DAA Lab - Session 4 - Traveling Salesman Problem

**Brute Force:** Solving the travelling salesperson problem with the exhaustive search method.

**Problem Definition:** Find the shortest path from the first vertex traversing all other vertices in a directed graph and returning back to the first vertex. In other words, find a solution to a **Travelling** **Salesperson Problem** using an exhaustive search algorithm.

**Input:** For the TSP, input begins with n (1 ≤ n ≤ 15) of number of vertices of a directed graph. The following n lines to have the cost matrix of the graph with n nonnegative integers (0 ≤ cost ≤ 220) in each line separated by spaces.

**Output:** For the TSP, print the cost of the shortest circuit starting at the first vertex and traversing through all other vertices of the directed graph. Print the time taken (in seconds) in a new line upto 6 decimal places.

**Sample Input:**

5

0 100 125 100 75

100 0 50 75 125

125 50 0 100 125

100 75 100 0 50

75 125 125 50 0

**Sample Output:**

375

0.000000 sec.

ALGORITHM **Travelling Salesman Problem**

//Input: n x n adjacency matrix A. Assumed n > 1.

//Output: Cost of min-cost Hamiltonian circuit.

//getNextPermutation(P[]) returns true with next permn

//in lexicographic order if it exists, false otherwise.

**mincost ← INFINITY**

**Permn[0..n-2] ← [1, 2, 3, …, n-1]** //1st permn.

**do**

**cost ← A[0, Permn[0]]** //1st edge of the ckt

**cost ← cost + A[Permn[n-2], 0]** //last edge

**for i ← 0 to n-3**

**cost ← cost + A[Permn[i], Permn[i+1]]**

**if (cost < mincost) mincost ← cost**

**while(getNextPermutation(Permn[]))**

**return mincost**

**References:** A video of the intro session is available at <https://youtu.be/jN30GM61MBQ> under the title “Introductory session on the lab in Design and Analysis of Algorithms”.

**Practice Problems:**

1. Along the cost of a min-cost Hamiltonian circuit, write a new function a slightly different signature of the function to return the Hamiltonian circuit too.
2. For the TSP, if the graph is not complete (that is, some edges doesn’t exist, which could be represented as -1 cost matrix), we can skip the cases where the the non-existing edge is present so that the algorithm can be made more efficient for an average case. Implement it for such a general graph and compare the execution time with that the typical implementation for the complete graph.
3. Here is a greedy approach to attempt solving the TSP. Starting from the origin vertex, for the first n-1 hops, select the lowest cost edge going out of the current vertex which is not yet visited. The last edge obviously has to come back to the origin vertex. It’s a Hamilton circuit for sure, but not sure whether it’s the min-cost one. Compare the cost of such a circuit with the exhaustive search solution implemented earlier. One counterexample is good enough to prove that the above greedy approach does not solve the TSP problem with the optimal solution, but may give fairly acceptable near-optimal solution, which may be good enough in practical cases when the exhaustive search algorithm takes unreasonable amount of time to find the optimal solution.