

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

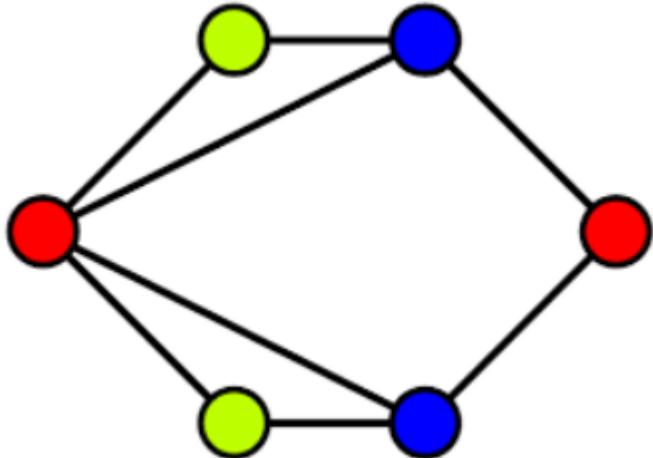
frequency range.

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 the Programiz C Online Compiler interface. The code in the editor is:

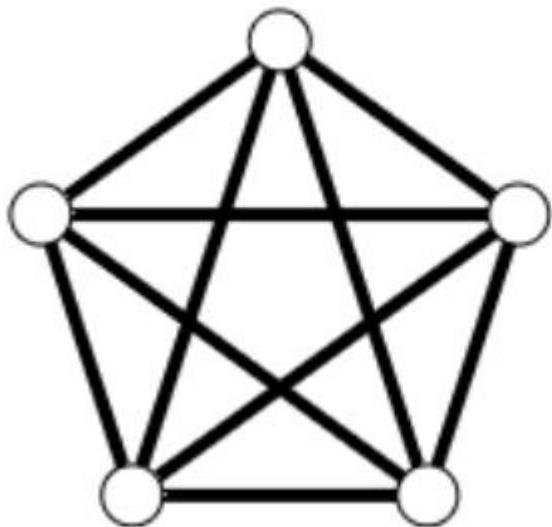
```

main.c
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);
34
35     Enter number of vertices: 5
36     Enter adjacency matrix (5 x 5):
37     0 1 1 1 0
38     1 0 1 0 0
39     1 1 0 0 1
40     1 0 0 0 1
41     0 0 1 1 0
42
43     Enter number of colors: 5
44     Colors assigned:
45     Node 1 -> Color 1
46     Node 2 -> Color 2
47     Node 3 -> Color 3
48     Node 4 -> Color 2
49     Node 5 -> Color 1

```

The output window shows the user input for vertices (5), the adjacency matrix, and the number of colors (5). It then displays the "Colors assigned" for each node, which is a valid solution for the 5-vertex graph.

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 the Programiz C Online Compiler interface. On the left, the code for `main.c` is displayed. On the right, the terminal window shows the execution process. The user enters the number of vertices (5) and the adjacency matrix (5x5). The program then asks for the number of colors (5) and prints the assigned colors for each node.

```

Programiz C Online Compiler
main.c
1 Enter number of vertices: 5
2 Enter adjacency matrix (5 x 5):
3 0 1 1 1 1
4 1 0 1 1 1
5 1 1 0 1 1
6 1 1 1 0 1
7 1 1 1 1 0
8 Enter number of colors: 5
9 Colors assigned:
10 Node 1 -> Color 1
11 Node 2 -> Color 2
12 Node 3 -> Color 3
13 Node 4 -> Color 4
14 Node 5 -> Color 5
15 === Code Execution Successful ===

```

Competitive Coding Link:

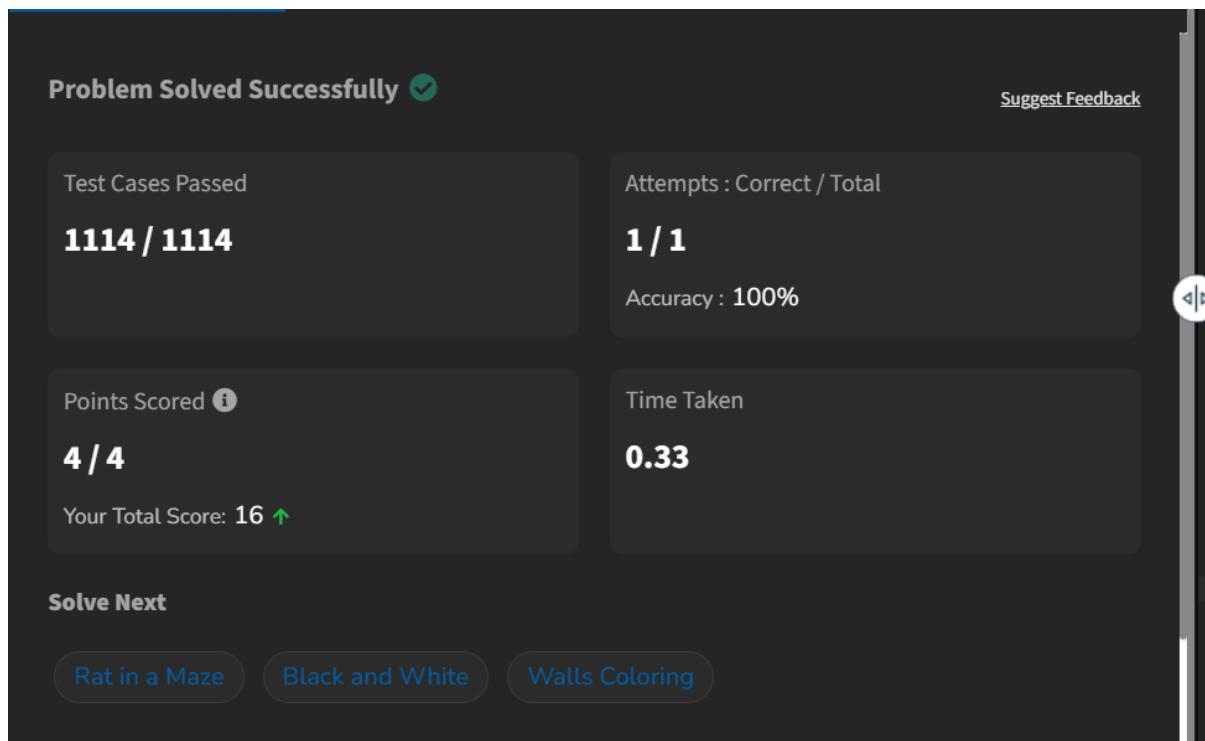
<https://www.geeksforgeeks.org/problems/m-coloring-problem-1587115620/1>

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;  
            }  
        }  
    }  
    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;  
    }  
}
```

```
        }  
        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:



The screenshot shows a summary of a solved problem on a platform. The top bar indicates "Problem Solved Successfully" with a green checkmark and a "Suggest Feedback" button. Below this, there are four main performance metrics displayed in a grid:

Test Cases Passed	Attempts : Correct / Total
1114 / 1114	1 / 1
Points Scored ⓘ	Accuracy : 100%
4 / 4	0.33

Additional information below the grid includes "Your Total Score: 16 ↑" and a "Solve Next" button. At the bottom, there are three navigation links: "Rat in a Maze", "Black and White", and "Walls Coloring".