Practical 7 (DAA)

Name : Mohammad Ejaz Mehendi

Section : A3-B3

Roll no : 48

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

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 :

def isSafe(v, graph, path, pos):

if graph[path[pos - 1]][v] == 0:

return False

if v in path:

return False

return True

def hamiltonianCycleUtil(graph, path, pos, V):

if pos == V:

if graph[path[pos - 1]][path[0]] == 1:

return True

else:

return False

for v in range(1, V):

if isSafe(v, graph, path, pos):

path[pos] = v

if hamiltonianCycleUtil(graph, path, pos + 1, V):

return True

path[pos] = -1

return False

def hamiltonianCycle(graph):

V = len(graph)

path = [-1] \* V

path[0] = 0

if not hamiltonianCycleUtil(graph, path, 1, V):

print("No Hamiltonian Cycle exists")

return False

print("Hamiltonian Cycle found:")

for vertex in path:

print(chr(ord('A') + vertex), end=" -> ")

print(chr(ord('A') + path[0]))

return True

graph1 = [

[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]

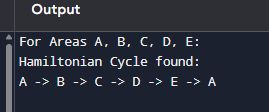
]

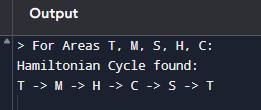
print("For Areas A, B, C, D, E:")

hamiltonianCycle(graph1)

print()

Output :





Screenshot :

