

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
```

The screenshot shows a LeetCode problem page for "M-Coloring Problem". The problem details are as follows:

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 $\text{edges}[[\cdot]]$, 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: $V = 4$, $\text{edges}[[\cdot]] = [[0, 1], [1, 3], [2, 3], [3, 0], [0, 2]]$, $m = 3$
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:

A diagram shows a graph with 4 vertices labeled 0, 1, 2, 3. Vertex 0 is at the top, vertex 1 is bottom-left, vertex 2 is bottom-right, and vertex 3 is top-right. Edges connect (0,1), (1,3), (2,3), (3,0), and (0,2).

The code editor contains the following Python3 solution:

```
1 class Solution:
2     def graphColoring(self, V, edges, m):
3         graph = {i: [] for i in range(V)}
4         for u, v in edges:
5             graph[u].append(v)
6             graph[v].append(u)
7
8         colors = [0] * V
9
10
11     def isSafe(node, c):
12         for neighbor in graph[node]:
13             if colors[neighbor] == c:
14                 return False
15         return True
16
17
18     def solve(node):
19         if node == V:
20             return True
21
22         for c in range(1, m + 1):
23             if isSafe(node, c):
24                 colors[node] = c
25                 if solve(node + 1):
26                     return True
27                 colors[node] = 0
28
29
30     return False
```

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

geeksforgeeks.org/problems/m-coloring-problem-1587115620/1

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 1114 / 1114

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

Points Scored 4 / 4 Time Taken 0.05

Your Total Score: 16 ↑

Solve Next

Rat in a Maze Black and White Walls Coloring

Python3 Start Timer

```
1 - class Solution:
2 -     def graphColoring(self, V, edges, m):
3 -
4 -         graph = {i: [] for i in range(V)}
5 -         for u, v in edges:
6 -             graph[u].append(v)
7 -             graph[v].append(u)
8 -
9 -         colors = [0] * V
10 -
11 -         def isSafe(node, c):
12 -             for neighbor in graph[node]:
13 -                 if colors[neighbor] == c:
14 -                     return False
15 -
16 -             return True
17 -
18 -         def solve(node):
19 -             if node == V:
20 -                 return True
21 -
22 -             for c in range(1, m + 1):
23 -                 if isSafe(node, c):
24 -                     colors[node] = c
25 -                     if solve(node + 1):
26 -                         return True
27 -                     colors[node] = 0
28 -
29 -             return False
30 -
```

Custom Input Compile & Run Submit