

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:

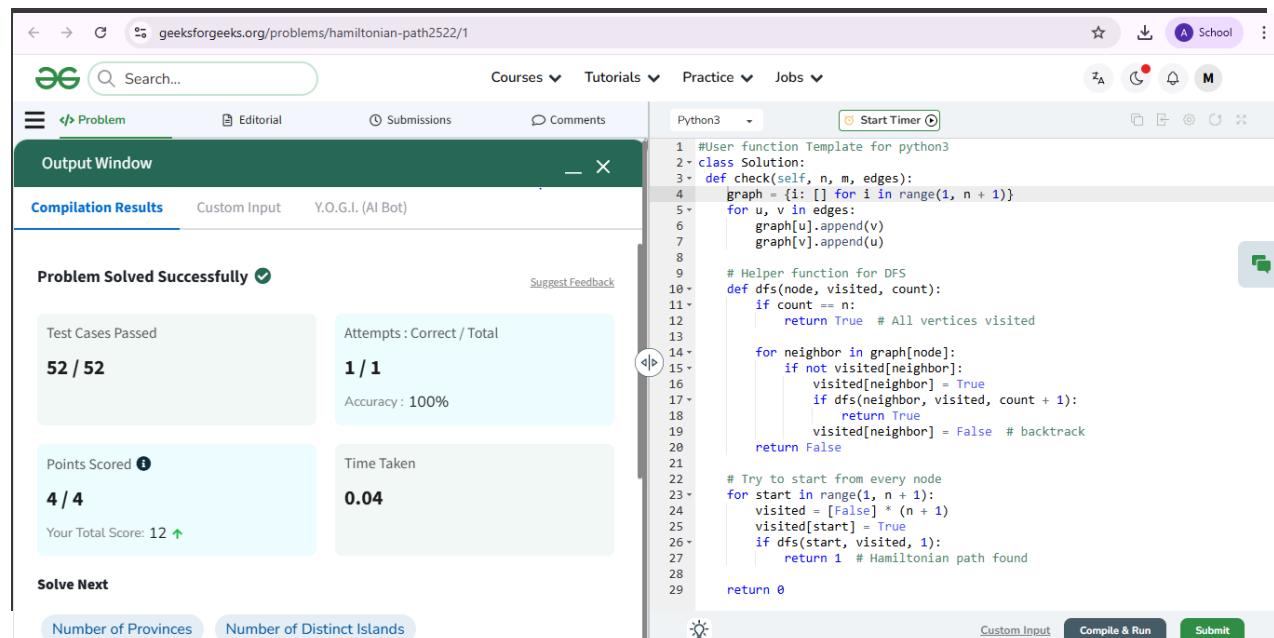
Example 1:

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 shows a successful submission on the GeeksforGeeks Competitive Coding platform for the "Hamiltonian Path" problem. The submission was made in Python3 and passed all test cases.

Compilation Results:

- Test Cases Passed: 52 / 52
- Attempts : Correct / Total: 1 / 1
- Accuracy: 100%
- Points Scored: 4 / 4
- Your Total Score: 12

Code Snippet:

```

1 #User function Template for python3
2 class Solution:
3     def check(self, n, m, edges):
4         graph = [i: [] for i in range(1, n + 1)]
5         for u, v in edges:
6             graph[u].append(v)
7             graph[v].append(u)
8
9         # Helper function for DFS
10        def dfs(node, visited, count):
11            if count == n:
12                return True # All vertices visited
13
14            for neighbor in graph[node]:
15                if not visited[neighbor]:
16                    visited[neighbor] = True
17                    if dfs(neighbor, visited, count + 1):
18                        return True
19                    visited[neighbor] = False # backtrack
20
21        # Try to start from every node
22        for start in range(1, n + 1):
23            visited = [False] * (n + 1)
24            visited[start] = True
25            if dfs(start, visited, 1):
26                return 1 # Hamiltonian path found
27
28
29        return 0

```

Platform UI Elements:

- Search bar: Search...
- Navigation: Courses, Tutorials, Practice, Jobs
- User Profile: School
- Code Editor: Python3, Start Timer
- Output Window: Problem Solved Successfully
- Submission Status: Compilation Results
- Feedback: Suggest Feedback
- Performance Metrics: Test Cases Passed, Attempts, Accuracy, Points Scored, Time Taken
- Buttons: Custom Input, Y.O.G.I. (AI Bot), Solve Next, Number of Provinces, Number of Distinct Islands, Compile & Run, Submit