Assignment -10

33119

Title : Travelling Salesman Problem using Dynamic Programming.

Problem Statement :

A traveller needs to visit all the cities from a list, where distances between all the cities are known and each city should be visited once. Find shortest possible route that he visits each city exactly. once and return to origin city

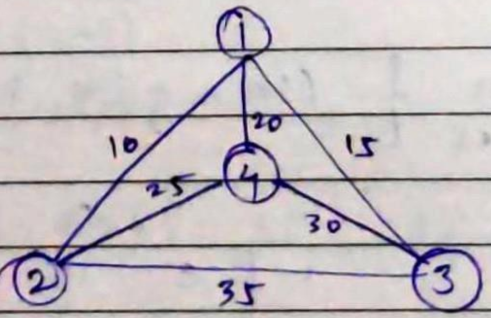
Objective : To implement the solution for Travelling Salesman Problem using Dynamic Programming.

Theory :

1.) What is TSP?

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once & returns to the starting point.

For ex.



The TSP tour in above graph i, 1-2-4-3-1 The cost is 80, TSP is a famous NP hard problem.

Q2.) What is Dynamic Programming ? and how to solve TSP using DP?

Dynamic Programming is mainly an optimization over plain recurrsion, wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using Dynamic programming. The idea is to simply store the results of sub-problems, so that we do not have to re-compute them when needed later.

Let the cost of this path be cost (i), the cost of corresponding cycle would be cost(i) + dist (i,1) where dist (i,1) is distance from i to 1.

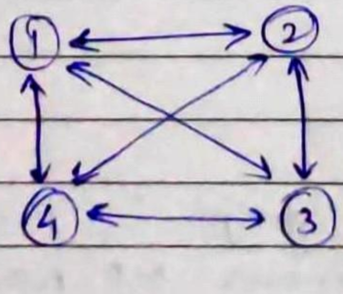
- Algorithmic approach for the TSP using DP:

We divide given main problem into multiple subproblems by constructing recursive tree and obtaining solutions of these subproblems to use them to solve the main problem collectively.

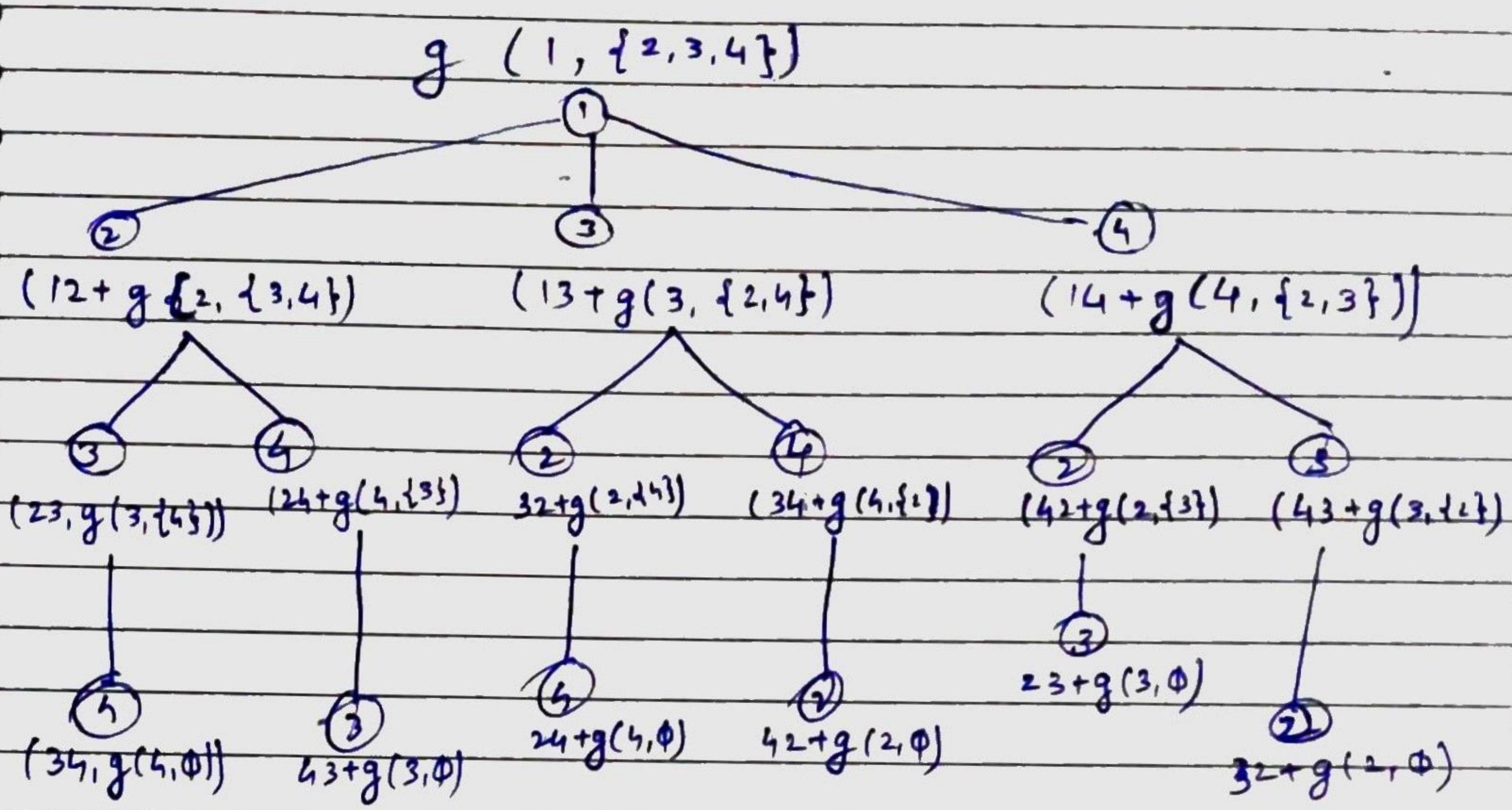
The Recursive formula :-

$$g(i,s) = \min_{k \in s} \{ (ik + g(k, s-\{k\}) \}$$

eg.



$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 10 & 15 & 20 \\ 2 & 5 & 0 & 9 & 10 \\ 3 & 6 & 13 & 0 & 12 \\ 4 & 8 & 8 & 9 & 0 \end{array}$$



g (2, Φ) = 5
g (3, Φ) = 6
g (4, Φ) = 8

g (2, {3}) = 15
g (2, {4}) = 18
g (3, {2}) = 18
g (3, {4}) = 20
g (4, {2}) = 13
g (4, {3}) = 15

g (2, {3,4}) = 25
g (3, {2,4}) = 25
g (4, {2,3}) = 23

g (1, {2,3,4}) = 35 is the minimum cost.

- Datastructure used for TSP using DP is graph

Time Complexity : $= \underbrace{O(n.2^n)}_{\text{Total no. of unique subproblems}} * \underbrace{O(n)}_{\text{Time Taken to solve each problem}}$

$= O(n^2 2^n)$

Space Complexity $= O(n 2^n)$

Using DP we need to construct a table of size $(n-1)\, 2^{n-2}$ to solve

We get better result using DP as compared to Brute Force method to solve TSP .

(Time Complexity using BF method = n!)

Conclusion :

Hence I have implemented and understood the concept of TSP and solved it using Dynamic Programming.