Name-Yash A Gunjal

Div-A

PRN-202201040106

Roll No-39

```cpp
#include <bits/stdc++.h>
using namespace std;

#define INF 1e9
int N;
vector<vector<int>> dist;
vector<vector<int>> dp;

// Function to count the number of set bits (visited cities)
int countSetBits(int n) {
    return __builtin_popcount(n); // Built-in function for counting bits in GCC/Clang
}

// Function to solve TSP
int tsp(int mask, int pos) {
    int s = countSetBits(mask); // Number of visited cities

    if (mask == (1 << N) - 1)
        return dist[pos][0];

    if (dp[mask][pos] != -1)
        return dp[mask][pos];

    int ans = INF;
    for (int city = 0; city < N; city++) {
        if ((mask & (1 << city)) == 0) {
            int newAns = dist[pos][city] + tsp(mask | (1 << city), city);
            ans = min(ans, newAns);

            cout << "Cost(" << pos + 1 << ", {";
            for (int i = 0; i < N; i++) {
                if (mask & (1 << i)) cout << i + 1 << " ";
            }
            cout << city + 1 << "}, 1) = " << newAns << endl;

            if (s == 2 || s == 3) {
                cout << "Min cost for subset of size " << s << ": " << ans << endl;
            }
        }
    }
    return dp[mask][pos] = ans;
}

// Reconstructs the path
void findPath() {
    int mask = 1, pos = 0;
    vector<int> path = {1};

    while (mask != (1 << N) - 1) {
```

```
        int bestCity = -1, bestCost = INF;
        for (int city = 0; city < N; city++) {
            if ((mask & (1 << city)) == 0) {
                int newCost = dist[pos][city] + dp[mask | (1 << city)][city];
                if (newCost < bestCost) {
                    bestCost = newCost;
                    bestCity = city;
                }
            }
        }
        mask |= (1 << bestCity);
        pos = bestCity;
        path.push_back(bestCity + 1);
    }

    path.push_back(1);
    cout << "\nOptimal Path: ";
    for (size_t i = 0; i < path.size(); i++) {
        cout << path[i];
        if (i != path.size() - 1) cout << " -> ";
    }
    cout << endl;
}

int main() {
    cout << "Enter the number of cities: ";
    cin >> N;

    dist.resize(N, vector<int>(N));
    dp.assign(1 << N, vector<int>(N, -1));

    cout << "Enter the distance matrix (NxN):\n";
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cin >> dist[i][j];
        }
    }

    int minCost = tsp(1, 0);
    cout << "\nMinimum Cost: " << minCost << endl;
    findPath();
    return 0;
}
```

OutPut :-

```
Active code page: 65001

D:\TYBtech\DAA>cd "d:\TYBtech\DAA\output"

d:\TYBtech\DAA\output>.\"TSP by DP.exe"
Enter the number of cities: 4
Enter the distance matrix (NxN):
0 10 15 20
5 0 9 10
6 13 0 12
8 8 9 0
Cost(3, {1 2 3 4}, 1) = 20
Min cost for subset of size 3: 20
Cost(2, {1 2 3}, 1) = 29
Min cost for subset of size 2: 29
Cost(4, {1 2 4 3}, 1) = 15
Min cost for subset of size 3: 15
Cost(2, {1 2 4}, 1) = 25
Min cost for subset of size 2: 25
Cost(1, {1 2}, 1) = 35
Cost(2, {1 2 3 4}, 1) = 18
Min cost for subset of size 3: 18
Cost(3, {1 3 2}, 1) = 31
Min cost for subset of size 2: 31
Cost(4, {1 3 4 2}, 1) = 13
Min cost for subset of size 3: 13
Cost(3, {1 3 4}, 1) = 25
Min cost for subset of size 2: 25
Cost(1, {1 3}, 1) = 40
Cost(2, {1 2 4 3}, 1) = 15
Min cost for subset of size 3: 15
Cost(4, {1 4 2}, 1) = 23
Min cost for subset of size 2: 23
Cost(3, {1 3 4 2}, 1) = 18
Min cost for subset of size 3: 18
Cost(4, {1 4 3}, 1) = 27
Min cost for subset of size 2: 23
Cost(1, {1 4}, 1) = 43

Minimum Cost: 35

Optimal Path: 1 -> 2 -> 4 -> 3 -> 1

d:\TYBtech\DAA\output>
```