

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 displays a coding platform interface for the 'M-Coloring Problem'. The problem description states: '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:'. A diagram shows a graph with 4 vertices (0, 1, 2, 3) and 5 edges. Vertex 0 is colored red, 1 is blue, 2 is green, and 3 is red. The solution code is written in Python, defining a `graphColoring` method that initializes the graph, sets up a `colors` array, and uses a recursive `solve` function to check if the graph can be colored with  $m$  colors. The code includes a `graphColoring` method that initializes the graph, sets up a `colors` array, and uses a recursive `solve` function to check if the graph can be colored with  $m$  colors. The code includes a `graphColoring` method that initializes the graph, sets up a `colors` array, and uses a recursive `solve` function to check if the graph can be colored with  $m$  colors.

```
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
12     def isSafe(node, c):
13         for neighbor in graph[node]:
14             if colors[neighbor] == c:
15                 return False
16         return True
17
18
19     def solve(node):
20         if node == V:
21             return True
22
23         for c in range(1, m + 1):
24             if isSafe(node, c):
25                 colors[node] = c
26                 if solve(node + 1):
27                     return True
28                 colors[node] = 0
29
30         return False
```

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

Search...

CoursesTutorialsPracticeJobs

⌕🔔🔔M

ProblemEditorialSubmissionsComments

Output Window

Compilation ResultsCustom InputY.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 MazeBlack and WhiteWalls Coloring

Python3Start 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
12     def isSafe(node, c):
13         for neighbor in graph[node]:
14             if colors[neighbor] == c:
15                 return False
16         return True
17
18
19     def solve(node):
20         if node == V:
21             return True
22
23         for c in range(1, m + 1):
24             if isSafe(node, c):
25                 colors[node] = c
26                 if solve(node + 1):
27                     return True
28                 colors[node] = 0
29
30         return False
```

Custom InputCompile & RunSubmit