**Name:Mansi Jumde**

**Class:A3**

**Batch:B1**

**Roll no.:09**

**DAA Lab**

**Practical 8**

**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 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-**

**Code:**

#include <stdio.h>

#include <stdbool.h>

#define V 5

void printSolution(int color[]) {

printf("\nColor assigned to vertices:\n");

for (int i = 0; i < V; i++)

printf("Vertex %d ---> Color %d\n", i, color[i]);

}

bool isSafe(int v, bool graph[V][V], int color[], int c) {

for (int i = 0; i < V; i++)

if (graph[v][i] && color[i] == c)

return false;

return true;

}

bool graphColoringUtil(bool graph[V][V], int m, int color[], int v) {

if (v == V)

return true;

for (int c = 1; c <= m; c++) {

if (isSafe(v, graph, color, c)) {

color[v] = c;

if (graphColoringUtil(graph, m, color, v + 1))

return true;

color[v] = 0;

}

}

return false;

}

bool graphColoring(bool graph[V][V], int m) {

int color[V] = {0};

if (graphColoringUtil(graph, m, color, 0) == false) {

printf("\nSolution does not exist\n");

return false;

}

printSolution(color);

return true;

}

int main() {

bool graph[V][V] = {

{0, 1, 0, 1, 1},

{1, 0, 1, 0, 0},

{0, 1, 0, 1, 1},

{1, 0, 1, 0, 0},

{1, 0, 1, 0, 0}

};

int m = 3;

graphColoring(graph, m);

return 0;

}

**Output-**



**Graph 2-**

**Code-**

#include <stdio.h>

#include <stdbool.h>

#define V 5

void printSolution(int color[]) {

printf("\nColor assigned to vertices:\n");

for (int i = 0; i < V; i++)

printf("Vertex %d ---> Color %d\n", i, color[i]);

}

bool isSafe(int v, bool graph[V][V], int color[], int c) {

for (int i = 0; i < V; i++)

if (graph[v][i] && color[i] == c)

return false;

return true;

}

bool graphColoringUtil(bool graph[V][V], int m, int color[], int v) {

if (v == V)

return true;

for (int c = 1; c <= m; c++) {

if (isSafe(v, graph, color, c)) {

color[v] = c;

if (graphColoringUtil(graph, m, color, v + 1))

return true;

color[v] = 0;

}

}

return false;

}

bool graphColoring(bool graph[V][V], int m) {

int color[V] = {0};

if (graphColoringUtil(graph, m, color, 0) == false) {

printf("\nSolution does not exist\n");

return false;

}

printSolution(color);

return true;

}

int main() {

bool graph[V][V] = {

{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}

};

int m = 5;

graphColoring(graph, m);

return 0;

}

**Output-**

