**Ramdeobaba University, Nagpur**

**Department of Computer Science and Engineering**

**Session: 2024-2025**

# Design and Analysis of Algorithms Lab III Semester

**Name: Jatin khatnani Roll No.: A8-B1-04**

**PRACTICAL NO. 7**

**Aim:** Implement Hamiltonian Cycle using Backtracking.

**Problem Statement:**

The Smart City Transportation Department is designing a night-patrol route for security vehicles.

Each area of the city is represented as a vertex in a graph, and a road between two areas is represented as an edge.

The goal is to find a route that starts from the main headquarters (Area A), visits each area exactly once, and returns back to the headquarters — forming a Hamiltonian Cycle.

If such a route is not possible, display a suitable message.

1) Adjacency Matrix

# A B C D E

1. 0 1 1 0 1
2. 1 0 1 1 0
3. 1 1 0 1 0
4. 0 1 1 0 1
5. 1 0 0 1 0

1) Adjacency Matrix

# T M S H C

**T** 0 1 1 0 1

**M** 1 0 1 1 0

**S** 1 1 0 1 1

**H** 0 1 1 0 1

**C** 1 0 1 1 0

**CODE:**

#include <stdio.h>

void nextValue(int k, int n, int G[n+1][n+1], int x[n+1]);

void hamiltonian(int k, int n, int G[n+1][n+1], int x[n+1]) {

    while (1) {

        nextValue(k, n, G, x);

        if (x[k] == 0)

            return;

        if (k == n) {

            for (int i = 1; i <= n; i++)

                printf("%d ", x[i]);

            printf("%d\n", x[1]);

        } else {

            hamiltonian(k + 1, n, G, x);

        }

    }

}

void nextValue(int k, int n, int G[n+1][n+1], int x[n+1]) {

    int j;

    while (1) {

        x[k] = (x[k] + 1) % (n + 1);

        if (x[k] == 0)

            return;

        if (G[x[k - 1]][x[k]] != 0) {

            for (j = 1; j <= k - 1; j++)

                if (x[j] == x[k])

                    break;

            if (j == k) {

                if ((k < n) || ((k == n) && G[x[n]][x[1]] != 0))

                    return;

            }

        }

    }

}

int main() {

    int n = 5;

    int G1[6][6] = {

        {0,0,0,0,0,0},

        {0,0,1,1,0,1},

        {0,1,0,1,1,0},

        {0,1,1,0,1,0},

        {0,0,1,1,0,1},

        {0,1,0,0,1,0}

    };

    int G2[6][6] = {

        {0,0,0,0,0,0},

        {0,0,1,1,0,1},

        {0,1,0,1,1,0},

        {0,1,1,0,1,1},

        {0,0,1,1,0,1},

        {0,1,0,1,1,0}

    };

    int x[6];

    for (int i = 1; i <= n; i++)

        x[i] = 0;

    x[1] = 1;

    printf("For Graph 1 (Areas A-E):\n");

    hamiltonian(2, n, G1, x);

    for (int i = 1; i <= n; i++)

        x[i] = 0;

    x[1] = 1;

    printf("\nFor Graph 2 (T, M, S, H, C):\n");

    hamiltonian(2, n, G2, x);

    return 0;

}

O/P:

