

DAA Practical 7

Name: Mohammad Anas Ajaz

Class: A4(B4)

Roll No. 61

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.

Code:

```
def checking(v, graph, path, pos):
    if graph[path[pos - 1]][v] == 0:
        return False
    if v in path:
        return False
    return True

def hamiltonian_cycle_util(graph, path, pos):
    n = len(graph)
    if pos == n:
        if graph[path[pos - 1]][path[0]] == 1:
            return True
        return False

    for v in range(1, n):
```

```

        if checking(v, graph, path, pos):
            path[pos] = v
            if hamiltonian_cycle_util(graph, path, pos + 1):
                return True
            path[pos] = -1
    return False

def hamiltonian_cycle(graph, vertices):
    n = len(graph)
    path = [-1] * n
    path[0] = 0

    if not hamiltonian_cycle_util(graph, path, 1):
        print("No Hamiltonian Cycle exists")
        return

    print("Hamiltonian Cycle exists:")
    for vertex in path:
        print(vertices[vertex], end=" -> ")
    print(vertices[path[0]])

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]
]

vertices1 = ['A', 'B', 'C', 'D', 'E']
print("Example 1:")
hamiltonian_cycle(graph1, vertices1)
print()

graph2 = [
    [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]
]

```

```
vertices2 = ['T', 'M', 'S', 'H', 'C']
print("Example 2:")
hamiltonian_cycle(graph2, vertices2)
```

Output:

Example 1:

Hamiltonian Cycle exists:

A -> B -> C -> D -> E -> A

Example 2:

Hamiltonian Cycle exists:

T -> M -> S -> H -> C -> T

GFG Competitive Coding :

The screenshot displays the GeeksforGeeks website interface for a problem titled "Hamiltonian Path". The page is divided into several sections:

- Header:** Includes the GeeksforGeeks logo, a search bar, and navigation links for Courses, Tutorials, Practice, and Jobs.
- Problem Details:**
 - Output Window:** Shows "Compilation Results" for a custom input by "Y.O.G.I. (AI Bot)".
 - Problem Solved Successfully:** A green checkmark indicates the problem is solved.
 - Test Cases Passed:** 52 / 52.
 - Attempts:** 1 / 1.
 - Accuracy:** 100%.
 - Points Scored:** 4 / 4.
 - Time Taken:** 0.04.
 - Your Total Score:** 12 (with an upward arrow).
- Solve Next:** A section with buttons for "Number of Provinces" and "Number of Distinct Islands".
- Code Editor:**
 - Language: Python3.
 - Start Timer: A button to start a timer.
 - Code: A Python solution using Depth-First Search (DFS) to find a Hamiltonian path. The code defines a graph, a DFS function, and a main function to check for a path.
- Footer:** Includes a "Custom Input" button, a "Compile & Run" button, and a "Submit" button.