**DAA PRACTICAL NO. 7**

**NAME: DIVYA SONTAKKE**

**SECTION: A4\_B1 ROLL NO.: 11**

**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**

**A** 0 1 1 0 1

**B** 1 0 1 1 0

**C** 1 1 0 1 0

**D** 0 1 1 0 1

**E** 1 0 0 1 0

2) 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>

#define MAX 10

int n, x[MAX], G[MAX][MAX];

void NextValue(int k){

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; j++)

if (x[j] == x[k])

break;

if (j == k){

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

return;

}

}

}

}

void Hamiltonian(int k){

int i;

while (1){

NextValue(k);

if (x[k] == 0)

return;

if (k == n){

printf("\nHamiltonian Cycle: ");

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

printf("%d ", x[i]);

printf("%d", x[1]);

}

else

Hamiltonian(k + 1);

}

}

void findHamiltonian(int vertices, int graph[vertices][vertices]){

int i, j;

n = vertices;

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

for (j = 1; j <= n; j++)

G[i][j] = graph[i - 1][j - 1];

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

x[i] = 0;

x[1] = 1;

printf("Finding Hamiltonian Cycles for %d vertices\n", n);

Hamiltonian(2);

printf("\n");

}

int main(){

int graph1[5][5] = {

{0, 1, 1, 0, 1},

{1, 0, 1, 1, 0},

{1, 1, 0, 1, 0},

{0, 1, 1, 0, 1},

{1, 0, 0, 1, 0}

};

int graph2[5][5] = {

{0, 1, 1, 0, 1},

{1, 0, 1, 1, 0},

{1, 1, 0, 1, 1},

{0, 1, 1, 0, 1},

{1, 0, 1, 1, 0}

};

printf("City Areas (A–E) \n");

findHamiltonian(5, graph1);

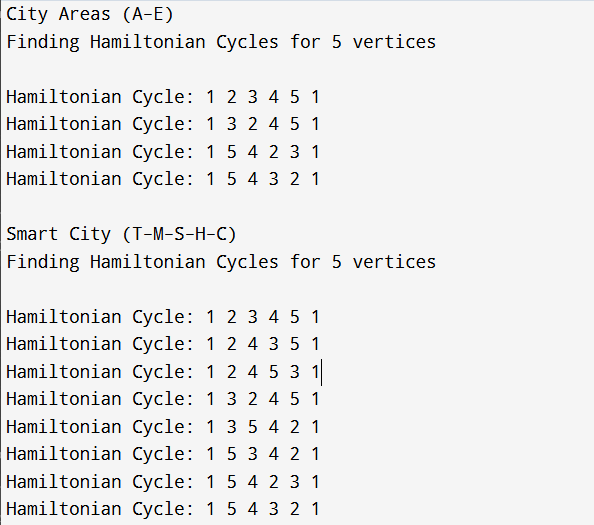
printf("\nSmart City (T–M–S–H–C)\n");

findHamiltonian(5, graph2);

return 0;

}

**OUTPUT**

****