**Design and Analysis of Algorithms**

**UE23CS241B**

**4th Semester, Academic Year 2023**

Date: 20/02/2025

|  |  |  |
| --- | --- | --- |
| Name: Pranav Hemanth | SRN: PES1UG23CS433 | Section: G |

TITLE:

**PROBLEM 1 – THE QUICK COURIER**

**QUESTION- IN THE TOWN OF GRAPHVILLE, THERE IS A QUICK COURIER NAMED LEXI WHO NEEDS TO DELIVER PACKAGES TO SEVERAL BUSINESSES LOCATED IN VARIOUS DISTRICTS. GRAPHVILLE IS WELL-CONNECTED, BUT LEXI WANTS TO ENSURE THE ROUTE TAKEN IS NOT ONLY EFFICIENT BUT ALSO PREDICTABLE IN ORDER, SO HER DISPATCH TEAM CAN EASILY TRACK HER PROGRESS.**

**LEXI DECIDES TO ALWAYS TAKE THE LEXICOGRAPHICALLY SMALLEST PATH THAT VISITS EACH DISTRICT EXACTLY ONCE AND RETURNS TO THE STARTING DISTRICT. AS A BUDDING PROGRAMMER, YOU ARE TASKED WITH HELPING LEXI FIND THIS PATH USING A PROGRAM.**

**INPUT FORMAT- THE FIRST LINE CONTAINS AN INTEGER N (2 ≤ N ≤ 10), THE NUMBER OF DISTRICTS.**

**EACH OF THE NEXT N LINES CONTAINS N INTEGERS, WHERE THE J-TH INTEGER IN THE I-TH LINE REPRESENTS THE TRAVEL COST FROM DISTRICT I TO DISTRICT J. IF THE NUMBER IS -1, IT MEANS THAT THE PATH BETWEEN THE TWO DISTRICTS DOES NOT EXIST.**

**OUTPUT FORMAT- OUTPUT THE LEXICOGRAPHICALLY SMALLEST PATH (AS A SEQUENCE OF DISTRICT INDICES) THAT VISITS EACH DISTRICT EXACTLY ONCE AND RETURNS TO THE STARTING DISTRICT. IF THE PATH DOES NOT EXIST, RETURN -1**

**Code:**

#include <stdio.h>

#include <stdlib.h>

void swap(int \*a, int \*b) {

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

int next(int \*arr, int n) {

    int i = n - 2;

    while (i >= 0 && arr[i] >= arr[i + 1]){

        i--;

    }

    if (i < 0){

        return 0;

    }

    int j = n - 1;

    while (arr[j] <= arr[i])

        j--;

    swap(&arr[i], &arr[j]);

    for (int l = i + 1, r = n - 1; l < r; l++, r--) {

        swap(&arr[l], &arr[r]);

    }

    return 1;

}

int main() {

    int n, cost[10][10];

    int visited[10];

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            scanf("%d", &cost[i][j]);

        }

    }

    int districts[10], path[11];

    for (int i = 0; i < n - 1; i++) {

        districts[i] = i + 1;

    }

    int found = 0;

    int min = 0;

    do {

        int valid = 1, total = 0;

        int prev = 0;

        for (int i = 0; i < n - 1; i++) {

            if (cost[prev][districts[i]] == -1) {

                valid = 0;

                break;

            }

            total += cost[prev][districts[i]];

            prev = districts[i];

        }

        if (valid && cost[prev][0] != -1) {

            total += cost[prev][0];

            if (!found || total < min) {

                found = 1;

                min = total;

                path[0] = 0;

                for (int i = 0; i < n - 1; i++) {

                    path[i + 1] = districts[i];

                }

                path[n] = 0;

            }

        }

    } while (next(districts, n - 1));

    if (!found) {

        printf("Path doesn't exist\n");

    } else {

        printf("Minimum Cost: %d\nOptimal Path: ", min);

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

            printf("%d ", path[i]);

        }

        printf("\n");

    }

    return 0;

}

**Output:**









