

**Name : Mohammad Anas Ajaz**

**Roll No. 61**

**Batch : A4(B4)**

**Practical No 7**

Competitive Coding Link:

<https://www.geeksforgeeks.org/problems/hamiltonian-path2522/1>

code :

class Solution:

```
def check(self, n, m, edges):  
  
    graph = {i: [] for i in range(1, n + 1)}  
  
    for u, v in edges:  
        graph[u].append(v)  
        graph[v].append(u)
```

```
def dfs(node, visited, count):
```

```
    if count == n:
```

```
        return True
```

```
    for neighbor in graph[node]:
```

```
        if not visited[neighbor]:
```

```
            visited[neighbor] = True
```

```
            if dfs(neighbor, visited, count + 1):
```

```
                return True
```

```
            visited[neighbor] = False
```

```
return False
```

```
for start in range(1, n + 1):
```

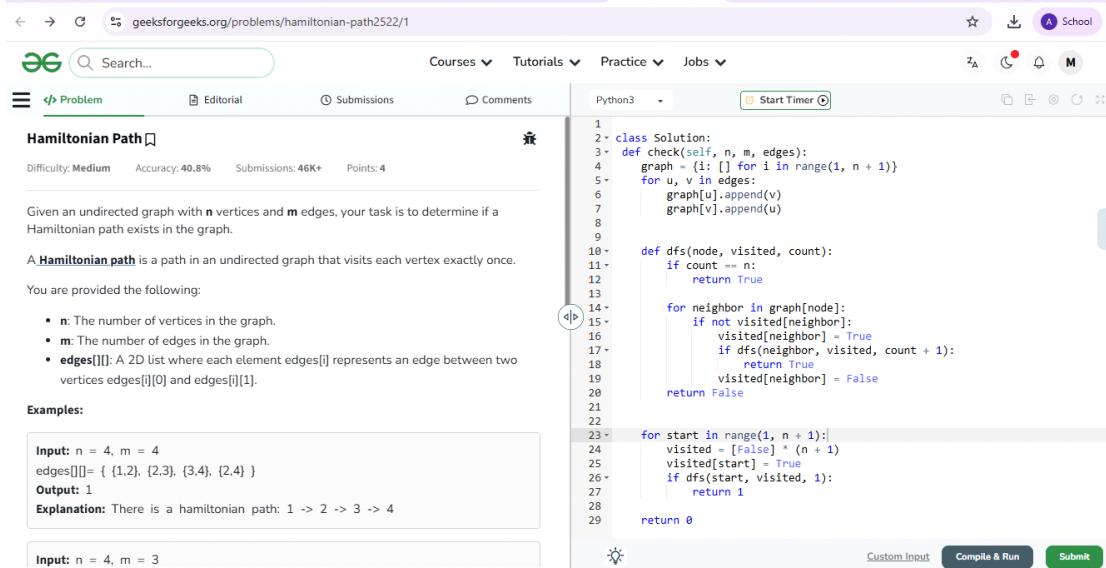
```
    visited = [False] * (n + 1)
```

```
    visited[start] = True
```

```
    if dfs(start, visited, 1):
```

```
        return 1
```

```
return 0
```



The screenshot shows a problem page from geeksforgeeks.org. The title is "Hamiltonian Path". The difficulty is Medium, accuracy is 40.8%, submissions are 46K+, and points are 4. The problem statement asks to determine if a Hamiltonian path exists in an undirected graph with n vertices and m edges. A Hamiltonian path is defined as a path that visits each vertex exactly once. The user is provided with the following:

- n: The number of vertices in the graph.
- m: The number of edges in the graph.
- edges[][]: A 2D list where each element edges[i] represents an edge between two vertices edges[i][0] and edges[i][1].

Examples:

```
Input: n = 4, m = 4
edges[][]= { {1,2}, {2,3}, {3,4}, {2,4} }
Output: 1
Explanation: There is a hamiltonian path: 1 -> 2 -> 3 -> 4
```

The code editor shows the following Python3 code:

```
1  class Solution:
2      def check(self, n, m, edges):
3          graph = {i: [] for i in range(1, n + 1)}
4          for u, v in edges:
5              graph[u].append(v)
6              graph[v].append(u)
7
8
9          def dfs(node, visited, count):
10             if count == n:
11                 return True
12
13             for neighbor in graph[node]:
14                 if not visited[neighbor]:
15                     visited[neighbor] = True
16                     if dfs(neighbor, visited, count + 1):
17                         return True
18                     visited[neighbor] = False
19
20     return False
21
22
23     for start in range(1, n + 1):
24         visited = [False] * (n + 1)
25         visited[start] = True
26         if dfs(start, visited, 1):
27             return 1
28
29     return 0
```

At the bottom of the code editor, there are buttons for "Custom Input", "Compile & Run", and "Submit".

Search...

Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓ Suggest Feedback

Test Cases Passed 52 / 52 Attempts : Correct / Total 3 / 3 Accuracy : 100%

Time Taken 0.03

You get marks only for the first correct submission if you solve the problem without viewing

Python3 Start Timer

```
1
2- class Solution:
3-     def check(self, n, m, edges):
4-         graph = [{} 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
10-    def dfs(node, visited, count):
11-        if count == n:
12-            return True
13-
14-        for neighbor in graph[node]:
15-            if not visited[neighbor]:
16-                visited[neighbor] = True
17-                if self.dfs(neighbor, visited, count + 1):
18-                    return True
19-                visited[neighbor] = False
20-
21-
22-
23-    for start in range(1, n + 1):
24-        visited = [False] * (n + 1)
25-        visited[start] = True
26-        if self.dfs(start, visited, 1):
27-            return 1
28-
29-    return 0
```