

Assignment 8

```
#include <iostream>

#include <climits>

using namespace std;

int** distanceMatrix; // Distance matrix between cities
int** dpTable;        // DP table to store the minimum cost for subsets of cities
int numCities;        // Number of cities

// Function to solve TSP using Dynamic Programming
int findMinCost(int currentCity, int visitedMask) {
    // Base case: All cities visited, return to the starting city
    if (visitedMask == (1 << numCities) - 1)
        return distanceMatrix[currentCity][0]; // Return to starting city (0)

    // If the result is already computed, return it
    if (dpTable[currentCity][visitedMask] != -1)
        return dpTable[currentCity][visitedMask];

    int minCost = INT_MAX; // Initialize minimum cost as infinity

    // Try to visit all unvisited cities
    for (int nextCity = 0; nextCity < numCities; nextCity++) {
        // If the city is unvisited
        if ((visitedMask & (1 << nextCity)) == 0) {
            // Calculate the cost of visiting the next city
            int cost = distanceMatrix[currentCity][nextCity] +
                findMinCost(nextCity, visitedMask | (1 << nextCity)); // Recur for the next city
            minCost = min(minCost, cost); // Update minimum cost
        }
    }
}
```

```
    return dpTable[currentCity][visitedMask] = minCost; // Store and return the result
}
```

```
// Function to reconstruct the path taken
```

```
void printPath(int currentCity, int visitedMask) {
    cout << currentCity + 1; // Print the current city (1-based indexing)

    if (visitedMask == (1 << numCities) - 1) {
        cout << " -> 1"; // Return to the starting city
        return;
    }
```

```
int nextCity; // Variable to store the next city to visit
int minCost = INT_MAX; // Initialize minimum cost as infinity
```

```
// Find the next city to visit
```

```
for (int city = 0; city < numCities; city++) {
    // If the city is unvisited
    if ((visitedMask & (1 << city)) == 0) {
        int cost = distanceMatrix[currentCity][city] +
            dpTable[city][visitedMask | (1 << city)];
        if (cost < minCost) {
            minCost = cost;
            nextCity = city; // Update the next city to visit
        }
    }
}
```

```
// Recursively print the path
```

```
printPath(nextCity, visitedMask | (1 << nextCity));
```

```
}
```

```
int main() {
```

```
    // Input number of cities
```

```
    cout << "Enter the number of cities: ";
```

```
    cin >> numCities;
```

```
    // Allocate memory for distance matrix and DP table
```

```
    distanceMatrix = new int*[numCities];
```

```
    dpTable = new int*[numCities];
```

```
    for (int i = 0; i < numCities; i++) {
```

```
        distanceMatrix[i] = new int[numCities];
```

```
        dpTable[i] = new int[1 << numCities]; // 2^numCities
```

```
    }
```

```
    // Input the distance matrix
```

```
    cout << "Enter the distance matrix:\n";
```

```
    for (int i = 0; i < numCities; i++) {
```

```
        for (int j = 0; j < numCities; j++) {
```

```
            cin >> distanceMatrix[i][j]; // Read distances
```

```
        }
```

```
    }
```

```
    // Initialize the DP table with -1
```

```
    for (int i = 0; i < numCities; i++)
```

```
        for (int j = 0; j < (1 << numCities); j++)
```

```
            dpTable[i][j] = -1;
```

```
    // Calculate the minimum cost
```

```
    int minCost = findMinCost(0, 1); // Start from city 0 with mask indicating city 0 is visited
```

```

// Output the results

cout << "\nMinimum cost of the Traveling Salesman Problem: " << minCost << endl;

cout << "Path taken: ";

printPath(0, 1); // Reconstruct and print the path

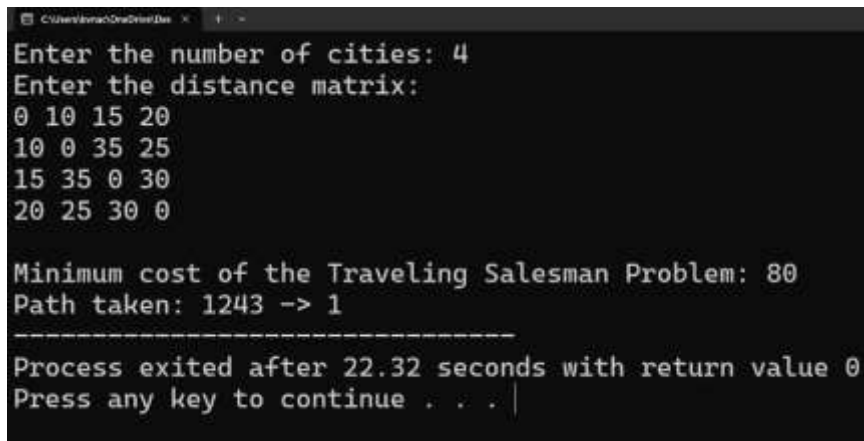

// Free allocated memory
for (int i = 0; i < numCities; i++) {
    delete[] distanceMatrix[i];
    delete[] dpTable[i];
}

delete[] distanceMatrix;
delete[] dpTable;

return 0;
}

```

Output:



```

C:\Users\mac\OneDrive\Des > .\TSP.exe
Enter the number of cities: 4
Enter the distance matrix:
0 10 15 20
10 0 35 25
15 35 0 30
20 25 30 0

Minimum cost of the Traveling Salesman Problem: 80
Path taken: 1243 -> 1
-----
Process exited after 22.32 seconds with return value 0
Press any key to continue . . .

```