Experiment 8

Aim: Implement Graph Coloring Problem with Backtracking Approach.

8.1 CO Attained: CO2,CO4 and CO5

8.2 Objective:

The problem is to find if it is possible to assign nodes with m different colors, such that no two adjacent vertices of the graph are of the same colors. If the solution exists, then display which color is assigned on which vertex.

8.3 Resources: Turbo c/Dev C++

8.4 Program Logic:

In graph theory, graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices are of the same color; this is called a vertex coloring.

```
Algorithm mColoring(k)
// This algorithm was formed using the recursive backtracking
// schema. The graph is represented by its boolean adjacency
// matrix G[1 : n, 1 : n]. All assignments of 1, 2, ..., m to the
// vertices of the graph such that adjacent vertices are
// assigned distinct integers are printed. k is the index
// of the next vertex to color.
{
    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: Finding all m-colorings of graph

8.5 Procedure:

1. Create: Open Dev C++/C and write a program after that save the program with the .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

8.6 Program Code:

```c
#include<stdio.h>
#include<conio.h>
static int m, n;

static int c=0;

static int count=0;

int g[50][50];

int x[50];
void nextValue(int k);

void GraphColoring(int k);

  void main() {

int i, j;
int temp;
clrscr();
printf("\nEnter the number of nodes: " );
scanf("%d", &n);

/*
printf("\nIf edge exists then enter 1 else enter 0 \n");

for(i=1; i<=n; i++)
 {

  x[i]=0;

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

{

   if(i==j)

    g[i][j]=0;

   else
{

    printf("%d -> %d: " , i, j);

    scanf("%d", &temp);
```

```
g[i][j]=g[j][i]=temp;

   } } }*/
printf("\nEnter Adjacency Matrix:\n");

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

  {
scanf("%d", &g[i][j]);

} }

printf("\nPossible Solutions are\n");

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

 {

  if(c==1)

  {
   break;
  }
GraphColoring(1);
 }
printf("\nThe chromatic number is %d", m-1);

 //in for loop, m gets incremented first and then the condition is checked

 //so it is m minus 1

printf("\nThe total number of solutions is %d", count);

getch();

}
void GraphColoring(int k)
{
 int i;
while(1)

 {

nextValue(k);
```

```
 if(x[k]==0)

 {

  return;
 }

if(k==n)
{

 c=1;

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

 {

printf("%d ", x[i]);

 }

 count++;

printf("\n");

}
else

GraphColoring(k+1);
}}
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]==1&&x[k]==x[j])

   break;
 }
 if(j==(n+1))
 {
return;
} }}
```

8.7Conclusion:

```
Enter the number of nodes: 5
Enter Adjacency Matrix:
0 1 0 1 1

1 0 1 1 0

0 1 0 1 1

1 1 1 0 1

1 0 1 1 0
Possible Solutions are:
1 2 1 3 2

1 3 1 2 3

2 1 2 3 1

2 3 2 1 3

3 1 3 2 1

3 2 3 1 2
The chromatic number is 3

The total number of solutions is 6
```

8.8Analysis: Time Complexity: $O(m^V)$. There is a total $O(m^V)$ combination of colors

Auxiliary Space: O(V). Recursive Stack of graph coloring(…) function will require O(V) space.

8.9Lab Viva Questions:

1. **What are backtracking algorithms?**
2. What do you mean by Chromatic number in graph?
3. What is the difference between recursion and backtracking?