Topic: Backtracking

Theory: In many applications of the backtrack method, the desired solution is expressible as an n-tuple *(x1,...,Xn)*, where the x*i* are chosen from some finite set Si. Often the problem to be solved calls for finding one vector that maximizes (or minimizes or satisfies) a *criterion function P(x1,..... . , xn). Sometime*s it seeks all vectors that satisfy *P*. For example, sorting the array of integers in. *a[1* : n] is a problem whose solution is expressible by an *n- tuple, w*here x*i* is the index in *a* of the ith smallest element. The criterion function P is the inequality *a[xi]* ≤ *a[xi+1]* for 1 ≤ i < *n*. The set *Si* is finite and includes the integers 1 through *n*. Though sorting is not usually one of the problems solved by backtracking, it is one example of a familiar problem whose solution can be formulated as an n-tuple.

Control abstraction:

```
void Backtrack( int k )

// This is a schema that describes the backtracking process //using recursion. On entering, the
first k-1 values x[1], x[2], //…., x[k-1] of the solution vector x[1:n] have been //assigned. x[]
and n are global.
{
        for (each x[k] such that x[k] Є T(x[1], …, x[k-1])
    {

            if (Bk  (x[1], x[2], …, x[k]))
             {

               if (x[1], x[2], …, x[k] is a path to an answer node)

                       output x[1:k];

               if (k < n) Backtrack(k+1);
             }
        }

    }
```

Batch: E-2 Roll No.: 16010123325

Experiment No. ___8___

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

Signature of the Staff In-charge with date

Title: Study, Implementation, and Analysis of Graph Coloring Problem.

Objective: To learn the Backtracking strategy of problem solving for Graph Coloring 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.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**

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

Historical Profile:The Graph Coloring Problem is a classical problem in graph theory and combinatorics with origins rooted in practical applications and mathematical curiosity. It has a rich history, spanning over two centuries, and remains a vibrant area of research due to its theoretical significance and real-world applications.

Origins and Early History

Map Coloring and the Four Color Theorem (1852):The problem of graph coloring originated from an attempt to color regions on maps so that no two adjacent regions share the same color. In 1852, Francis Guthrie, a British mathematician, conjectured the Four Color Theorem, stating that four colors are sufficient to color any map in a plane.

Graph Representation of Maps:In 1879, Arthur Cayley formulated the map coloring problem in terms of graph theory, representing regions of a map as vertices and adjacency as edges.

New Concepts to be learned:

Algorithm:

Backtracking Algorithm

The backtracking approach finds the optimal solution by trying all possible color assignments.

Steps:

1. **Try assigning each vertex a color from 1 to k, where k is the number of colors.**
2. **Backtrack if an assignment leads to a conflict (two adjacent vertices having the same color).**
3. **Continue until all vertices are colored or all possibilities are exhausted.**

Implementation(Code):

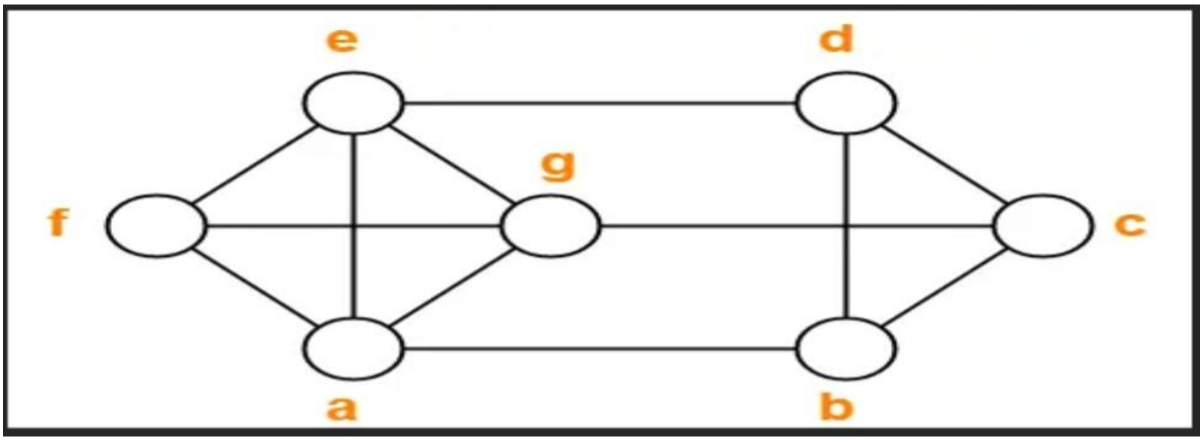
```java
import java.util.Scanner;

public class GraphColoring {

    int V;
    int[] color;


    public GraphColoring(int v) {
        V = v;
        color = new int[V];
    }

    boolean isSafe(int v, int[][] graph, int[] color, int c) {
        for (int i = 0; i < V; i++)
            if (graph[v][i] == 1 && color[i] == c)
                return false;
        return true;
    }

    boolean graphColoringUtil(int[][] graph, int m, int v) {
        if (v == V)
            return true;

        for (int c = 1; c <= m; c++) {
            if (isSafe(v, graph, color, c)) {
```

```
                color[v] = c;
                if (graphColoringUtil(graph, m, v + 1))
                    return true;
                color[v] = 0;
            }
        }
        return false;
    }


    boolean graphColoring(int[][] graph, int m) {
        if (!graphColoringUtil(graph, m, 0)) {
            System.out.println("Solution does not exist");
            return false;
        }

        System.out.println("Color assignment:");
        for (int i = 0; i < V; i++)
            System.out.println("Vertex " + i + " → Color " + color[i]);
        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of vertices: ");
        int v = sc.nextInt();

        int[][] graph = new int[v][v];
        System.out.println("Enter adjacency matrix:");
        for (int i = 0; i < v; i++) {
            for (int j = 0; j < v; j++) {
                graph[i][j] = sc.nextInt();
            }
        }

        System.out.print("Enter number of colors: ");
        int m = sc.nextInt();

        GraphColoring gc = new GraphColoring(v);
        gc.graphColoring(graph, m);
    }
}
```

Output:

```
Enter number of vertices: 7
Enter adjacency matrix:
0 1 0 0 1 1 1
1 0 1 1 0 0 0
0 1 0 1 0 0 1
0 1 1 0 1 0 0
1 0 0 1 0 1 1
1 0 0 0 1 0 1
1 0 1 0 1 1 0
Enter number of colors: 4
Color assignment:
Vertex 0 → Color 1
Vertex 1 → Color 2
Vertex 2 → Color 1
Vertex 3 → Color 3
Vertex 4 → Color 2
Vertex 5 → Color 3
Vertex 6 → Color 4
```

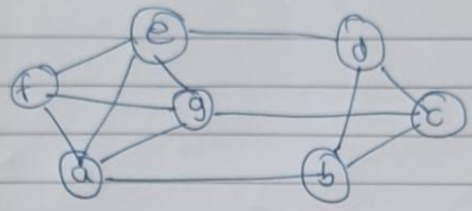
Example sum of subset Problem along with state space tree:



Analysis of Backtracking solution for :
Time Complexity : $O(k^n)$
k = number of colours **n = number of vertices**

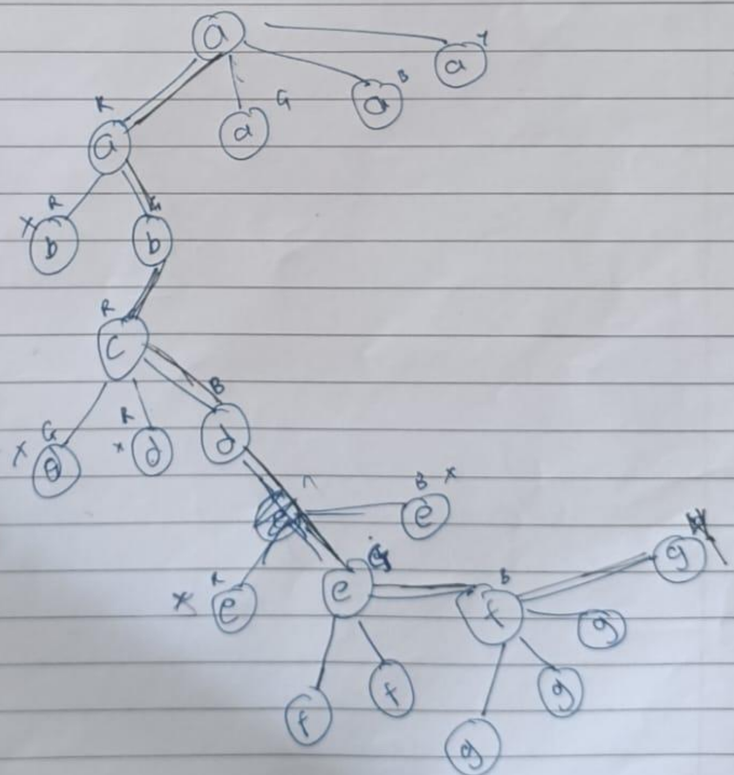
Page No.:
Date:

Graph Coloring



Lets start from a there will be 4 colors

1 → R , 2 → G , 3 → B , 4 → Y



T.C. = O(k^n) → no. of vertices
↳ no. of colors

12:44pm

17/4/25

Conclusion:

The above experiment highlights graph colouring using backtracking method, by efficient use of pruning to find the optimal approach.