Experiment No: 13 **Date:** 06/05/2021

Aim: Implementation of Graph Coloring problem

Theory:

Graph Coloring

- Graph Coloring is also called as Vertex Coloring.
- It ensures that there exists no edge in the graph whose end vertices are colored with the same color.
- Such a graph is called as a Properly colored graph.

Backtracking Approach

- The idea is to assign colors one by one to different vertices, starting from the vertex 0.
- Before assigning a color, check for safety by considering already assigned colors to the adjacent vertices i.e., check if the adjacent vertices have the same color or not.
- If there is any color assignment that does not violate the conditions, mark the color assignment as part of the solution.
- If no assignment of color is possible then backtrack and return false.

EXPERIMENT 13

Complexity

- Time Complexity: ***O(m^V)***
  - There is total O(m^V) combination of colors.
  - So, time complexity is O(m^V).
  - The upper bound time complexity remains the same but the average time taken will be less.
- Space Complexity: ***O(V)***
  - Recursive Stack of graphColoring(…) function will require O(V) space.

Applications of Graph Coloring

- Map Coloring
- Scheduling the tasks
- Preparing Time Table
- Assignment
- Conflict Resolution
- Sudoku

EXPERIMENT 13

Algorithm Writing

- Create a recursive function that takes the graph, current index, number of vertices, and output color array.
- If the current index is equal to the number of vertices. Print the color configuration in output array.
- Assign a color to a vertex (1 to m).
- For every assigned color, check if the configuration is safe, (i.e., check if the adjacent vertices do not have the same color) recursively call the function with next index and number of vertices
- If any recursive function returns true break the loop and return true.
- If no recursive function returns true then return false.

Algorithm

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

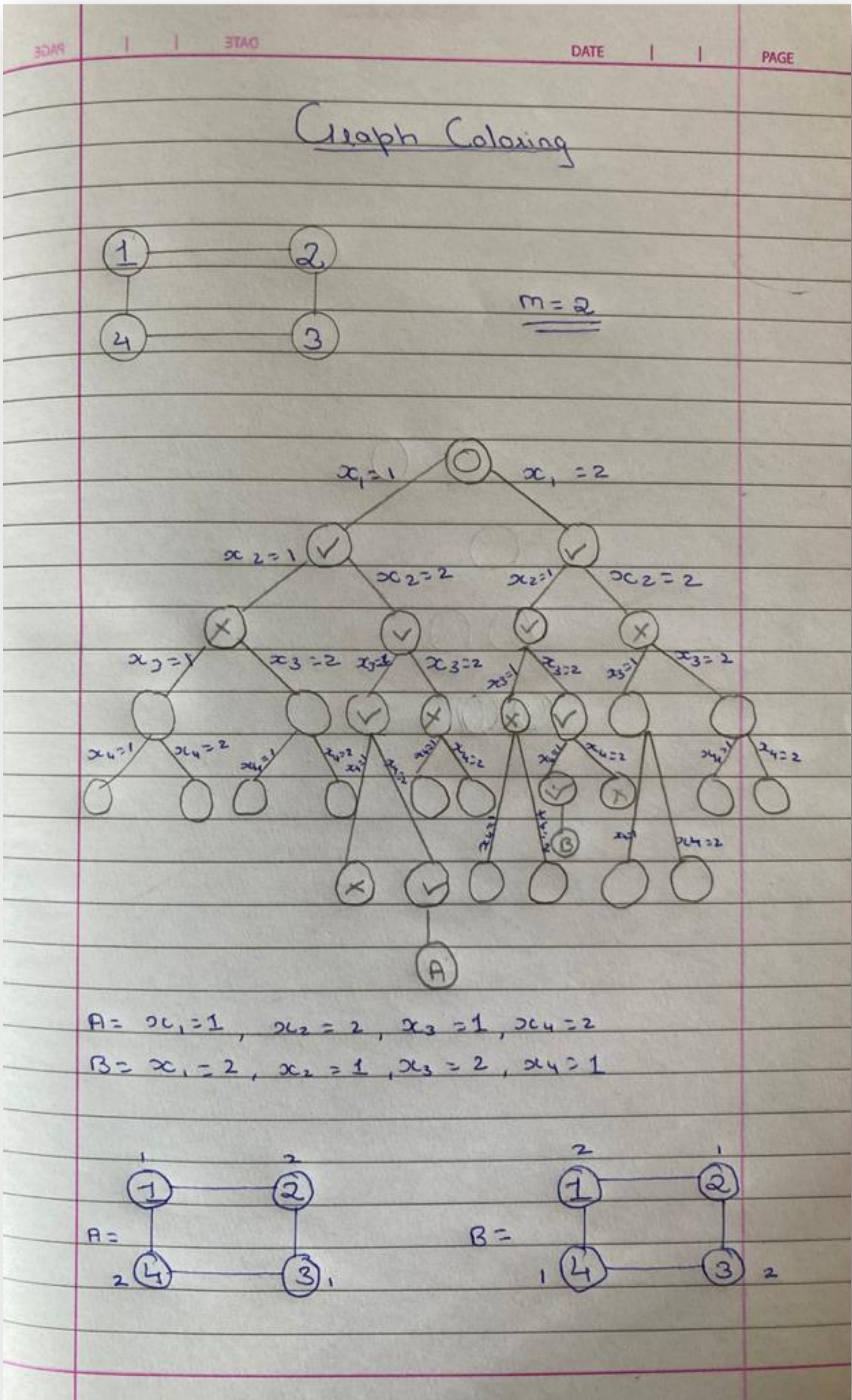
// of the next vertex to color.

EXPERIMENT 13

```
{
    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);
}
```

EXPERIMENT 13

Tracing with Example



EXPERIMENT 13

Program

```
#include<bits/stdc++.h>
using namespace std;


void mColoring(int p,vector<int> q);
void nextvalue(vector<int> &q,int p);


void print(vector<int> q);
vector<vector<int>> v;
int r;
int s;
int done=0;
int main()
{
  cout<<"Enter The No of vertices: ";
  cin>>r;
  v.resize(r+1,vector<int>(r+1));
  for(int i=1;i<=r;i++)
    for(int j=1;j<=r;j++)
                v[i][j] = 0;
```

EXPERIMENT 13

```
    cout<<"Enter The value of m: ";

    cin>>s;

    cout<<"Enter The Number of edges: ";

    int e;

    cin>>e;

    cout<<"Enter The Values of Edges: "<<endl;

    for(int i=1;i<=e;i++)

    {

        int q,y,w;

        cin>>q>>y;

        v[q][y] = 1;

        v[y][q] = 1;

    }



    vector<int> q(r+1,0);

    mColoring(1,q);

    if(done==0) cout<<"No Possible Solutions"<<endl;

}


void mColoring(int p,vector<int> q)
```

EXPERIMENT 13

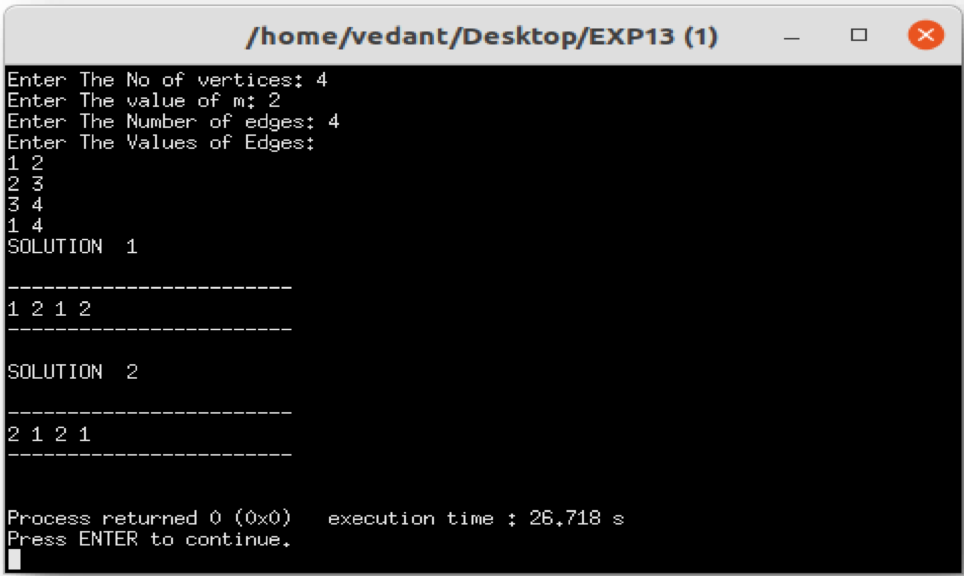
```
{

    do {

        nextvalue(q,p);
        if(q[p]==0) return;
        if(p==r) print(q);
        else mColoring(p+1,q);
    }while(true);
}

void nextvalue(vector<int> &q,int p)
{
    do{

        q[p] = (q[p] +1)%(s+1);
        if(q[p]==0) return;
        int j;
        for(j=1;j<=r;j++)
        {
```

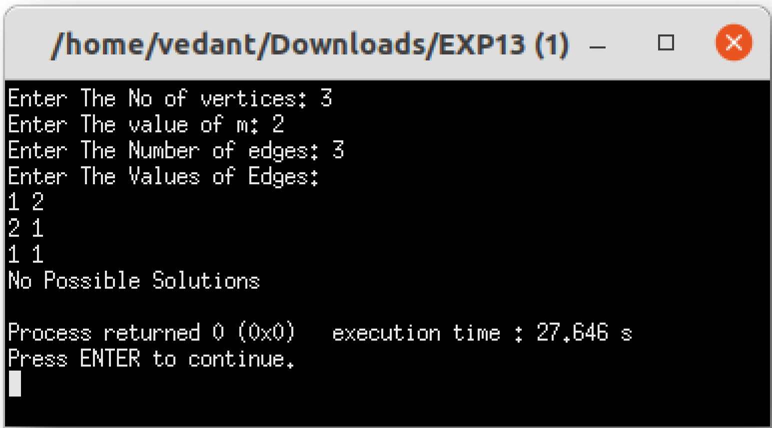
EXPERIMENT 13

```
                    if(v[p][j]!=0&&q[j]==q[p])
                    break;

                }

                if(j==r+1) return;
        }while(true);
}
void print(vector<int> q)
{
        done++;

        cout<<"SOLUTION  "<<done<<endl;
        cout<<endl<<"------------------------"<<endl;
        for(int i=1;i<=r;i++)
        {
                cout<<q[i]<<" ";
        }
        cout<<endl<<"------------------------\n"<<endl;
}
```

EXPERIMENT 13

Output





Conclusion

- Detailed concept of Graph Coloring problem was studied successfully.
- Program using Graph Coloring Algorithm was executed successfully.