the nodes of S is NP-complete.

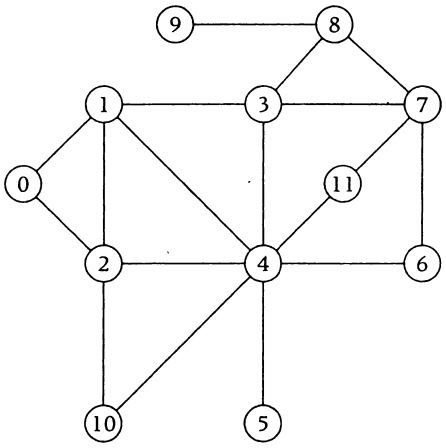
Section 27.4 The Facility Location Problem

27.16 For the value of k specified in each part, show the action of the greedy 2-approximation algorithm given in Section 27.4 for computing an approximation to a k-center in the graph G below, where $d(i, j)$ is the length of a shortest path from i to j.

a. $k = 2$.

b. $k = 3$.

c. $k = 6$.

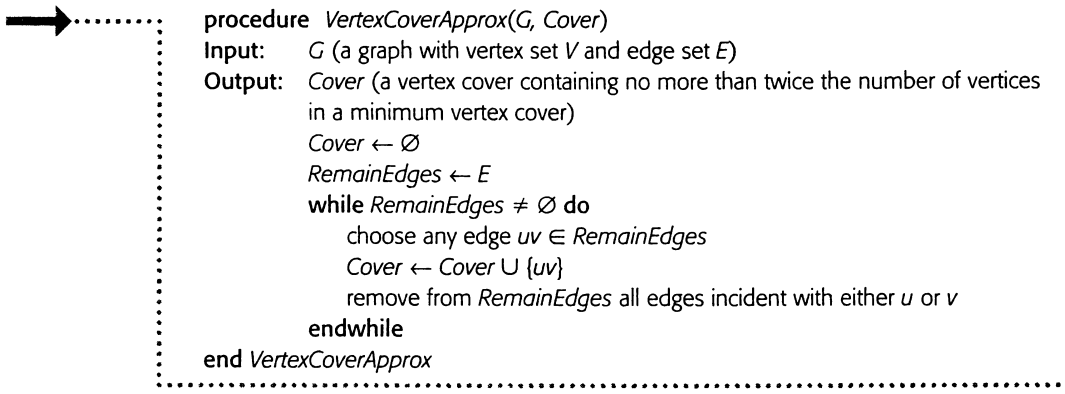


27.17 Show that the dominating-set problem is NP-complete. (*Hint*: Show that Vertex Cover $\propto$ Dominating Set.)

27.18 The decision version of the facility location problem is to determine whether the k-radius r_k^* is at most b for any given integer b. Show that the problem of finding a ρ-approximation to the facility location problem for any given $\rho < 2$ is NP-complete by a reduction from the dominating-set problem.

Additional Exercise

27.19 Consider the following algorithm for the vertex cover problem:



```
procedure VertexCoverApprox(G, Cover)
Input:   G (a graph with vertex set V and edge set E)
Output:  Cover (a vertex cover containing no more than twice the number of vertices
         in a minimum vertex cover)
         Cover ← ∅
         RemainEdges ← E
         while RemainEdges ≠ ∅ do
             choose any edge uv ∈ RemainEdges
             Cover ← Cover ∪ {uv}
             remove from RemainEdges all edges incident with either u or v
         endwhile
end VertexCoverApprox
```

Show that *VertexCoverApprox* is a 2-approximation algorithm.