$S_1^2 = \{(.72, 45), (.792, 75)\}$; $S_2^2 = \{(.864, 60)\}$. Note that the tuple $(.9504, 90)$ which comes from $(.99, 60)$ has been eliminated from S_2^2 as this leaves only $\$10$. This is not enough to allow $m_3 = 1$. The set $S_3^2 = \{(.8928, 75)\}$. Combining, we get $S^2 = \{(.72, 45), (.864, 60), (.8928, 75)\}$ as the tuple $(.792, 75)$ is dominated by $(.864, 60)$. The set $S_1^3 = \{(.36, 65), (.432, 80), (.4464, 95)\}$, $S_2^3 = \{(.54, 85), (.648, 100)\}$, and $S_3^3 = \{(.63, 105)\}$. Combining, we get $S^3 = \{(.36, 65), (.432, 80), (.54, 85), (.648, 100)\}$.

The best design has a reliability of $.648$ and a cost of 100. Tracing back through the S^i's, we determine that $m_1 = 1, m_2 = 2$, and $m_3 = 2$. $\square$

As in the case of the knapsack problem, a complete dynamic programming algorithm for the reliability problem will use heuristics to reduce the size of the S^i's. There is no need to retain any tuple (f, x) in S^i with x value greater that $c - \sum_{i \le j \le n} c_j$ as such a tuple will not leave adequate funds to complete the system. In addition, we can devise a simple heuristic to determine the best reliability obtainable by completing a tuple (f, x) in S^i. If this is less than a heuristically determined lower bound on the optimal system reliability, then (f, x) can be eliminated from S^i.

EXERCISE

1. (a) Present an algorithm similar to DKnap to solve the recurrence (5.19).

    (b) What are the time and space requirements of your algorithm?

    (c) Test the correctness of your algorithm using suitable test data.

5.9 THE TRAVELING SALESPERSON PROBLEM

We have seen how to apply dynamic programming to a subset selection problem (0/1 knapsack). Now we turn our attention to a permutation problem. Note that permutation problems usually are much harder to solve than subset problems as there are $n!$ different permutations of n objects whereas there are only 2^n different subsets of n objects $(n! > 2^n)$. Let $G = (V, E)$ be a directed graph with edge costs c_{ij}. The variable c_{ij} is defined such that $c_{ij} > 0$ for all i and j and $c_{ij} = \infty$ if $\langle i, j \rangle \notin E$. Let $|V| = n$ and assume $n > 1$. A *tour* of G is a directed simple cycle that includes every vertex in V. The cost of a tour is the sum of the cost of the edges on the tour. The *traveling salesperson problem* is to find a tour of minimum cost.

The traveling salesperson problem finds application in a variety of situations. Suppose we have to route a postal van to pick up mail from mail

boxes located at n different sites. An $n + 1$ vertex graph can be used to represent the situation. One vertex represents the post office from which the postal van starts and to which it must return. Edge $\langle i, j \rangle$ is assigned a cost equal to the distance from site i to site j. The route taken by the postal van is a tour, and we are interested in finding a tour of minimum length.

As a second example, suppose we wish to use a robot arm to tighten the nuts on some piece of machinery on an assembly line. The arm will start from its initial position (which is over the first nut to be tightened), successively move to each of the remaining nuts, and return to the initial position. The path of the arm is clearly a tour on a graph in which vertices represent the nuts. A minimum-cost tour will minimize the time needed for the arm to complete its task (note that only the total arm movement time is variable; the nut tightening time is independent of the tour).

Our final example is from a production environment in which several commodities are manufactured on the same set of machines. The manufacture proceeds in cycles. In each production cycle, n different commodities are produced. When the machines are changed from production of commodity i to commodity j, a change over cost c_{ij} is incurred. It is desired to find a sequence in which to manufacture these commodities. This sequence should minimize the sum of change over costs (the remaining production costs are sequence independent). Since the manufacture proceeds cyclically, it is necessary to include the cost of starting the next cycle. This is just the change over cost from the last to the first commodity. Hence, this problem can be regarded as a traveling salesperson problem on an n vertex graph with edge cost c_{ij}'s being the changeover cost from commodity i to commodity j.

In the following discussion we shall, without loss of generality, regard a tour to be a simple path that starts and ends at vertex 1. Every tour consists of an edge $\langle 1, k \rangle$ for some $k \in V - \{1\}$ and a path from vertex k to vertex 1. The path from vertex k to vertex 1 goes through each vertex in $V - \{1, k\}$ exactly once. It is easy to see that if the tour is optimal, then the path from k to 1 must be a shortest k to 1 path going through all vertices in $V - \{1, k\}$. Hence, the principle of optimality holds. Let $g(i, S)$ be the length of a shortest path starting at vertex i, going through all vertices in S, and terminating at vertex 1. The function $g(1, V - \{1\})$ is the length of an optimal salesperson tour. From the principal of optimality it follows that

$$g(1, V - \{1\}) = \min_{2 \leq k \leq n} \{c_{1k} + g(k, V - \{1, k\})\} \tag{5.20}$$

Generalizing (5.20), we obtain (for $i \notin S$)

$$g(i, S) = \min_{j \in S}\{c_{ij} + g(j, S - \{j\})\} \tag{5.21}$$

Equation 5.20 can be solved for $g(1, V - \{1\})$ if we know $g(k, V - \{1, k\})$ for all choices of k. The g values can be obtained by using (5.21). Clearly,

$g(i, \phi) = c_{i1}$, $1 \le i \le n$. Hence, we can use (5.21) to obtain $g(i, S)$ for all S of size 1. Then we can obtain $g(i, S)$ for S with $|S| = 2$, and so on. When $|S| < n - 1$, the values of i and S for which $g(i, S)$ is needed are such that $i \ne 1$, $1 \notin S$, and $i \notin S$.

Example 5.26 Consider the directed graph of Figure 5.21(a). The edge lengths are given by matrix c of Figure 5.21(b).

$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$
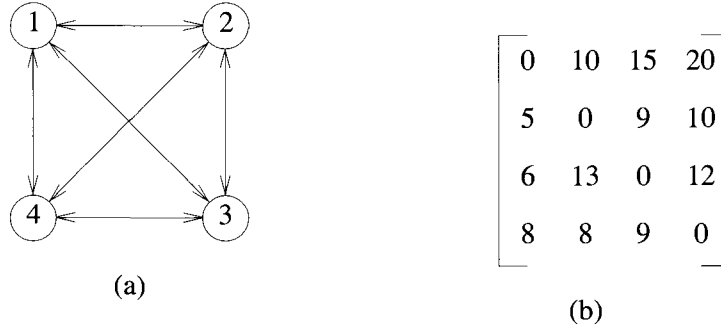
(a)

(b)

Figure 5.21 Directed graph and edge length matrix c

Thus $g(2, \phi) = c_{21} = 5, g(3, \phi) = c_{31} = 6$, and $g(4, \phi) = c_{41} = 8$. Using (5.21), we obtain

$$
\begin{array}{llll}
g(2, \{3\}) & = & c_{23} + g(3, \phi) & = & 15 & \quad g(2, \{4\}) & = & 18 \\
g(3, \{2\}) & = & 18 & & & \quad g(3, \{4\}) & = & 20 \\
g(4, \{2\}) & = & 13 & & & \quad g(4, \{3\}) & = & 15
\end{array}
$$

Next, we compute $g(i, S)$ with $|S| = 2$, $i \ne 1$, $1 \notin S$ and $i \notin S$.

$$
\begin{array}{llll}
g(2, \{3, 4\}) & = & \min\ \{c_{23} + g(3, \{4\}), c_{24} + g(4, \{3\})\} & = & 25 \\
g(3, \{2, 4\}) & = & \min\ \{c_{32} + g(2, \{4\}), c_{34} + g(4, \{2\})\} & = & 25 \\
g(4, \{2, 3\}) & = & \min\ \{c_{42} + g(2, \{3\}), c_{43} + g(3, \{2\})\} & = & 23
\end{array}
$$

Finally, from (5.20) we obtain

$$
\begin{aligned}
g(1, \{2, 3, 4\}) & = \min\{c_{12} + g(2, \{3, 4\}), c_{13} + g(3, \{2, 4\}), c_{14} + g(4, \{2, 3\})\} \\
& = \min\{35, 40, 43\} \\
& = 35
\end{aligned}
$$

An optimal tour of the graph of Figure 5.21(a) has length 35. A tour of this length can be constructed if we retain with each $g(i, S)$ the value of j that minimizes the right-hand side of (5.21). Let $J(i, S)$ be this value. Then, $J(1, \{2, 3, 4\}) = 2$. Thus the tour starts from 1 and goes to 2. The remaining tour can be obtained from $g(2, \{3, 4\})$. So $J(2, \{3, 4\}) = 4$. Thus the next edge is $\langle 2, 4 \rangle$. The remaining tour is for $g(4, \{3\})$. So $J(4, \{3\}) = 3$. The optimal tour is 1, 2, 4, 3, 1. $\qquad\square$

Let N be the number of $g(i, S)$'s that have to be computed before (5.20) can be used to compute $g(1, V - \{1\})$. For each value of $|S|$ there are $n - 1$ choices for i. The number of distinct sets S of size k not including 1 and i is $\binom{n-2}{k}$. Hence

$$N = \sum_{k=0}^{n-2} (n - 1) \binom{n-2}{k} = (n - 1)2^{n-2}$$

An algorithm that proceeds to find an optimal tour by using (5.20) and (5.21) will require $\Theta(n^2 2^n)$ time as the computation of $g(i, S)$ with $|S| = k$ requires $k - 1$ comparisons when solving (5.21). This is better than enumerating all $n!$ different tours to find the best one. The most serious drawback of this dynamic programming solution is the space needed, $O(n2^n)$. This is too large even for modest values of n.

EXERCISE

1. (a) Obtain a data representation for the values $g(i, S)$ of the traveling salesperson problem. Your representation should allow for easy access to the value of $g(i, S)$, given i and S. (i) How much space does your representation need for an n vertex graph? (ii) How much time is needed to retrieve or update the value of $g(i, S)$?

   (b) Using the representation of (a), develop an algorithm corresponding to the dynamic programming solution of the traveling salesperson problem.

   (c) Test the correctness of your algorithm using suitable test data.

5.10 FLOW SHOP SCHEDULING

Often the processing of a job requires the performance of several distinct tasks. Computer programs run in a multiprogramming environment are input and then executed. Following the execution, the job is queued for output