DESIGN AND ANALYSIS OF ALGORITHMS LAB III SEMESTER

Practical No. : 08

Name : Vardhan Ingole

Section : A4_B3

Roll No : 43

Aim: Implement Graph Colouring algorithm using Graph colouring concept.

Problem Statement:

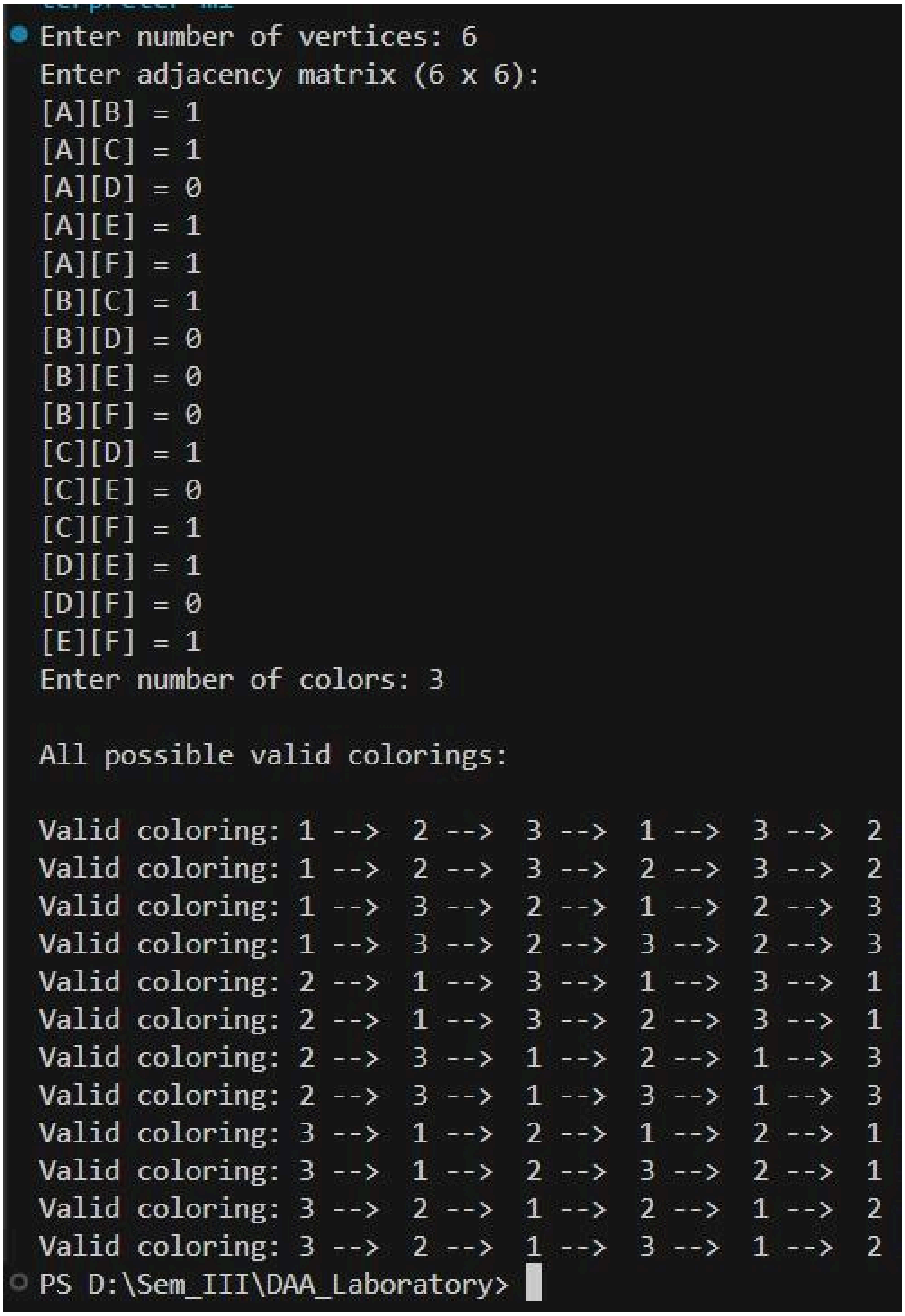
A GSM is a cellular network with its entire geographical range divided into hexadecimal cells. Each cell has a communication tower which connects with mobile phones within the cell. Assume this GSM network operates in different frequency ranges. Allot frequencies to each cell such that no adjacent cells have the same frequency range.

Code :

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

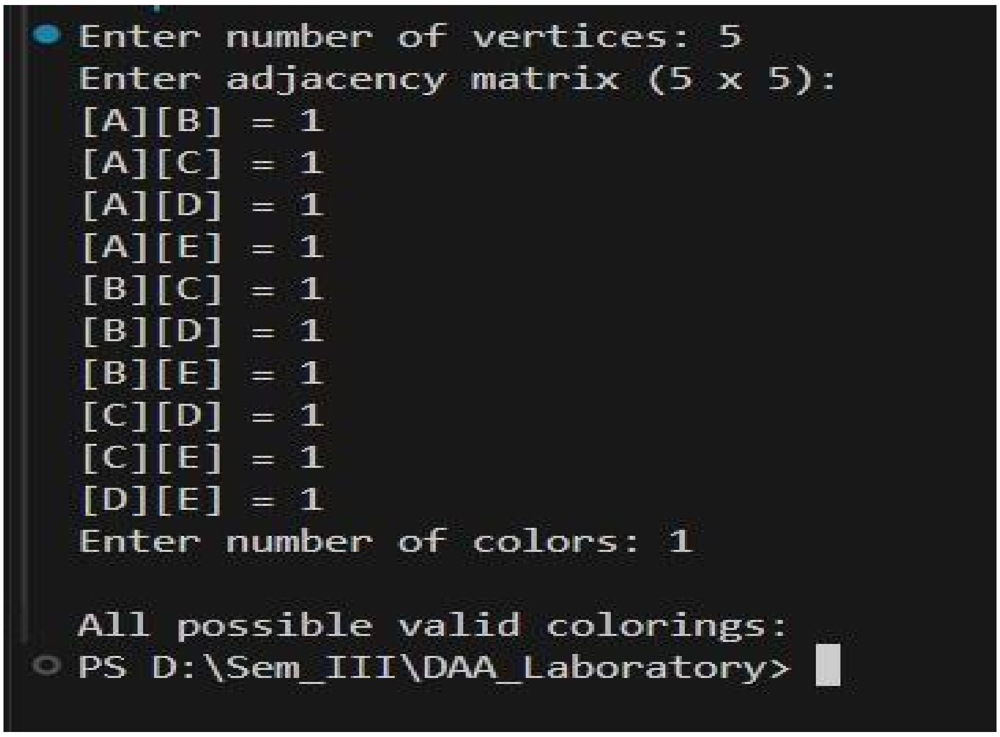
```c
#define MAX 10

int n;
int m;
int G[MAX][MAX];
int x[MAX];


void NextValue(int k)
{
    int j;

    while (1)
    {
        x[k] = (x[k] + 1) % (m + 1);

        if (x[k] == 0)
            return;


        for (j = 1; j <= n; j++)
        {
            if (G[k][j] != 0 && x[k] == x[j])
                break;
        }

        if (j == n + 1)
            return;
    }
}


void mColoring(int k)
{
    while (1)
    {
        NextValue(k);

        if (x[k] == 0)
            return;

        if (k == n)
        {

            printf("\nValid coloring: ");
            for (int i = 1; i <= n; i++)
                printf("%d ", x[i]);
        }
        else
        {

            mColoring(k + 1);
        }
    }
}

int main()
{
    int i, j;
    char ch = 65;

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

    printf("Enter adjacency matrix (%d x %d):\n", n, n);
    for (i = 1; i <= n; i++)
    {
        for (j = i; j <= n; j++)
        {
            if (i != j)
            {
                printf("[%c][%c] = ", ch+ i - 1, ch + j - 1);
                scanf("%d", &G[i][j]);
                G[j][i] = G[i][j];
            }
            else
            {
                G[i][j] = 0;
            }
        }
    }

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

    for (i = 1; i <= n; i++)
        x[i] = 0;

    printf("\nAll possible valid colorings:\n");
    mColoring(1);

    return 0;
}
```

Test Case 1:

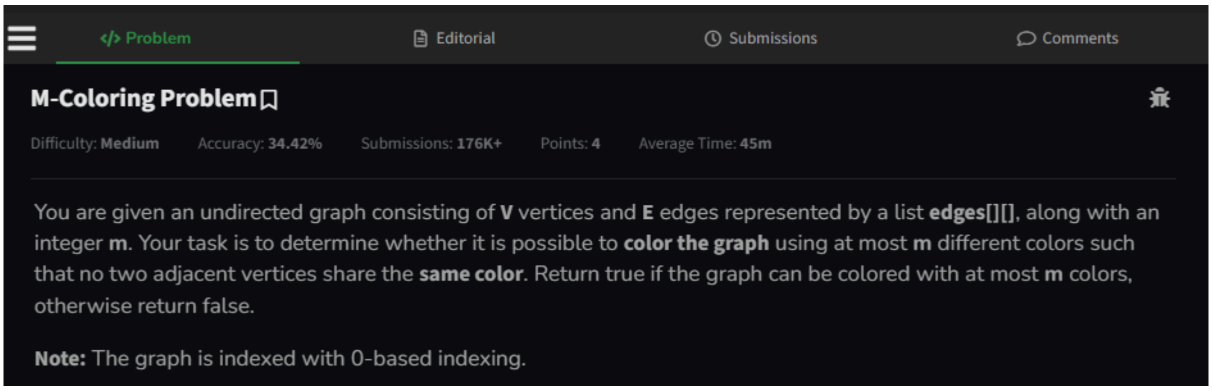


```
Enter number of vertices: 6
Enter adjacency matrix (6 x 6):
[A][B] = 1
[A][C] = 1
[A][D] = 0
[A][E] = 1
[A][F] = 1
[B][C] = 1
[B][D] = 0
[B][E] = 0
[B][F] = 0
[C][D] = 1
[C][E] = 0
[C][F] = 1
[D][E] = 1
[D][F] = 0
[E][F] = 1
Enter number of colors: 3

All possible valid colorings:

Valid coloring: 1 -->  2 -->  3 -->  1 -->  3 -->  2
Valid coloring: 1 -->  2 -->  3 -->  2 -->  3 -->  2
Valid coloring: 1 -->  3 -->  2 -->  1 -->  2 -->  3
Valid coloring: 1 -->  3 -->  2 -->  3 -->  2 -->  3
Valid coloring: 2 -->  1 -->  3 -->  1 -->  3 -->  1
Valid coloring: 2 -->  1 -->  3 -->  2 -->  3 -->  1
Valid coloring: 2 -->  3 -->  1 -->  2 -->  1 -->  3
Valid coloring: 2 -->  3 -->  1 -->  3 -->  1 -->  3
Valid coloring: 3 -->  1 -->  2 -->  1 -->  2 -->  1
Valid coloring: 3 -->  1 -->  2 -->  3 -->  2 -->  1
Valid coloring: 3 -->  2 -->  1 -->  2 -->  1 -->  2
Valid coloring: 3 -->  2 -->  1 -->  3 -->  1 -->  2
PS D:\Sem_III\DAA_Laboratory>
```

Test Case 2:



```
Enter number of vertices: 5
Enter adjacency matrix (5 x 5):
[A][B] = 1
[A][C] = 1
[A][D] = 1
[A][E] = 1
[B][C] = 1
[B][D] = 1
[B][E] = 1
[C][D] = 1
[C][E] = 1
[D][E] = 1
Enter number of colors: 1

All possible valid colorings:
PS D:\Sem_III\DAA_Laboratory>
```

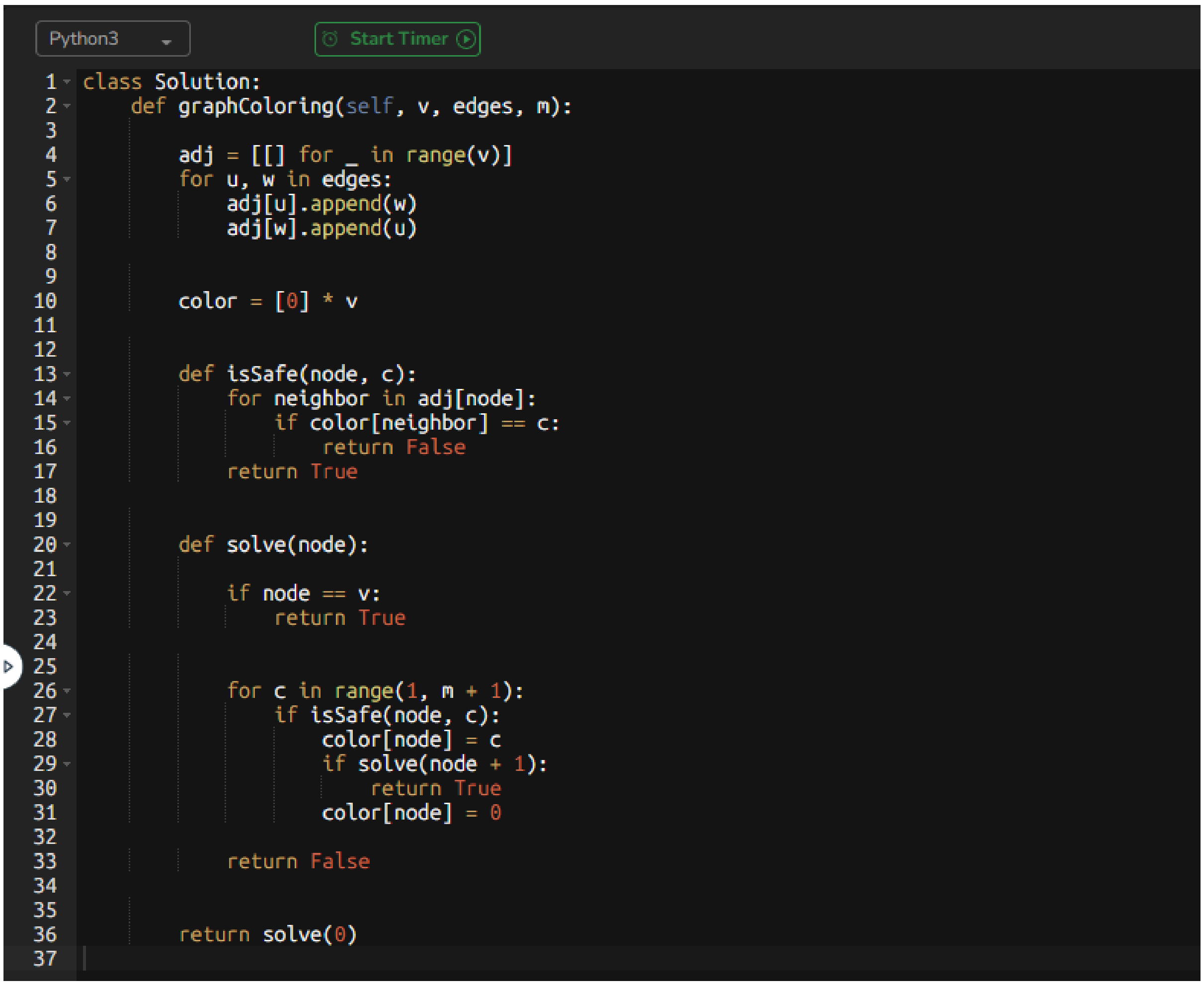


M-Coloring Problem

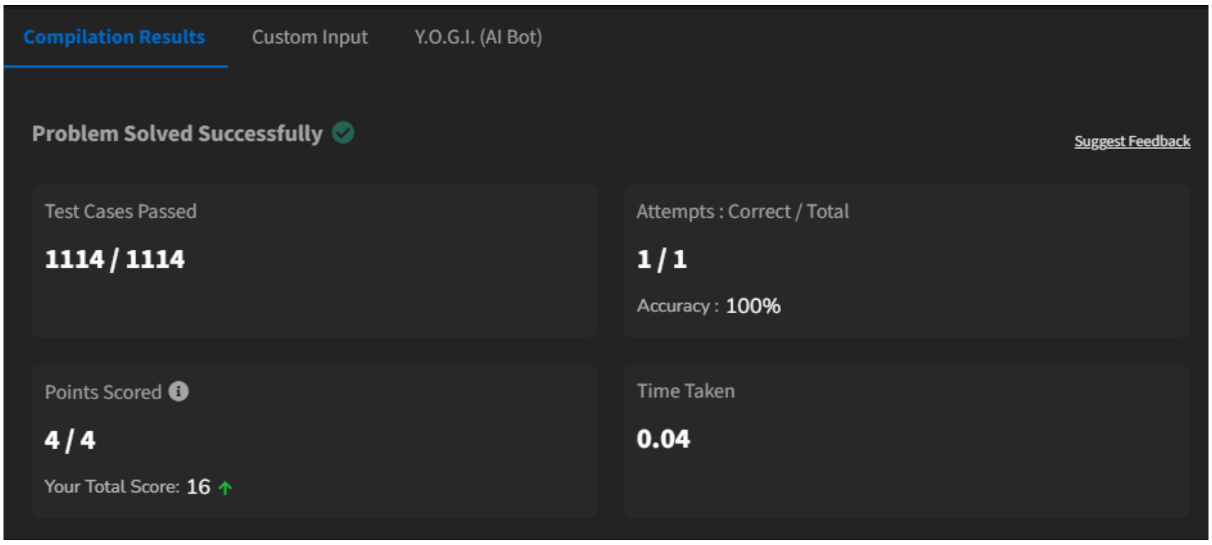
Difficulty: **Medium** Accuracy: **34.42%** Submissions: **176K+** Points: **4** Average Time: **45m**

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.



```python
class Solution:
    def graphColoring(self, v, edges, m):

        adj = [[] for _ in range(v)]
        for u, w in edges:
            adj[u].append(w)
            adj[w].append(u)


        color = [0] * v


        def isSafe(node, c):
            for neighbor in adj[node]:
                if color[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):
                    color[node] = c
                    if solve(node + 1):
                        return True
                    color[node] = 0

            return False


        return solve(0)

```



Problem Solved Successfully

Test Cases Passed

1114 / 1114

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 16

Time Taken

0.04

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