بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ـ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۰۱)

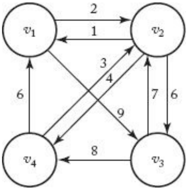
طراحی الگوریتم‌ها

حسین فلسفین

The Traveling Salesperson Problem

Suppose a salesperson is planning a sales trip that includes 20 cities. Each city is connected to some of the other cities by a road. To minimize travel time, we want to determine a shortest route that starts at the salesperson's home city, visits each of the cities once, and ends up at the home city. This problem of determining a shortest route is called the Traveling Salesperson problem.

A tour (also called a Hamiltonian circuit) in a directed graph is a path from a vertex to itself that passes through each of the other vertices exactly once. An optimal tour in a weighted, directed graph is such a path of minimum length. The Traveling Salesperson problem is to find an optimal tour in a weighted, directed graph when at least one tour exists. Because the starting vertex is irrelevant to the length of an optimal tour, we will consider v_1 *to be the starting vertex. (We assume that the weights are nonnegative numbers.)*



$length\,[v_1, v_2, v_3, v_4, v_1] = 22$

$length\,[v_1, v_3, v_2, v_4, v_1] = 26$

$length\,[v_1, v_3, v_4, v_2, v_1] = 21$

The last tour is optimal.

We solved this instance by simply considering all possible tours. In general, there can be an edge from every vertex to every other vertex. If we consider all possible tours, the second vertex on the tour can be any of $n-1$ vertices, the third vertex on the tour can be any of $n-2$ vertices, ..., the nth vertex on the tour can be only one vertex. Therefore, the total number of tours is $(n-1)(n-2)\cdots 1 = (n-1)!$, which is worse than exponential.

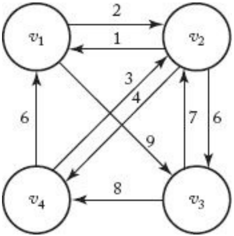
We represent the graph by an adjacency matrix W.

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$V = \text{set of all the vertices}$$

$$A = \text{a subset of } V$$

$$D[v_i][A] = \text{length of a shortest path from } v_i \text{ to } v_1 \text{ passing through each vertex in } A \text{ exactly once.}$$



	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$A = \{v_3\} \text{ ☞ } D[v_2][A] = length[v_2, v_3, v_1] = \infty.$$

If $A = \{v_3, v_4\}$***, then***

$$\begin{aligned} D[v_2] &= minimum\,(length\,[v_2, v_3, v_4, v_1]\,,\; length\,[v_2, v_4, v_3, v_1]) \\ &= minimum\,(20, \infty) = 20. \end{aligned}$$

Because $V - \{v_1, v_j\}$ contains all the vertices except v_1 and v_j we have

$$\text{Length of an optimal tour} = \underset{2\le j\le n}{minimum}(W[1][j] + D[v_j][V - \{v_1, v_j\}]),$$

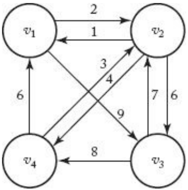
and, in general for $i \neq 1$ and v_i not in A,

$$D[v_i][A] = \underset{j:v_j\in A}{minimum}(W[i][j] + D[v_j][A - \{v_j\}]) if A \neq \varnothing$$
$$D[v_i][\varnothing] = W[i][1].$$

الگوریتم مبتنی بر راهبرد برنامه‌ریزی پویا برای مسئلهٔ ***TSP*** اینگونه عمل می‌کند که ابتدا مقادیر $D[v_i][\varnothing]$ها را محاسبه می‌کند. سپس مقادیر $D[v_i][A]$ را برای مجموعه‌های A با تنها یک عضو محاسبه می‌کند. سپس مقادیر $D[v_i][A]$ را برای مجموعه‌های A با دو عضو محاسبه می‌کند. و به‌همین ترتیب پیش می‌رود تا آنکه مقادیر $D[v_i][A]$ را برای مجموعه‌های A با $n-2$ عضو محاسبه می‌کند. سرانجام باید از رابطهٔ

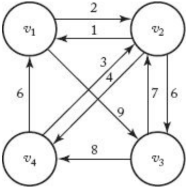
$$\text{Length of an optimal tour} = \underset{2\le j\le n}{minimum}(W[1][j] + D[v_j][V - \{v_1, v_j\}]),$$

استفاده کرد.



	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_2][\varnothing] = 1$$
$$D[v_3][\varnothing] = \infty$$
$$D[v_4][\varnothing] = 6$$



	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_3][\{v_2\}] = \underset{j:v_j\in\{v_2\}}{minimum}(W[3][j] + D[v_j][\{v_2\} - \{v_j\}])$$

$$= W[3][2] + D[v_2][\varnothing] = 7 + 1 = 8$$

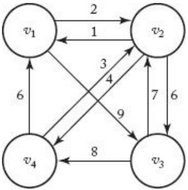
$$D[v_4][\{v_2\}] = 3 + 1 = 4$$

$$D[v_2][\{v_3\}] = 6 + \infty = \infty$$

$$D[v_4][\{v_3\}] = \infty + \infty + \infty$$

$$D[v_2][\{v_4\}] = 4 + 6 = 10$$

$$D[v_3][\{v_4\}] = 8 + 6 = 14$$

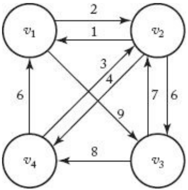


	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$
\begin{aligned}
D[v_4][\{v_2, v_3\}] &= \underset{j: v_j \in \{v_2, v_3\}}{minimum}(W[4][j] + D[v_j][\{v_2, v_3\} - \{v_j\}]) \\
&= minimum(W[4][2] + D[v_2][\{v_3\}], W[4][3] + D[v_3][\{v_2\}]) \\
&= minimum(3 + \infty, \infty + 8) = \infty
\end{aligned}
$$

$$D[v_3][\{v_2, v_4\}] = minimum(7 + 10, 8 + 4) = 12$$

$$D[v_2][\{v_3, v_4\}] = minimum(6 + 14, 4 + \infty) = 20$$



	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$
\begin{aligned}
D[v_1][\{v_2, v_3, v_4\}] &= \underset{j: v_j \in \{v_2, v_3, v_4\}}{minimum} (W[1][j] + D[v_j][\{v_2, v_3, v_4\} - \{v_j\}]) \\
&= minimum(W[1][2] + D[v_2][\{v_3, v_4\}], \\
&\qquad\qquad W[1][3] + D[v_3][\{v_2, v_4\}], \\
&\qquad\qquad W[1][4] + D[v_4][\{v_2, v_3\}]) \\
&= minimum(2 + 20, 9 + 12, \infty + \infty) = 21
\end{aligned}
$$

The Dynamic Programming Algorithm for TSP

***Problem:** Determine an optimal tour in a weighted, directed graph. The weights are nonnegative numbers.*

***Inputs:** A weighted, directed graph, and n, the number of vertices in the graph. The graph is represented by a two-dimensional array W, which has both its rows and columns indexed from 1 to n, where $W[i][j]$ is the weight on the edge from ith vertex to the jth vertex.*

***Outputs:** A variable $minlength$, whose value is the length of an optimal tour, and a two-dimensional array P from which an optimal tour can be constructed. P has its rows indexed from 1 to n and its columns indexed by all subsets of $V - \{v_1\}$. $P[i][A]$ is the index of the first vertex after v_i on a shortest path from v_i to v_1 that passes through all vertices in A exactly once.*

```
void travel (int n, const number W[][], index P[][],
             number& minlength)
{
  index i, j, k; number D[1..n][subset of V − {v1}];

  for (i = 2; i <= n; i++)
     D[i][∅] = W[i][1];
  for (k = 1; k <= n − 2; k++)
     for (all subsets A ⊆ V − {v1} containing k vertices)
        for (i such that i ≠ 1 and vi is not in A){
           D[i][A] = minimum   (W[i][j] + D[j][A − {vj}]);
                     j: vj ∈ A
           P[i][A] = value of j that gave the minimum;
        }
  D[1][V − {v1}] = minimum   (W[1][j] + D[j][V − {v1, vj}]);
                   2 ≤ j ≤ n
  P[1][V − {v1}] = value of j that gave the minimum;
  minlength = D[1][V − {v1}];
}
```

Every-Case Time and Space Complexity

Basic operation: The time in both the first and last loops is insignificant compared to the time in the middle loop because the middle loop contains various levels of nesting. Therefore, we will consider the instructions executed for each value of v_j to be the basic operation. They include an addition instruction.

Input size: n, the number of vertices in the graph. For each set A containing k vertices, we must consider $n-1-k$ vertices, and for each of these vertices, the basic operation is done k times. Because the number of subsets A of $V-\{v_1\}$ containing k vertices is equal to $\binom{n-1}{k}$, the total number of times the basic operation is done is given by

$$T(n)=\sum_{k=1}^{n-2}(n-1-k)k\binom{n-1}{k}.$$