

**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 the GeeksforGeeks website interface for the 'Hamiltonian Path' problem. The page layout includes a header with navigation links (Courses, Tutorials, Practice, Jobs), a search bar, and a sidebar with problem details. The main content area displays the problem description, examples, and a code editor with a Python3 solution.

**Hamiltonian Path** 🔖

Difficulty: Medium Accuracy: 40.8% Submissions: 46K+ Points: 4

Given an undirected graph with **n** vertices and **m** edges, your task is to determine if a Hamiltonian path exists in the graph.

A **Hamiltonian path** is a path in an undirected graph that visits each vertex exactly once.

You are provided 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

**Input:** n = 4, m = 3

```
1 class Solution:
2     def check(self, n, m, edges):
3         graph = {}
4         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 dfs(neighbor, visited, count + 1):
18                    return True
19                visited[neighbor] = False
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
```

Custom Input Compile & Run Submit



Search...

CoursesTutorialsPracticeJobs

ProblemEditorialSubmissionsComments

Python3Start Timer

Output Window

Compilation ResultsCustom InputY.O.G.I. (AI Bot)

Problem Solved Successfully

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

```
1
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
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 dfs(neighbor, visited, count + 1):
18                        return True
19                    visited[neighbor] = False
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
```