

**Name: Nachiket Jagdale**

**Class: A3-B1**

**Roll no.-02**

## **PRACTICAL NO. 8**

**Aim:** Implement Graph Colouring algorithm use Graph colouring concept.

**Problem Statement:**

A GSM is a cellular network with its entire geographical range divided into

hexadecimal cells. Each cell has a communication tower which connects with mobile

phones within cell. Assume this GSM network operates in different frequency

ranges. Allot frequencies to each cell such that no adjacent cells have same

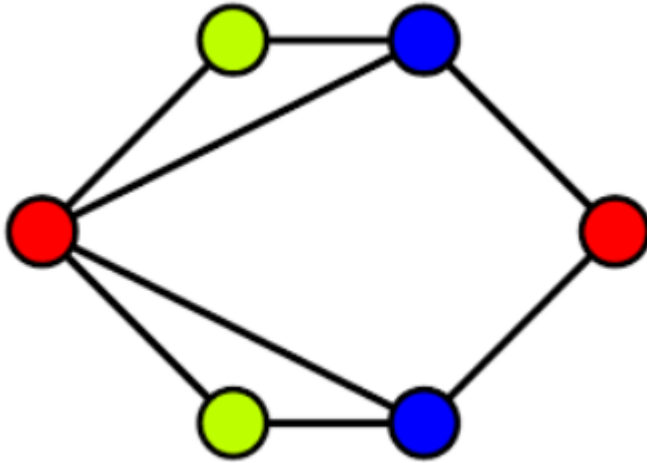
frequency range.

Consider an undirected graph  $G = (V, E)$  shown in fig. Find the colour assigned to each node

using Backtracking method. Input is the adjacency matrix of a graph  $G(V, E)$ , where  $V$  is the

number of Vertices and  $E$  is the number of edges.

**Graph 1:**



### CODE:

```
#include <stdio.h>
```

```
#include <stdbool.h>
```

```
#define V 10
```

```
bool isSafe(int v, bool graph[V][V], int color[], int c, int n) {  
    for (int i = 0; i < n; i++)  
        if (graph[v][i] && color[i] == c)  
            return false;  
    return true;  
}
```

```
bool solveGraph(bool graph[V][V], int m, int color[], int v, int n) {  
    if (v == n)  
        return true;  
    for (int c = 1; c <= m; c++) {
```

```
if (isSafe(v, graph, color, c, n)) {  
    color[v] = c;  
    if (solveGraph(graph, m, color, v + 1, n))  
        return true;  
    color[v] = 0;  
}  
}  
return false;  
}
```

```
int main() {  
    bool graph[V][V];  
    int n, m;  
    int color[V] = {0};  
  
    printf("Enter number of vertices: ");  
    scanf("%d", &n);  
  
    printf("Enter adjacency matrix (%d x %d):\n", n, n);  
    for (int i = 0; i < n; i++)  
        for (int j = 0; j < n; j++)  
            scanf("%d", &graph[i][j]);  
  
    printf("Enter number of colors: ");  
    scanf("%d", &m);  
}
```

```

if (solveGraph(graph, m, color, 0, n)) {
printf("Colors assigned:\n");
for (int i = 0; i < n; i++)
printf("Node %d -> Color %d\n", i + 1, color[i]);
} else {
printf("No solution exists\n");
}

return 0;
}

```

## OUTPUT:

The screenshot shows a C++ program being executed in an online compiler. The code defines a graph with 5 vertices and 5 colors, and prints the assigned colors for each node.

```

1 #include <stdio.h>
2 #include <stdbool.h>
3
4 #define V 10
5
6 bool isSafe(int v, bool graph[V][V], int color[], int c, int n) {
7     for (int i = 0; i < n; i++)
8         if (graph[v][i] && color[i] == c)
9             return false;
10    return true;
11 }
12
13 bool solveGraph(bool graph[V][V], int m, int color[], int v, int n) {
14     if (v == n)
15         return true;
16     for (int c = 1; c <= m; c++) {
17         if (isSafe(v, graph, color, c, n)) {
18             color[v] = c;
19             if (solveGraph(graph, m, color, v + 1, n))
20                 return true;
21             color[v] = 0;
22         }
23     }
24     return false;
25 }
26
27 int main() {
28     bool graph[V][V];
29     int n, m;
30     int color[V] = {0};
31
32     printf("Enter number of vertices: ");
33     scanf("%d", &n);

```

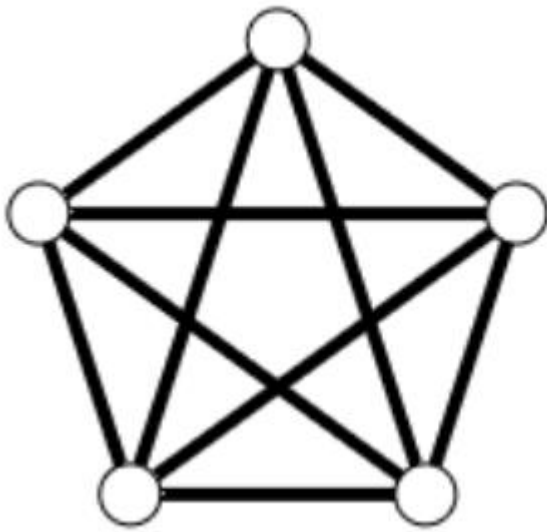
Output:

```

Enter number of vertices: 5
Enter adjacency matrix (5 x 5):
0 1 1 0
1 0 1 0 0
1 1 0 0 1
1 0 0 0 1
0 0 1 1 0
Enter number of colors: 5
Colors assigned:
Node 1 -> Color 1
Node 2 -> Color 2
Node 3 -> Color 3
Node 4 -> Color 2
Node 5 -> Color 1
--- Code Execution Successful ---

```

## GRAPH 2:



**CODE:**

```
#include <stdio.h>
```

```
#include <stdbool.h>
```

```
#define V 10
```

```
bool isSafe(int v, bool graph[V][V], int color[], int c, int n) {  
    for (int i = 0; i < n; i++)  
        if (graph[v][i] && color[i] == c)  
            return false;  
    return true;  
}
```

```
bool solveGraph(bool graph[V][V], int m, int color[], int v, int n) {  
    if (v == n)
```

```

return true;
for (int c = 1; c <= m; c++) {
    if (isSafe(v, graph, color, c, n)) {
        color[v] = c;
        if (solveGraph(graph, m, color, v + 1, n))
            return true;
        color[v] = 0;
    }
}
return false;
}

```

```

int main() {
    bool graph[V][V];
    int n, m;
    int color[V] = {0};

    printf("Enter number of vertices: ");
    scanf("%d", &n);

    printf("Enter adjacency matrix (%d x %d):\n", n, n);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);
}

```

```

printf("Enter number of colors: ");
scanf("%d", &m);

if (solveGraph(graph, m, color, 0, n)) {
printf("Colors assigned:\n");
for (int i = 0; i < n; i++)
printf("Node %d -> Color %d\n", i + 1, color[i]);
} else {
printf("No solution exists\n");
}

return 0;
}

```

## OUTPUT:

The screenshot shows a C program being executed in an online compiler. The code on the left defines a graph with 5 vertices and 5 colors, and the output on the right shows the assigned colors for each node.

```

main.c
22 }
23 }
24 return false;
25 }
26
27 int main() {
28     bool graph[V][V];
29     int n, m;
30     int color[V] = {0};
31
32     printf("Enter number of vertices: ");
33     scanf("%d", &n);
34
35     printf("Enter adjacency matrix (%d x %d):\n", n, n);
36     for (int i = 0; i < n; i++)
37         for (int j = 0; j < n; j++)
38             scanf("%d", &graph[i][j]);
39
40     printf("Enter number of colors: ");
41     scanf("%d", &m);
42
43     if (solveGraph(graph, m, color, 0, n)) {
44         printf("Colors assigned:\n");
45         for (int i = 0; i < n; i++)
46             printf("Node %d -> Color %d\n", i + 1, color[i]);
47     } else {
48         printf("No solution exists\n");
49     }
50
51     return 0;
52 }
53
54
55

```

Enter number of vertices: 5  
Enter adjacency matrix (5 x 5):  
0 1 1 1 1  
1 0 1 1 1  
1 1 0 1 1  
1 1 1 0 1  
1 1 1 1 0  
Enter number of colors: 5  
Colors assigned:  
Node 1 -> Color 1  
Node 2 -> Color 2  
Node 3 -> Color 3  
Node 4 -> Color 4  
Node 5 -> Color 5  
== Code Execution Successful ==

Competitive Coding Link:

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

### CODE:

```
import java.util.*;

class Solution {
    boolean graphColoring(int v, int[][] edges, int m) {
        boolean[][] graph = new boolean[v][v];
        for (int[] e : edges) {
            graph[e[0]][e[1]] = true;
            graph[e[1]][e[0]] = true;
        }
        int[] color = new int[v];
        return solve(0, graph, color, m, v);
    }

    boolean solve(int node, boolean[][] graph, int[] color, int m, int v) {
        if (node == v) return true;
        for (int c = 1; c <= m; c++) {
            if (isSafe(node, graph, color, v, c)) {
                color[node] = c;
                if (solve(node + 1, graph, color, m, v)) return true;
                color[node] = 0;
            }
        }
    }
}
```



```

    }
    return false;
}

boolean isSafe(int node, boolean[][] graph, int[] color, int v,
int c) {
    for (int i = 0; i < v; i++)
        if (graph[node][i] && color[i] == c)
            return false;
    return true;
}
}

```

## OUTPUT:

**Problem Solved Successfully**

[Suggest Feedback](#)

Test Cases Passed <b>1114 / 1114</b>	Attempts : Correct / Total <b>1 / 1</b> Accuracy : 100%
Points Scored <b>4 / 4</b> Your Total Score: 16	Time Taken <b>0.33</b>

**Solve Next**

Rat in a Maze

Black and White

Walls Coloring