Shreya Thigale
TE-IX
33168

PAGE NO.
DATE / /

ASSIGNMENT - 10

TITLE: Travelling Salesman Problem (DP)

PROBLEM STATEMENT:

Write a program to solve the travelling salesman problem and to print the path and the cost using dynamic programming

OBJECTIVE:

To understand and implement dynamic programming algorithm for solving Travelling salesman problem and study dynamic programming.

THEORY:

A travelling salesman is getting ready for big sales tour. Starting at his hometown, suitcase in hand, he will conduct a journey in which each of his target cities is visited exactly once before he returns home. Given pairwise costs/ distances between cities, we have to find best order in which he can visit them so as to find minimum cost.

Let G(V,E) be directed graph defining an instance of travelling Salesperson problem. Let C_{ij} be the cost of edge $<i,j>$, $C_{ij} = \infty$ if $<i,j>$ does not belong to E and let $|V| = n$. Without loss of generality, we may assume that every tour starts and ends at vertex 1.

Smeeya Thigale
TE-IX
33168

So, solution space S is given by
$S = \{1, \pi, 1 \mid \pi$ is pumutation of $(2,3,\dots,n)\}$.
$|S| = (n-1)!$ The size of S may be reduced by restricting S so that $(1, i_1, i_2, \dots i_{n-1}, 1) \in S$ iff $\langle i_j, i_{j+1} \rangle \in E$, $0 \le j \le n+1$ $i_0 = i_n = 1$.
The travelling salesman problem is to find a tour of minimum cost.

Dynamic Programming Strategy:

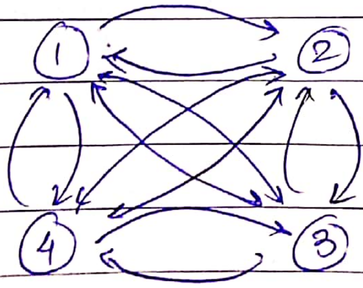
An optimality policy has the property that whatever the initial state and initial descision, are, the remaining descisions must constitute an optimal policy with regard to state resulting from first descision.
From the principle of optimality it follows:

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

Generalising, we obtain:

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

EXAMPLE:



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

Scanned by CamScanner

Shreya Thigale
TE-IX
33168

$g(2,\phi) = c_{21} = 5$; $g(3,\phi) = c_{31} = 6$; $g(4,\phi) = c_{41} = 8$

$g(2,\{3\}) = c_{23} + g(3,\phi) = 15$;
$g(2,\{4\}) = c_{24} + g(4,\phi) = 18$;
$g(3,\{2\}) = 18$; $g(3,\{4\}) = 20$;
$g(4,\{2\}) = 13$; $g(4,\{3\}) = 15$;

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

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

$J(1,\{2,3,4\}) = 2$
$J(2,\{3,4\}) = 4$
$J(4,\{3\}) = 3$
Final optimal tour : 1-2-4-3-1
Cost of tour = 35

ALGORITHM FOR TSP using DP:

```
g(1, {1}) = 0
for s=2 to n do
    for all subsets S of s and containing 1:
        g(1,S) = 8
        for all j∈S; j≠1 do
```

Shreeya Thigale
TE-IX
33168

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

return min $g(i,S)$

Analysis:

The brute force approach is to evaluate every possible tour and return the best one. Since there are $(n-1)!$ possibilities of visiting the n points, which results in runtime of $O(n!)$

By dynamic programming:

There are at most $2^n.n$ subproblems and each one takes linear time to solve. The total running time is therefore $O(n^2.2^n)$

CONCLUSION:

Thus we have studied the TSP problem and studied that TSP by DP takes $O(n^2 2^n)$ running time as compared to $(n-1)!$ by brute force and also implemented TSP by DP.