**M-Coloring Problem**

**Medium**Accuracy: 47.46% Submissions: 21378 Points: 4

Given an undirected graph and an integer **M**. The task is to determine if the graph can be coloured with at most M colours such that no two adjacent vertices of the graph are coloured with the same colour. Here colouring of a graph means the assignment of colours to all vertices. Print 1 if it is possible to colour vertices and 0 otherwise.

Chart

Description automatically generated

**Example 1:**

**Input:**

N = 4

M = 3

E = 5

Edges[] = {(0,1),(1,2),(2,3),(3,0),(0,2)}

**Output:** 1

**Explanation:** It is possible to colour the

given graph using 3 colours.

**Example 2:**

**Input:**

N = 3

M = 2

E = 3

Edges[] = {(0,1),(1,2),(0,2)}

**Output:** 0

**Your Task:**

Your task is to complete the function **graphColoring()** which takes the 2d-array graph[], the number of colours and the number of nodes as inputs and returns **true** if answer exists otherwise **false**. 1 is printed if the returned value is **true,**0 otherwise. The printing is done by the driver's code.  
**Note**: In Example there are Edges not the graph.Graph will be like, if there is an edge between vertex X and vertex Y graph[] will contain 1 at graph[X-1][Y-1], else 0. In 2d-array graph[ ], nodes are 0-based indexed, i.e. from 0 to N-1.Function will be contain 2-D graph not the edges.  
  
**Expected Time Complexity:** O(MN).  
**Expected Auxiliary Space:** O(N).

**Constraints:**  
1 ≤ N ≤ 20  
1 ≤ E ≤ (N\*(N-1))/2  
1 ≤ M ≤ N

bool check(int v, vector<vector<int>> &graph,  int c, vector<int> &color) {

    for (auto x : graph[v]) {

        if (color[x]!=-1 and color[x]==c) return false;

    }

    return true;

}

bool Mcoloring(int v, vector<vector<int>> &graph, vector<int> &color, int V, int m) {

    if (v==V) return true;

    for (int i=0; i<m; i++) {

        if (check(v, graph, i, color)) {

            color[v]=i;

            bool next=Mcoloring(v+1, graph, color, V, m);

            if (next) return true;

            color[v]=-1;

        }

    }

    return false;

}

bool graphColoring(bool adj[101][101], int m, int V){

    // your code here

    vector<vector<int>> graph(V);

    for (int i=0; i<V; i++) {

        for (int j=0; j<V; j++) {

            if (adj[i][j]==1) graph[i].push\_back(j);

        }

    }

    vector<int> color(V, -1);

    color[0]=0;

    return Mcoloring(0, graph, color, V, m);

}