Traveling Salesperson

Given

- A directed graph $G = (V, E)$ with edge costs $c_{ij}$
- $c_{ij}$ is defined such that $c_{ij} > 0$ for all $i$ and $j$ and $c_{ij} = \infty$ if $\langle i,j \rangle \not\in E$.
- $|V| = n$ and $n > 1$

A tour of $G$ is a directed cycle that includes every vertex in $V$, and no vertex occurs more than once except for the starting vertex. The cost of a tour is the sum of the cost of edges on the tour. The traveling salesperson problem is to find a tour of minimum cost.

Greedy Algorithm

- Start with vertex $v_1$; call it $v_i$
- Visit the vertex $v_j$ that is nearest to $v_i$, or can be reached from $v_i$ with least cost
- Repeat the above starting at vertex $v_j$ (call it as new $v_i$) taking care never to visit a vertex already visited

Dynamic Programming Solution

- Regard the 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.
- 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 $V - \{1,k\}$.
- 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.
- $g(1,V - \{1\})$ is the length of an optimal salesperson tour.
• From the principle of optimality

\[ g(1, V - \{1\}) = \min_{2 \leq k \leq n} \{ c_{1k} + g(k, V - \{1, k\}) \} \]  

(1)

• Generalizing

\[ g(i, S) = \min_{j \in S} \{ c_{ij} + g(j, S - \{j\}) \} \]  

(2)

• Equation 1 may be solved for \( g(1, V - \{1\}) \) if we know \( g(k, V - \{1, k\}) \) for all values of \( k \)

• The \( g \) values may be obtained by using Equation 2
  
  – \( g(i, \phi) = C_{i,1}, 1 \leq i \leq n \).
  
  – We can use Equation 2 to obtain \( g(i, S) \) for all \( S \) of size 1.
  
  – Then we can obtain \( g(i, S) \) for \( S \) with \( |S| = 2 \).
  
  – When \( |S| < n - 1 \), the values of \( i \) and \( S \) for which \( g(i, S) \) is needed are such that \( i \neq 1, 1 \notin S \), and \( i \notin S \).

Example

Consider the directed graph presented below

\[
\begin{array}{cccc}
0 & 10 & 15 & 20 \\
5 & 0 & 9 & 10 \\
6 & 13 & 0 & 12 \\
8 & 8 & 9 & 0 \\
\end{array}
\]

\[
\begin{align*}
g(2, \phi) &= c_{21} = 5 \\
g(3, \phi) &= c_{31} = 6 \\
g(4, \phi) &= c_{41} = 8 \\
\end{align*}
\]

Using Equation 2, we get

\[
\begin{align*}
g(2, \{3\}) &= c_{23} + g(3, \phi) = 15 \\
g(3, \{2\}) &= 18 \\
g(4, \{2\}) &= 13 \\
\end{align*}
\]

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

\[
\begin{align*}
g(2, \{3, 4\}) &= \min \{ c_{23} + g(3, \{4\}), c_{24} + g(4, \{3\}) \} = 25 \\
g(3, \{2, 4\}) &= \min \{ c_{32} + g(2, \{4\}), c_{34} + g(4, \{2\}) \} = 25 \\
g(4, \{2, 3\}) &= \min \{ c_{42} + g(2, \{3\}), c_{43} + g(3, \{2\}) \} = 23 \\
\end{align*}
\]

Finally, from Equation 1, we obtain

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

So, an optimal tour of the above graph has cost 35. A tour of this length may be constructed if we retain with each \( g(i, S) \) the value of \( j \) that minimizes the right hand side of Equation 2. Let this value be called \( J(i, S) \). Then, \( J(1, \{2, 3, 4\}) = 2 \). Thus the tour starts from 1 and goes to 2. The remaining tour may be obtained from \( g(2, \{3, 4\}) \).
Now, $J(2, \{3, 4\}) = 4$. Thus the next edge is $(2, 4)$. The remaining tour is for $g(4, \{3\})$. $J(4, \{3\}) = 3$. The optimal tour is $1, 2, 4, 3, 1$.

**Algorithm Analysis**

Let $N$ be the number of $g(i, S)$s that have to be computed before Equation 1 may be used to compute $g(1, V - \{1\})$. For each value of $|S|$, there are $n - 1$ choices of $i$. 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 making use of Equations 1 and 2 will require $\Theta(n^22^n)$ time as the computation of $g(i, S)$ with $|S| = k$ requires $k - 1$ comparisons when solving Equation 2. This is better than enumerating all $n!$ different tours to find the best one. The most serious drawback of the dynamic programming solution is the space needed ($O(n2^n)$). This can be too large even for modest values of $n$.

**Programming Assignment**

Write three programs to solve the Traveling Salesperson problem by greedy algorithm, dynamic programming using recursion, and dynamic programming using iteration. The programs should be general enough to account for $n$ vertices in a graph. Clearly identify the algorithm used within the program in the header with a comment.

The output must contain the cost of the minimum tour and the optimal tour itself (in each of the three cases).