Name : Mohammad Anas Ajaz

Roll NO. 61

Batch A4(B4)

Practical No. 8 : DAA Lab

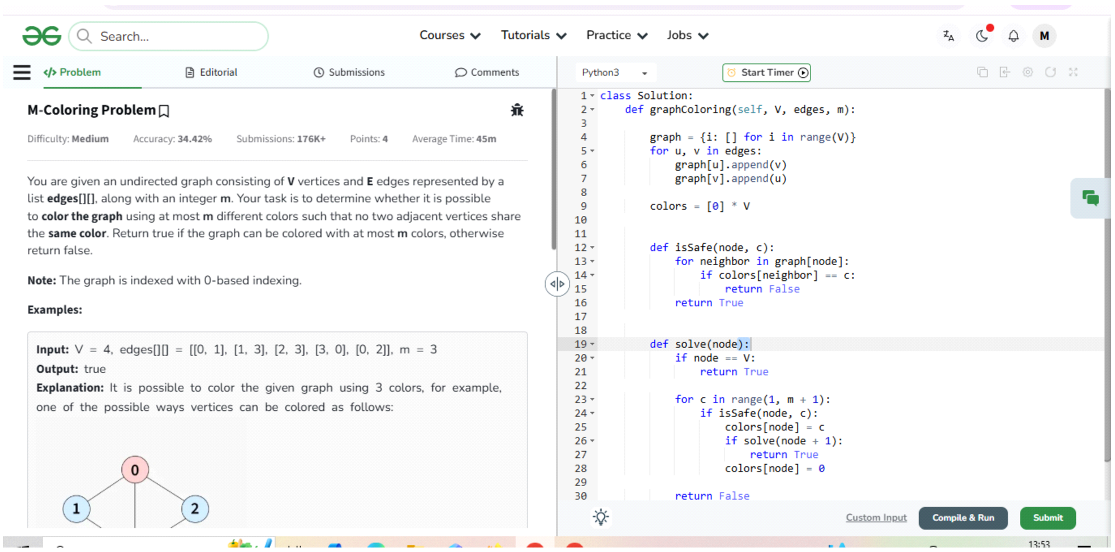
Competitive Coding Link:

https://www.geeksforgeeks.org/problems/m-coloring-problem-1587115620/1

Code :

```
class Solution:
    def graphColoring(self, V, edges, m):


        graph = {i: [] for i in range(V)}
        for u, v in edges:
            graph[u].append(v)
            graph[v].append(u)


        colors = [0] * V




        def isSafe(node, c):
            for neighbor in graph[node]:
                if colors[neighbor] == c:
                    return False
            return True




        def solve(node):
```

```
if node == V:
    return True

for c in range(1, m + 1):
    if isSafe(node, c):
        colors[node] = c
        if solve(node + 1):
            return True
        colors[node] = 0

return False
```



M-Coloring Problem

Difficulty: **Medium** Accuracy: **34.42%** Submissions: **176K+** Points: **4** Average Time: **45m**

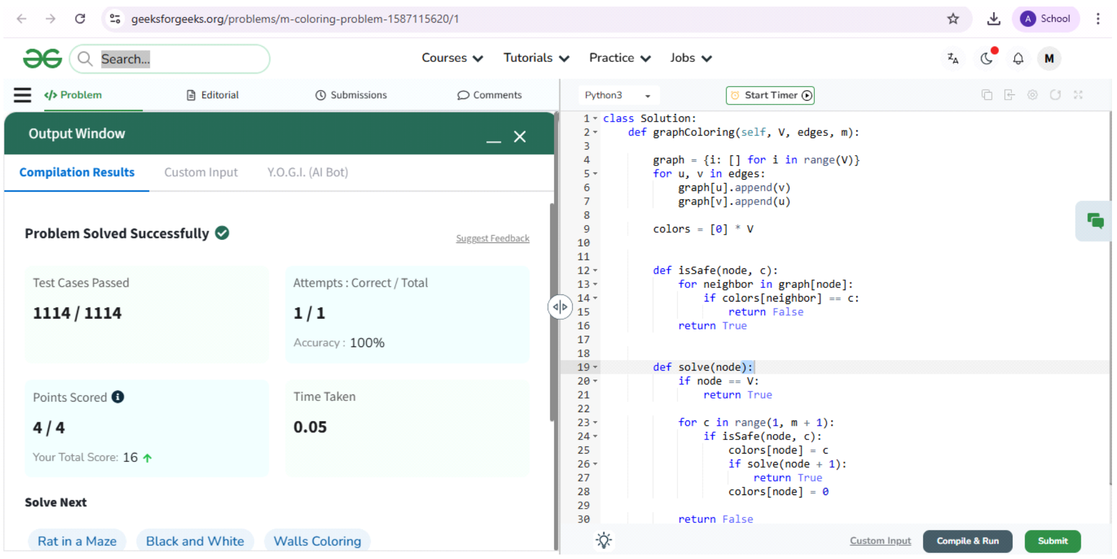
You are given an undirected graph consisting of **V** vertices and **E** edges represented by a list **edges[][]**, along with an integer **m**. Your task is to determine whether it is possible to **color the graph** using at most **m** different colors such that no two adjacent vertices share the **same color**. Return true if the graph can be colored with at most **m** colors, otherwise return false.

Note: The graph is indexed with 0-based indexing.

Examples:

Input: V = 4, edges[][] = [[0, 1], [1, 3], [2, 3], [3, 0], [0, 2]], m = 3
Output: true
Explanation: It is possible to color the given graph using 3 colors, for example, one of the possible ways vertices can be colored as follows:

```
class Solution:
    def graphColoring(self, V, edges, m):

        graph = {i: [] for i in range(V)}
        for u, v in edges:
            graph[u].append(v)
            graph[v].append(u)

        colors = [0] * V


        def isSafe(node, c):
            for neighbor in graph[node]:
                if colors[neighbor] == c:
                    return False
            return True


        def solve(node):
            if node == V:
                return True

            for c in range(1, m + 1):
                if isSafe(node, c):
                    colors[node] = c
                    if solve(node + 1):
                        return True
                    colors[node] = 0

            return False
```



Problem Solved Successfully

Test Cases Passed
1114 / 1114

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 16

Time Taken
0.05

Solve Next

Rat in a Maze · Black and White · Walls Coloring

```python
class Solution:
    def graphColoring(self, V, edges, m):

        graph = {i: [] for i in range(V)}
        for u, v in edges:
            graph[u].append(v)
            graph[v].append(u)

        colors = [0] * V


        def isSafe(node, c):
            for neighbor in graph[node]:
                if colors[neighbor] == c:
                    return False
            return True


        def solve(node):
            if node == V:
                return True

            for c in range(1, m + 1):
                if isSafe(node, c):
                    colors[node] = c
                    if solve(node + 1):
                        return True
                    colors[node] = 0

            return False
```