33304

Assignment - 10

* **Title** :- Travelling salesman problem using dynamic programming

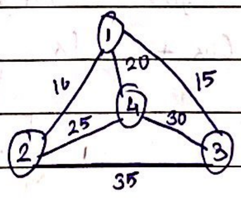
* **Problem statement** :- A traveller needs to visit all the cities from a list, where distances between all the cities are known and each city should be visited once. Find the shortest possible route that he visits each city and returns to the origin city.

* **Objective** :- To implement the solution for travelling salesman problem using dynamic programming

* **Theory** :-

• **What is TSP ?**
Given a set of cities and distance ~~from~~ btw^n every pair of cities, the problem is to find the shortest possible route that visits every city exactly once, and returns to the starting point.

Example →



The TSP tour in above graph is 1-2-4-3-1. The cost is 80.

• **What is Dynamic Programming ?**
Dynamic Programming is mainly an optimization over plain recursion. Whenever we see a recursive solution that has repeated calls for some inputs, we can optimize it using Dynamic Programming. The idea is to simply store the results of subproblems, so that we do not have to re-compute them when needed later.

33304

Let given set of vertices be {1, 2, 3, ... n}. Let us consider 1 as starting and ending point. For every other vertex i (other than 1), we find minimum cost path with 1 as starting point, i as the ending point and all vertices appearing exactly once.

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.

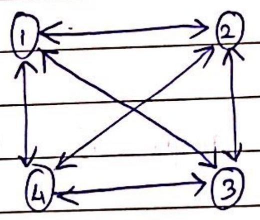
Finally we return the min. of all values [cost (i) + dist (i, 1)] values. To calculate cost using DP we need to have recursive relⁿ.

- **Algorithm →**
  We divide given main problem into multiple sub-problems by constructing recursive tree and obtaining solutions of these sub-problems to use them to solve the main problem collectively.

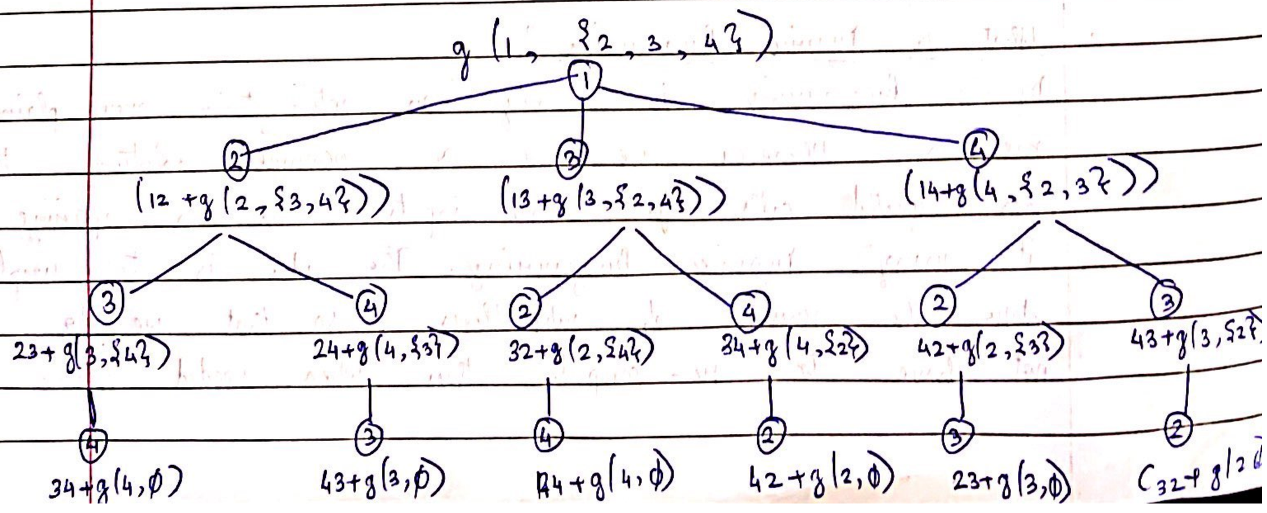
The Recusion formula used for TSP :-

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

Ex :-



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



$\because g(2,\phi) = 5$
$\because g(3,\phi) = 6$
$\because g(4,\phi) = 8$

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

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

$\therefore g(1,\{2,3,4\}) = 35$ is the min. cost.

- The data structure used for TSP DP is graph
- Time complexity →

$$= \underbrace{O(n \cdot 2^n)}_{\text{No. of unique subproblems}} * \underbrace{O(n)}_{\text{Time taken to solve each problem}}$$

$\therefore$ Total time complexity $= O(n^2 2^n)$

Space complexity $= O(n \cdot 2^n)$

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

We get better results using DP as compared to

33304

Brute Force method to solve TSP.

$\downarrow$

The time complexity is $n!$

* Conclusion :- Thus we implemented and understood the concept of TSP and solved it using Dynamic Programming.

Scanned with CamScanner