**TRAVELLING SALESMAN PROBLEM**

#include <iostream>

#include <vector>

#include <algorithm>

#include <climits>

using namespace std;

#define V 4 // Number of vertices

// Function to solve the TSP problem using dynamic programming

int tsp(int graph[][V], int mask, int pos, vector<vector<int> >& dp) {

if (mask == ((1 << V) - 1)) // If all cities have been visited

return graph[pos][0]; // Return to the starting city (0)

// If the subproblem has already been solved

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

return dp[mask][pos];

int ans = INT\_MAX;

// Try to visit all cities not yet visited

for (int city = 0; city < V; ++city) {

if ((mask & (1 << city)) == 0) { // If city not visited

int newAns = graph[pos][city] + tsp(graph, mask | (1 << city), city, dp);

ans = min(ans, newAns);

}

}

return dp[mask][pos] = ans;

}

// Function to initialize dp array and call tsp function

int tsp(int graph[][V]) {

vector<vector<int> > dp(1 << V, vector<int>(V, -1));

return tsp(graph, 1, 0, dp);

}

int main() {

// Example graph (4x4 adjacency matrix)

int graph[][V] = {

{0, 10, 15, 20},

{10, 0, 35, 25},

{15, 35, 0, 30},

{20, 25, 30, 0}

};

// Output the minimum cost of the TSP

cout << "Minimum cost of TSP: " << tsp(graph) << endl;

return 0;

}