**ASSIGNMENT 10**

import java.util.Arrays;

import java.util.Scanner;

class TSP {

private int n;

private int[][] dist;

private int minCost = Integer.MAX\_VALUE;

private int[] bestPath;

public TSP(int[][] dist) {

this.dist = dist;

this.n = dist.length;

this.bestPath = new int[n + 1];

}

public void solve() {

boolean[] visited = new boolean[n];

visited[0] = true;

findPath(0, 0, 1, visited, new int[n]);

printResult();

}

private void findPath(int currentCity, int cost, int level, boolean[] visited, int[] currentPath) {

currentPath[level - 1] = currentCity;

if (level == n) {

cost += dist[currentCity][0];

if (cost < minCost) {

minCost = cost;

System.arraycopy(currentPath, 0, bestPath, 0, n);

bestPath[n] = 0;

}

return;

}

int lowerBound = cost + calculateLowerBound(visited, currentCity);

if (lowerBound >= minCost) {

return;

}

for (int i = 0; i < n; i++) {

if (!visited[i]) {

visited[i] = true;

findPath(i, cost + dist[currentCity][i], level + 1, visited, currentPath);

visited[i] = false;

}

}

}

private int calculateLowerBound(boolean[] visited, int currentCity) {

int bound = 0;

for (int i = 0; i < n; i++) {

if (!visited[i]) {

int minEdge = Integer.MAX\_VALUE;

for (int j = 0; j < n; j++) {

if (i != j && !visited[j]) {

minEdge = Math.min(minEdge, dist[i][j]);

}

}

if (minEdge != Integer.MAX\_VALUE) {

bound += minEdge;

}

}

}

return bound;

}

private void printResult() {

System.out.println("Minimum cost: " + minCost);

System.out.print("Optimal path: ");

for (int city : bestPath) {

System.out.print(city + " ");

}

System.out.println();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of cities: ");

int n = scanner.nextInt();

int[][] dist = new int[n][n];

System.out.println("Enter the distance matrix:");

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

dist[i][j] = scanner.nextInt();

}

}

new TSP(dist).solve();

scanner.close();

}

}

OUTPUT:

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: 80

Optimal path: 0 1 3 2 0