**Practical No : 8**

**Name :** Samrudhi Pawar

**Roll No :** 44

**Title :**Develop a program to implement Graph Coloring using backtracking

CO1,CO3 method.

**Program :**

// C++ program for solution of M

// Coloring problem using backtracking

#include <bits/stdc++.h>

using namespace std;

// Number of vertices in the graph

#define V 4

void printSolution(int color[]);

/\* A utility function to check if

the current color assignment

is safe for vertex v i.e. checks

whether the edge exists or not

(i.e, graph[v][i]==1). If exist

then checks whether the color to

be filled in the new vertex(c is

sent in the parameter) is already

used by its adjacent

vertices(i-->adj vertices) or

not (i.e, color[i]==c) \*/

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

{

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

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

return false;

return true;

}

/\* A recursive utility function

to solve m coloring problem \*/

bool graphColoringUtil(bool graph[V][V], int m, int color[],

int v)

{

/\* base case: If all vertices are

assigned a color then return true \*/

if (v == V)

return true;

/\* Consider this vertex v and

try different colors \*/

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

{

/\* Check if assignment of color

c to v is fine\*/

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

{

color[v] = c;

/\* recur to assign colors to

rest of the vertices \*/

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

return true;

/\* If assigning color c doesn't

lead to a solution then remove it \*/

color[v] = 0;

}

}

/\* If no color can be assigned to

this vertex then return false \*/

return false;

}

/\* This function solves the m Coloring

problem using Backtracking. It mainly

uses graphColoringUtil() to solve the

problem. It returns false if the m

colors cannot be assigned, otherwise

return true and prints assignments of

colors to all vertices. Please note

that there may be more than one solutions,

this function prints one of the

feasible solutions.\*/

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

{

// Initialize all color values as 0.

// This initialization is needed

// correct functioning of isSafe()

int color[V];

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

color[i] = 0;

// Call graphColoringUtil() for vertex 0

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

{

cout << "Solution does not exist";

return false;

}

// Print the solution

printSolution(color);

return true;

}

/\* A utility function to print solution \*/

void printSolution(int color[])

{

cout << "Solution Exists:"

<< " Following are the assigned colors"

<< "\n";

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

cout << " " << color[i] << " ";

cout << "\n";

}

// Driver code

int main()

{

/\* Create following graph and test

whether it is 3 colorable

(3)---(2)

| / |

| / |

| / |

(0)---(1)

\*/

bool graph[V][V] = {

{0, 1, 1, 1},

{1, 0, 1, 0},

{1, 1, 0, 1},

{1, 0, 1, 0},

};

// Number of colors

int m = 3;

// Function call

graphColoring(graph, m);

return 0;

}

/\* Output:

Solution Exists: Following are the assigned colors

1 2 3 2

\*/