**Experiment No: 6**

**GRAPH COLOURING ALGORITHM**

Aim: Write a program to implement graph colouring algorithm using backtracking.

Theory:

**Backtracking**

The Backtracking is an algorithmic-method to solve a problem with an additional way. It uses a recursive approach to explain the problems. Backtracking is needed to find all possible combination to solve an optimization problem.

It uses recursive calling to find the solution by building a solution step by step increasing values with time. It removes the solutions that doesn't give rise to the solution of the problem based on the constraints given to solve the problem.

**Graph coloring**

The graph coloring problem is to discover whether the nodes of the graph G can be covered in such a way, that no two adjacent nodes have the same color yet only m colors are used. This graph coloring problem is also known as M-colorability decision problem.

The M – colorability optimization problem deals with the smallest integer m for which the graph G can be colored. The integer is known as a chromatic number of the graph.

**Graph coloring problem** can also be solved using a state space tree, whereby applying a backtracking method required results are obtained.

For solving the **graph coloring problem**, we suppose that the graph is represented by its adjacency matrix G[ 1:n, 1:n], where, G[ i, j]= 1 if (i, j) is an edge of G, and G[i, j] = 0 otherwise.

The colors are represented by the integers 1, 2, ..., m and the solutions are given by the n-tuple (x1, x2, x3, ..., xn), where x1 is the color of node i.

**Algorithm for Graph coloring problem**

Algorithm mcoloring ( k )

{

repeat

{

// generate all legal assignments for x[k],

nextValue (k); // assign to x[k] a legal color.

If ( x[k] = 0 ) then

return; // no new color possible

If (k = n) then // at most m colors have been used to color the n vertices.

Write (x[1 : n ]);

Else mcoloring (k + 1);

}

Until (false);

}

Algorithm nextValue( k)

{

repeat

{

x[k]=(x[k]+1)%(m+1);// next highest color

if(x[k]==0) // All colors have been used

{

return;

}

for(j=1;j<=n; j++)

{

//Check if this color is distinct from adjacent colors

if(g[k][j]==1&&x[k]==x[j])//If (k,j) is an edge and if adjacent vertices have the same color.

break;

}

if(j==(n+1)) // New color found

{

return;

}

}until(false);// Otherwise try to find another color

}

Conclusion: Graph colouring algorithm using backtracking was studied and implemented successfully.