Batch: B3 Roll No.: 16010121110

Experiment No. ____8__

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Graph Colouring Backtracking Algorithm

Objective: To learn the Backtracking strategy of problem solving for Graph Colouring problem

CO to be achieved:

CO 2	Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajasekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algorithms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://www.math.utah.edu/~alfeld/queens/queens.html**
4. **http://www-isl.ece.arizona.edu/ece175/assignments275/assignment4a/Solving%208%20queen%20problem.pdf**
5. **http://www.slideshare.net/Tech_MX/8-queens-problem-using-back-tracking**
6. **http://www.mathcs.emory.edu/~cheung/Courses/170.2010/Syllabus/Backtracking/8queens.html**
7. **http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/**
8. **http://www.hbmeyer.de/backtrack/achtdamen/eight.htm**

Pre Lab/ Prior Concepts:
Data structures, Concepts of algorithm analysis

Historical Profile:

Given an undirected graph and a number m, determine if the graph can be colored with at most m colors such that no two adjacent vertices of the graph are colored with the same color. Here coloring of a graph means assignment of colors to all vertices.

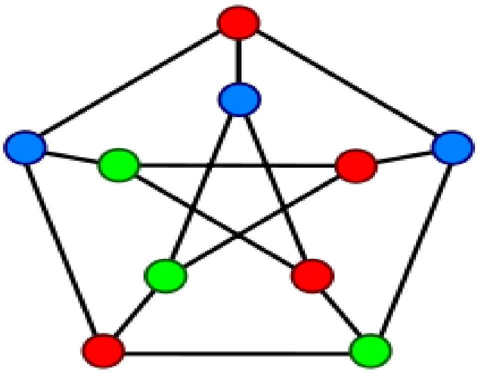
Input:

1) A 2D array graph [V][V] where V is the number of vertices in the graph and graph[V][V] is the adjacency matrix representation of the graph.

Output:

An array color [V] that should have numbers from 1 to m. color[i] should represent the color assigned to the ith vertex. The code should also return false if the graph cannot be colored with m colors.

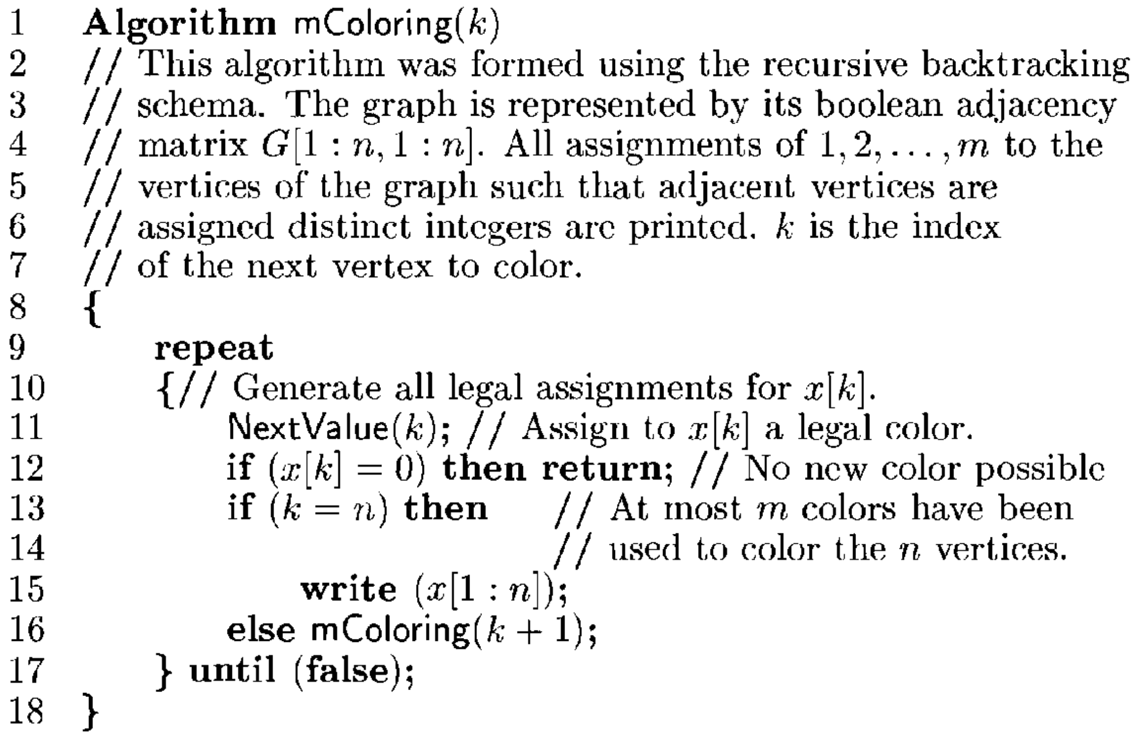
Following is an example graph that can be colored with 3 colors.



New Concepts to be learned:

Application of algorithmic design strategy to any problem, Backtracking method of problem solving Vs other methods of problem solving problem graph colouring and its applications.

Algorithm Graph colouring Problem-



```
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);
}
```

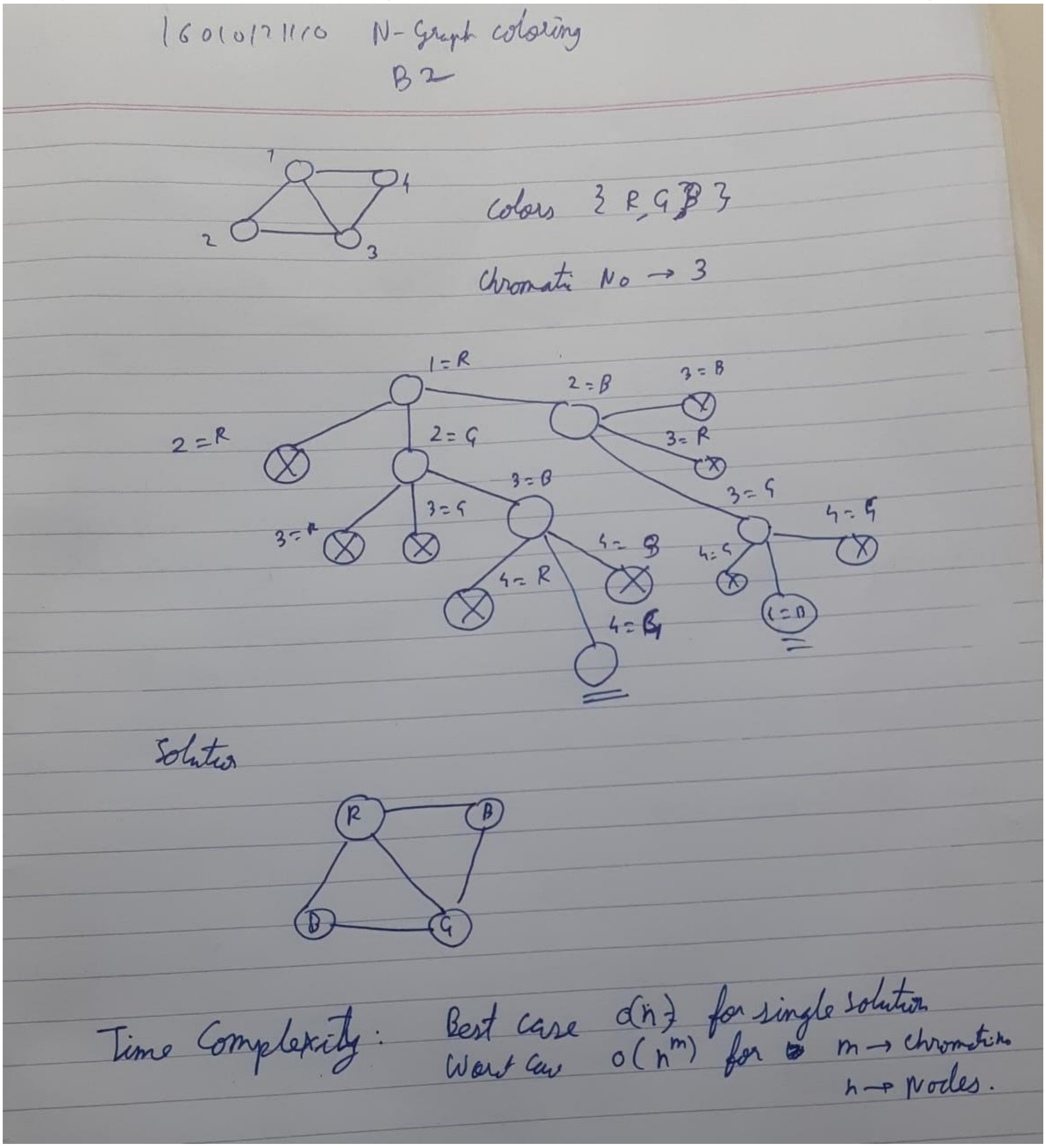
Example Graph Colouring Problem:

```
/*************************************************************************
****
Need to put number of colors in colorsnum feild;
*************************************************************************
****/
public class Main
{
        public static void main(String[] args) {
                System.out.println("Hello World");
                int [][] adjustancyMatrix={{0, 1, 1, 1},
                 {1, 0, 1, 0},
                 {1, 1, 0, 1},
                 {1, 0, 1, 0}};
     int colorsnum=3;
                int [] color = new int [4];
                if(color(0,adjustancyMatrix,color,colorsnum)==false){
                 System.out.println("In sufficiant number of colors")   ;
                }


        }
        static boolean color(int node, int [][] adjustancyMatrix,int[]color, int colorsnum){
          for(int i=1;i<=colorsnum;i++){ //iterate over colors
          if(checkAdjusantVerticesColor(i,node,adjustancyMatrix,color)==true){
             //color not used, set color
             color[node]=i;
             System.out.println("Node "+node+" is colored "+i);



          if(AllNodesColored(color)==true){
             System.out.println("All nodes exausted");

             return true;

          }
          else{
             //recur
             if(color(node+1,adjustancyMatrix,color,colorsnum)==true){//adjusant node
                return true;
             }
             else{
                //forward tracking failed, so try next color
               System.out.println("Went wrong!");

                continue;
             }
          }
          }
          }
          //all color exausted.
          //backtrack
          System.out.println("All colors exausted");
          return false;

       }
       static  boolean  checkAdjusantVerticesColor(int  currrentColor,int  node,  int  [][]
adjustancyMatrix,int[]color){
          for(int i=0;i<adjustancyMatrix[0].length;i++){ //iterate nodes

             if(color[i]==currrentColor & adjustancyMatrix[node][i]==1){
                return false;
             }
          }
          return true;
       }
       static boolean AllNodesColored(int [] color){
          for (int i=0;i<color.length;i++){ //iterate nodes
              if(color[i]==0){

                return false;
```

```
                    }
                }
                return true;
            }
}
```

Output

```
Hello World
Node 0 is colored 1
Node 1 is colored 2
Node 2 is colored 3
Node 3 is colored 2
All nodes exausted
```



16010121110 N-Graph coloring
B2

colors {R, G, B}

Chromatic No → 3

1
4
2
3
1=R
2=R
2=B
3=B
2=G
3=R
3=B
3=G
3=R
3=G
4=G
4=B
4=G
4=R
4=G
1=B

Solution

R
B
B
G

Time Complexity: Best case $O(n)$ for single solution
Worst case $O(n^m)$ for m → chromatic
n → nodes.

Conclusion:

Thus we have implemented n graph coloring using backtracking. We understood the concept behind the algorithm, how backtracking works. We used chromatic number to find out the best way to color graphs. This has various applications like mobile frequency adjustment, register allocation etc.