Design and Analysis of Algorithms Lab

Name: Arya Bodkhe

Section-A4 Batch: B-1

Roll No.:09

PRACTICAL NO. 8

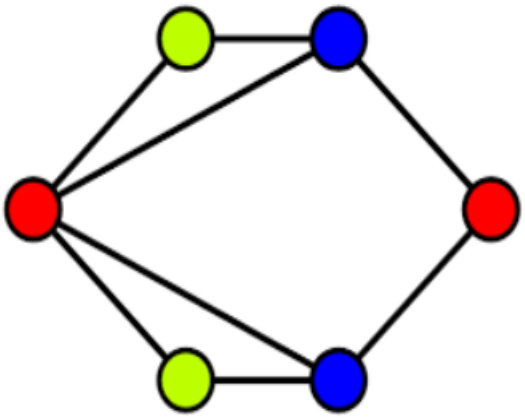
Aim: Implement Graph Colouring algorithm use Graph colouring concept.

Problem Statement:

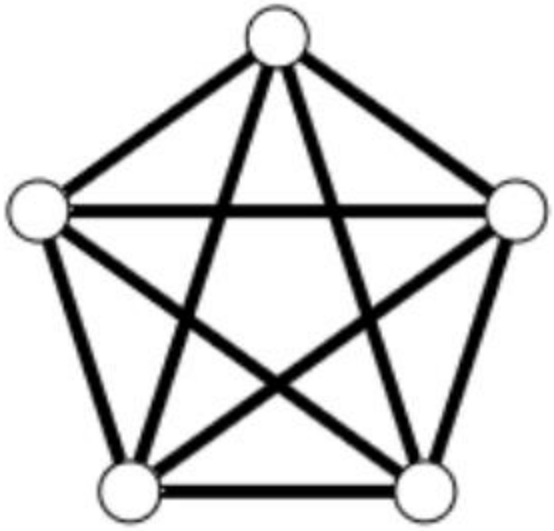
A GSM is a cellular network with its entire geographical range divided into hexadecimal cells. Each cell has a communication tower which connects with mobile phones within cell. Assume this GSM network operates in different frequency ranges. Allot frequencies to each cell such that no adjacent cells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

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



Graph –2:



CODE:

```c
#include <stdio.h>
#define MAX 10

void NextValue(int k, int n, int m, int G[MAX][MAX], int x[MAX]) {
    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, int n, int m, int G[MAX][MAX], int x[MAX]) {
    while (1) {
        NextValue(k, n, m, G, x);
        if (x[k] == 0)
            return;
        if (k == n) {
            printf("Valid coloring: ");
            for (int i = 1; i <= n; i++)
```

```
                printf("%d ", x[i]);
            printf("\n");
        } else {
            mColoring(k + 1, n, m, G, x);
        }
    }
}

int main() {
    int n, m;
    int G[MAX][MAX], x[MAX];
    int i, j;

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

    printf("Enter adjacency matrix :\n");
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            scanf("%d", &G[i][j]);

    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, n, m, G, x);

    return 0;
}
```

OUTPUT:

Graph-1:

```
Enter number of vertices: 6
Enter adjacency matrix:
0 1 1 0 1 1
1 0 1 0 0 0
1 1 0 1 0 0
0 0 1 0 1 0
1 0 0 1 0 1
1 0 0 0 1 0
Enter number of colors: 3

All possible valid colorings:
Valid coloring: 1 2 3 1 2 3
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 1 3 2
Valid coloring: 1 3 2 3 2 3
Valid coloring: 2 1 3 1 3 1
Valid coloring: 2 1 3 2 1 3
Valid coloring: 2 1 3 2 3 1
Valid coloring: 2 3 1 2 1 3
Valid coloring: 2 3 1 2 3 1
Valid coloring: 2 3 1 3 1 3
Valid coloring: 3 1 2 1 2 1
Valid coloring: 3 1 2 3 1 2
Valid coloring: 3 1 2 3 2 1
Valid coloring: 3 2 1 2 1 2
Valid coloring: 3 2 1 3 1 2
Valid coloring: 3 2 1 3 2 1
```