



100%

Normal text

Arial

11

B

I

U

A

P

GD

D

E

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

ONnb1wdx2Ma&index=55

Day10: (Recursion and Backtracking)

1. Print all Permutations of a string/array

https://www.youtube.com/watch?v=f2lc2Rsc9pU&list=PLgUwDviBIf0p4ozDR_kJJkO
Nb1wdx2Ma&index=52

2. N queens Problem

https://www.youtube.com/watch?v=i05Ju7AftcM&list=PLgUwDviBIf0p4ozDR_kJJkON
nb1wdx2Ma&index=57

3. Sudoku Solver

https://www.youtube.com/watch?v=FWAlf_EVUKE&list=PLgUwDviBIf0p4ozDR_kJJk
ONnb1wdx2Ma&index=58

4. M coloring Problem

5. Rat in a Maze

6. Word Break (print all ways)



0:05 / 24:36



TUF

M-Coloring Problem

Medium Accuracy: 33.66% Submissions: 6921 Points: 4

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

Example 1:

Input:

N = 4

M = 3

E = 5

Edges[] = {(1,2),(2,3),(3,4),(4,1),(1,3)}

Output: 1

Explanation: It is possible to colour the given graph using 3 colours.

L16. M-Coloring Problem | Backtracking

M-Coloring Problem



Medium Accuracy: 33.66% Submissions: 6921 Points: 4

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

Example 1:

Input:

N = 4

M = 3

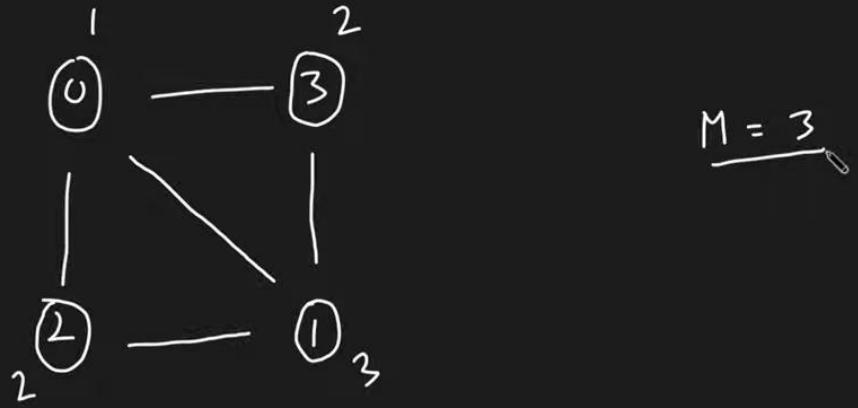
E = 5

Edges[] = {(1,2),(2,3),(3,4),(4,1),(1,3)}

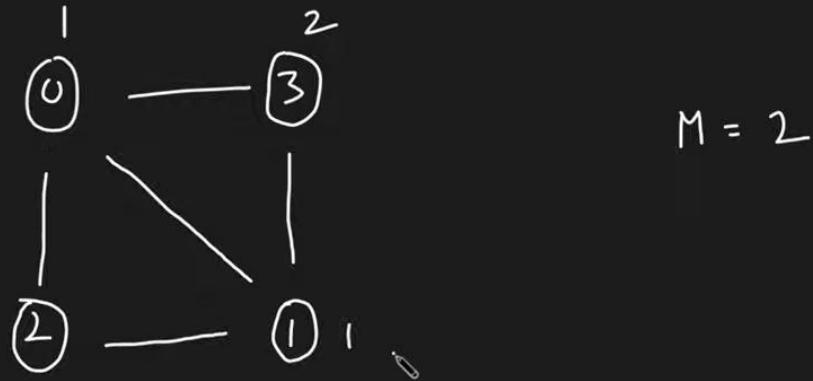
Output:

Explanation: It is possible to colour the graph using 3 colours.

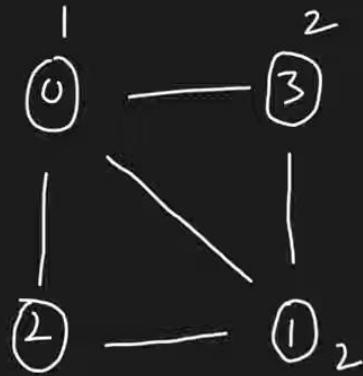
TAKEU FORWARD



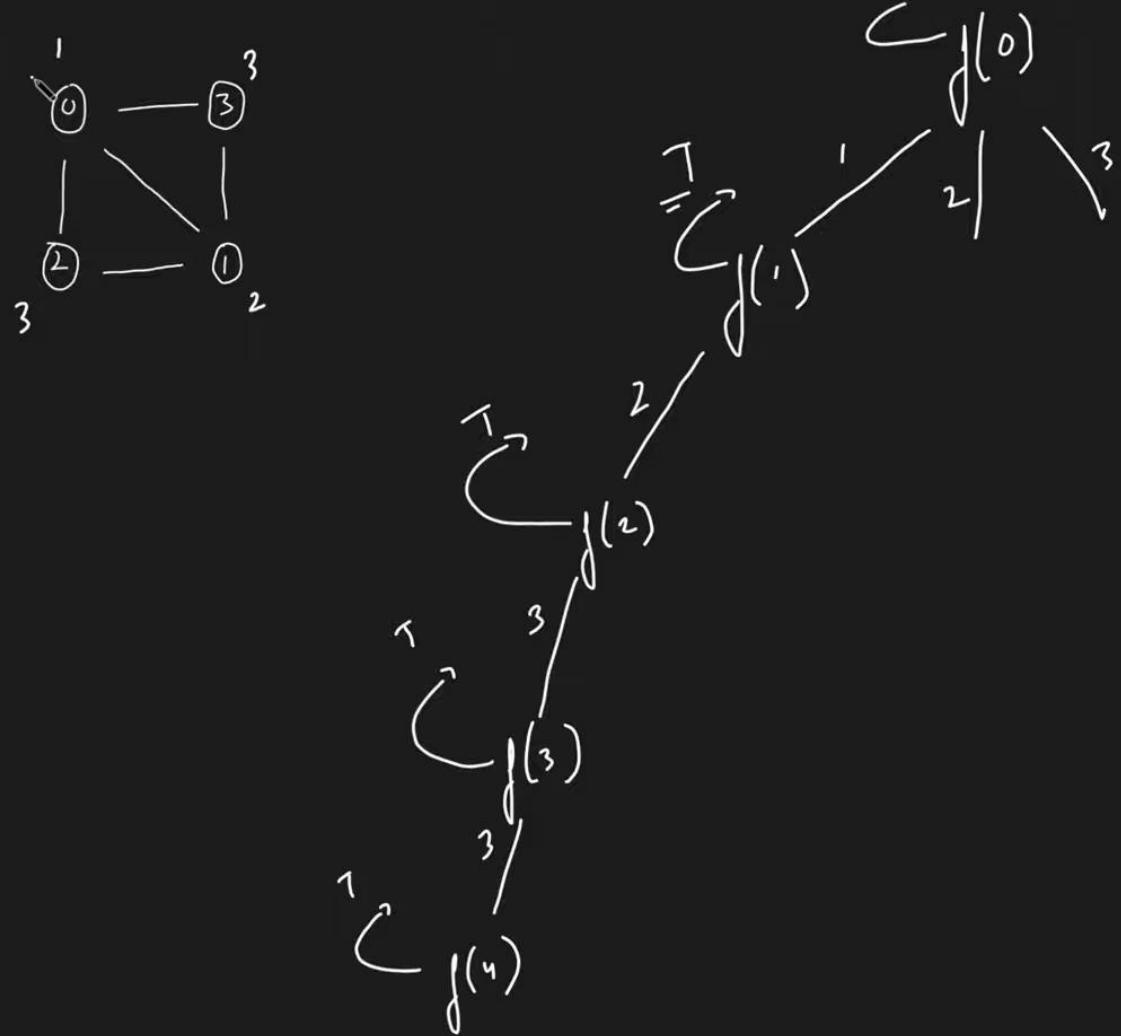
$$\underline{M = 3}$$



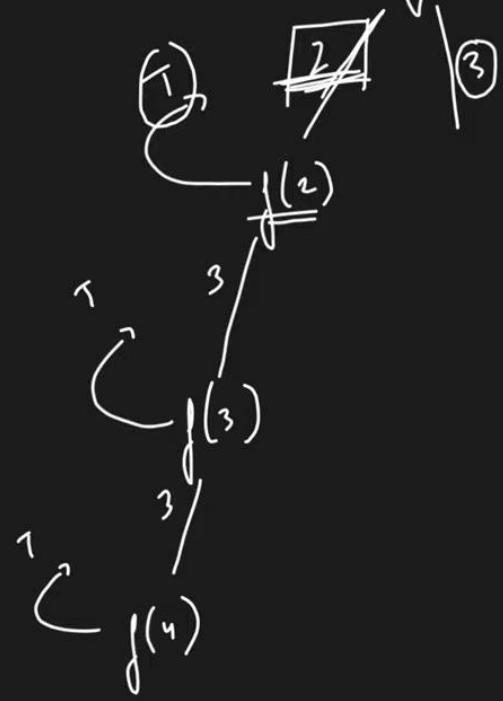
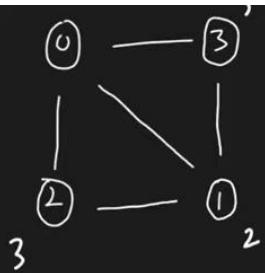
$$M = 2$$



M ≈ 2



$$\begin{array}{c} M=3 \\ \hline N=4 \end{array}$$



$f(\text{node})$

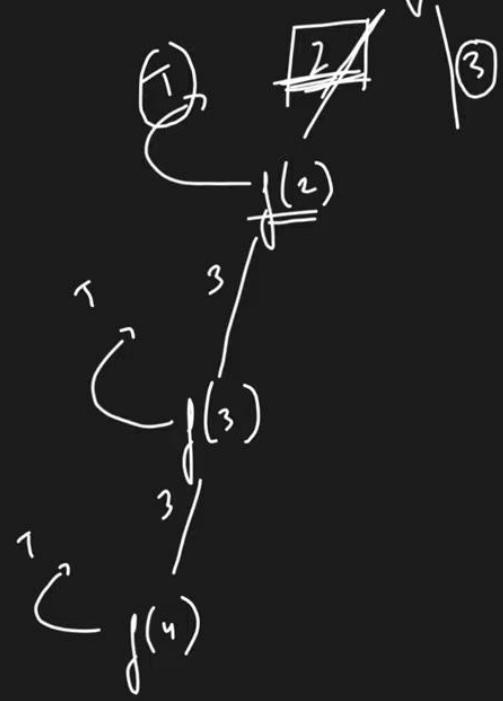
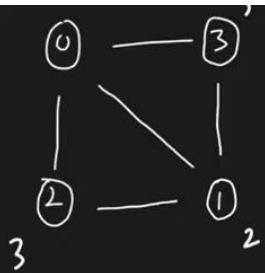
i (node == N) return (;

$$\{ \text{color}_j \text{ node} \} = \text{col}_j$$

if ($f(\text{node} + 1) = -1$)
 return T;

3. $\text{color}[\underline{\text{node}}] = 0;$

7



$f(\underline{\text{node}})$

$y(\text{node} == N)$ return T;

$$\int_{\text{ca}} \left(\underline{C_0} \underline{\underline{L}} = \underline{\underline{I}} \rightarrow \underline{\underline{M}} \right)$$

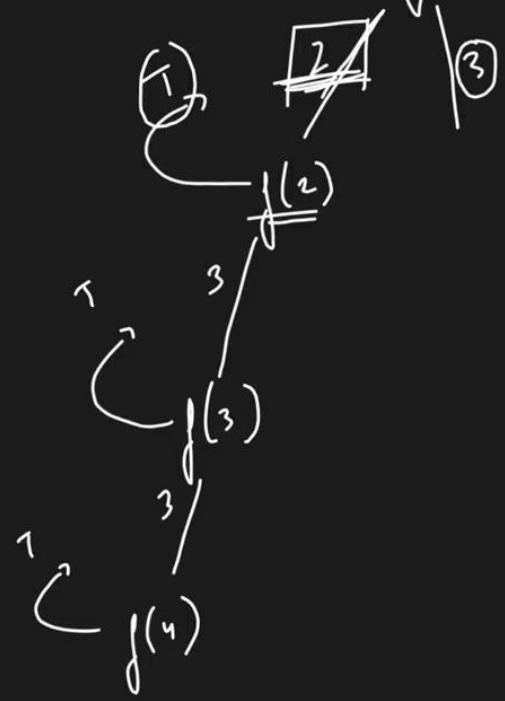
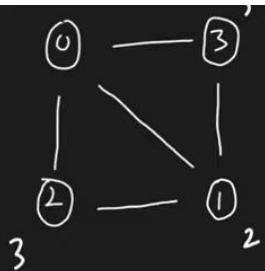
$\stackrel{=}{\sim}$ (possible $\rightarrow \checkmark$)

$$\text{color}[\eta_{\text{node}}] = \text{col}_j$$

if $\left(\frac{\text{node} + 1}{\text{node}} = -1 \right)$

3. $\text{column}[\underline{\text{node}}] = 0;$

6



$f(\underline{\text{node}})$

$y(\text{node} == N)$ return T ;

$$\int_{\text{ca}} \left(\underline{L_0} = \underline{I} \rightarrow \underline{m} \right)$$

$\stackrel{=}{\sim}$ (possible $\rightarrow \checkmark$)

$$\text{color}[\eta_{\text{node}}] = \text{col}_j$$

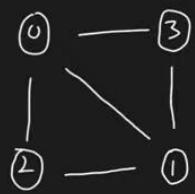
$$ij \left(j(\text{node}+1) = -7 \right)$$

$$\text{color}[\text{node}] = 0;$$

{ rub } F ;

M-3

N=7
=



$$\begin{array}{l} M=3 \\ N=4 \\ \hline \end{array}$$

$f(\underline{\text{node}})$

{ if ($\text{node} == N$) return T;

fun($\underline{\text{col}} = (\underline{l} \rightarrow \underline{m})$)

{ if (possible $\rightarrow \checkmark$)

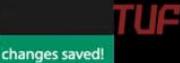
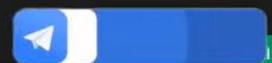
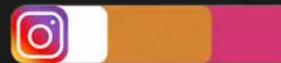
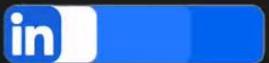
{ color[$\underline{\text{node}}$] = $\underline{\text{col}}$;

if ($f(\underline{\text{node}+1}) == \checkmark$)

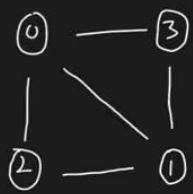
$\underline{\text{node}}$ T;

, color[$\underline{\text{node}}$] = 0 ;

} F ;



1 changes saved!



$$\boxed{M=3^2}$$

$$\boxed{N=9}$$

$f(\underline{\text{node}})$

{ if ($\text{node} == N$) return T;

fun ($\underline{\text{col}} = (\underline{l} \rightarrow \underline{m})$)

{ if ($\text{possible} \rightarrow \checkmark$)

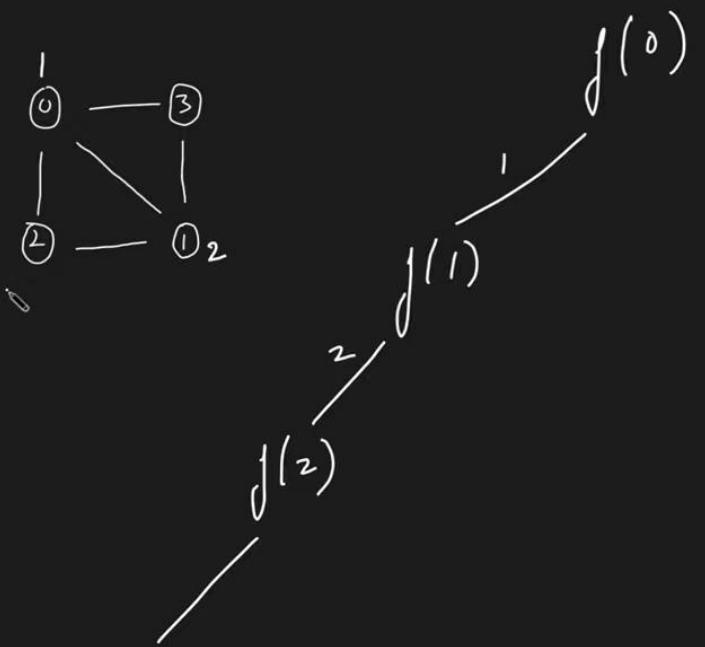
{ $\text{color}[\text{node}] = \text{col}$;

if ($f(\text{node} + 1) == \checkmark$)

$\boxed{\text{return } T}$;

, $\text{color}[\underline{\text{node}}] = 0$;

, return F ;



$\boxed{M=3 \text{ } 2}$
 $N=4$
 $=$
 $f(\underline{\text{node}})$
 $\left\{ \begin{array}{l} \text{if } (\text{node} == N) \text{ return } T; \\ \dots \end{array} \right.$
 $f(\underline{\text{col}}) = (\underline{l} \rightarrow \underline{m})$
 $\left\{ \begin{array}{l} \text{if } (\text{possible} \rightarrow \checkmark) \\ \quad \text{color}[\text{node}] = \text{col}; \\ \quad \text{if } (f(\text{node}+1) == \checkmark) \\ \quad \quad \boxed{\text{return } T}; \\ \quad \quad \text{color}[\underline{\text{node}}] = 0; \\ \quad \quad \text{return } F; \end{array} \right.$

i

Let's give back to the community :)

$$\boxed{M=3^2}$$

$$N=9 \\ =$$

$f(\underline{\text{node}})$

if ($\text{node} == N$) return T;

fun($\underline{\text{col}} = (\underline{l} \rightarrow \underline{m})$)

if (possible $\rightarrow \checkmark$)

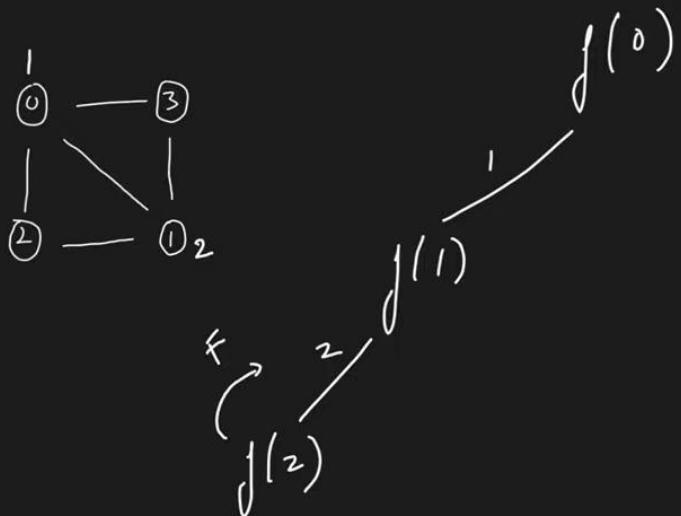
{
 $\text{color}[\text{node}] = \text{col}$;
 if ($f(\text{node}+1) == \checkmark$)

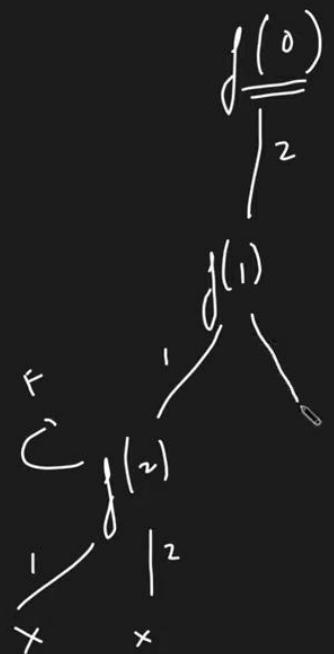
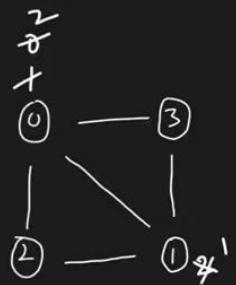
$\boxed{\text{return } T}$;

, $\text{color}[\underline{\text{node}}] = 0$;

, $\boxed{\text{return } F}$;

TUF





$f(\underline{\text{node}})$

if ($\text{node} == N$) return T;

fun ($\underline{\text{color}} = (\underline{\text{L}} \rightarrow \underline{\text{m}})$)
if ($\text{possible} \rightarrow \checkmark$)

 if ($\text{color}[\text{node}] == \underline{\text{color}}$)
 if ($f(\text{node} + 1) == \underline{\text{T}}$)

 color[node] = 0;

f(2) is possible ?

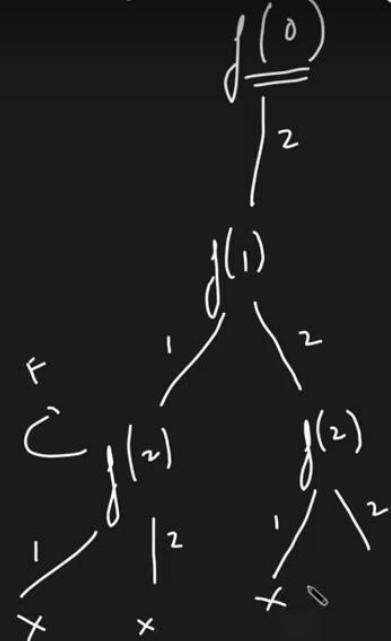
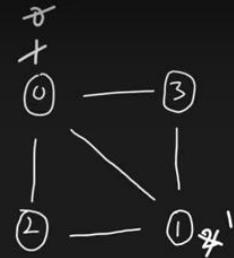
At f(1), we cannot use color 2, since adjacent nodes cannot have same color, I in a hurry just drew the tree of f(2), ignore that

, return F,

TUF

All changes saved!

L16, M-Coloring Problem | Backtracking



$$\boxed{M=3 \quad 2}$$

$$N=4$$

$f(\underline{\text{node}})$

if ($\text{node} == N$) return T;

fun ($\underline{\text{color}} = (\underline{\text{color}} \rightarrow \underline{\text{color}})$)

if (possible $\rightarrow \checkmark$)

 color[node] = color;

 if ($f(\text{node} + 1) == \checkmark$)

f(2) is possible ?

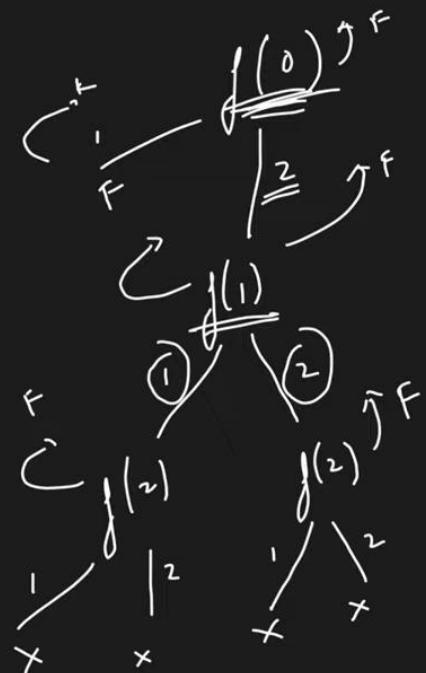
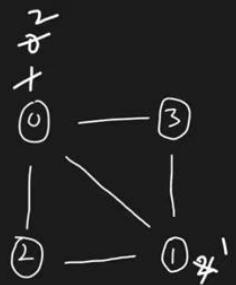
At f(1), we cannot use color 2, since adjacent nodes cannot have same color, I in a hurry just drew the tree of f(2), ignore that

TUF



16:08 / 24:36





$$\begin{aligned} TC &\rightarrow \binom{N^m}{m} \\ SC &\rightarrow O(N) + O(N) \end{aligned}$$

M=3 2

$$N=4$$

f(node)

iif (node == N) return T;

0
1
1
2
3

fun (col = (1 → m))

iif (possible → ✓)

 ? column[node] = col; ✓

 iif (f(node+1) == T)

 ? column[T];

 , column[node] = 0;

 → return F;

```

37
38
39 class solve
40 {
41     private static boolean isSafe(int node, List<Integer>[] G, int[] color, int n, int col) {
42         for(int it: G[node]) {
43             if(color[it] == col) return false;
44         }
45         return true;
46     }
47     private static boolean solve(int node, List<Integer>[] G, int[] color, int n, int m) {
48         if(node == n) return true;
49
50         for(int i = 1;i<=m;i++) {
51             if(isSafe(node, G, color, n, i)) {
52                 color[node] = i;
53                 if(solve(node+1, G, color, n, m) == true) return true;
54                 color[node] = 0;
55             }
56         }
57         return false;
58     }
59     public static boolean graphColoring(List<Integer>[] G, int[] color, int i, int m)
60     {
61         int n = G.length;
62         if(solve(i, G, color, n, m) == true) return true;
63         return false;
64         // Your code here
65     }
66 }
67
68

```

```

37
38
39 class solve
40 {
41     private static boolean isSafe(int node, List<Integer>[] G, int[] color, int n, int col) {
42         for(int it: G[node]) {
43             if(color[it] == col) return false;
44         }
45         return true;
46     }
47     private static boolean solve(int node, List<Integer>[] G, int[] color, int n, int m) {
48         if(node == n) return true;
49
50         for(int i = 1;i<=m;i++) {
51             if(isSafe(node, G, color, n, i)) {
52                 color[node] = i;
53                 if(solve(node+1, G, color, n, m) == true) return true;
54                 color[node] = 0;
55             }
56         }
57         return false;
58     }
59     public static boolean graphColoring(List<Integer>[] G, int[] color, int i, int m)
60     {
61         int n = G.length;
62         if(solve(i, G, color, n, m) == true) return true;
63         return false;
64         // Your code here
65     }
66 }
67
68

```

```

10
11- bool isSafe(int node, int color[], bool graph[101][101], int n, int col) {
12-     for(int k = 0;k<n;k++) {
13-         if(k != node && graph[k][node] == 1 && color[k] == col) {
14-             return false;
15-         }
16-     }
17-     return true;
18 }
19- bool solve(int node, int color[], int m, int N, bool graph[101][101]) {
20-     if(node == N) {
21-         return true;
22-     }
23
24-     for(int i = 1;i<=m;i++) {
25-         if(isSafe(node, color, graph, N, i)) {
26-             color[node] = i;
27-             if(solve(node+1, color, m, N, graph)) return true;
28-             color[node] = 0;
29-         }
30
31     }
32     return false;
33 }
34
35 //Function to determine if graph can be coloured with at most M colours such
36 //that no two adjacent vertices of graph are coloured with same colour.
37 bool graphColoring(bool graph[101][101], int m, int N)
38 {
39     int color[N] = {0};
40     if(solve(0,color,m,N,graph)) return true;
41     return false;
42 }
43 // } Driver Code Ends

```