GRAPH COLORING

Submitted by

Ninaad Arora [RA2111032010014]
Parth Galhotra [RA2111032010029]

Under the Guidance of

Dr.A Prabhu Chakkaravarthy

Assistant Professor, Department of Networking And Communications

In partial satisfaction of the requirements for the degree of

BACHELORS OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in Internet Of Things(IOT)

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

May 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that the Course 18CSE204J-Design and Analysis of Algorithm Assignment Report titled **"GRAPH COLORING"** is the bonafide work done by **NINAAD ARORA [RA2110320014] and PARTH GALHOTRA [RA2111032010029]** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE
Faculty In-Charge
Dr.A Prabhu Chakkaravarthy
Assistant Professor
Department of Networking And Communications,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

HEAD OF THE DEPARTMENT
Dr. Annapurani Panaiyappan. K
Professor and Head ,
Department of Networking and Communications
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

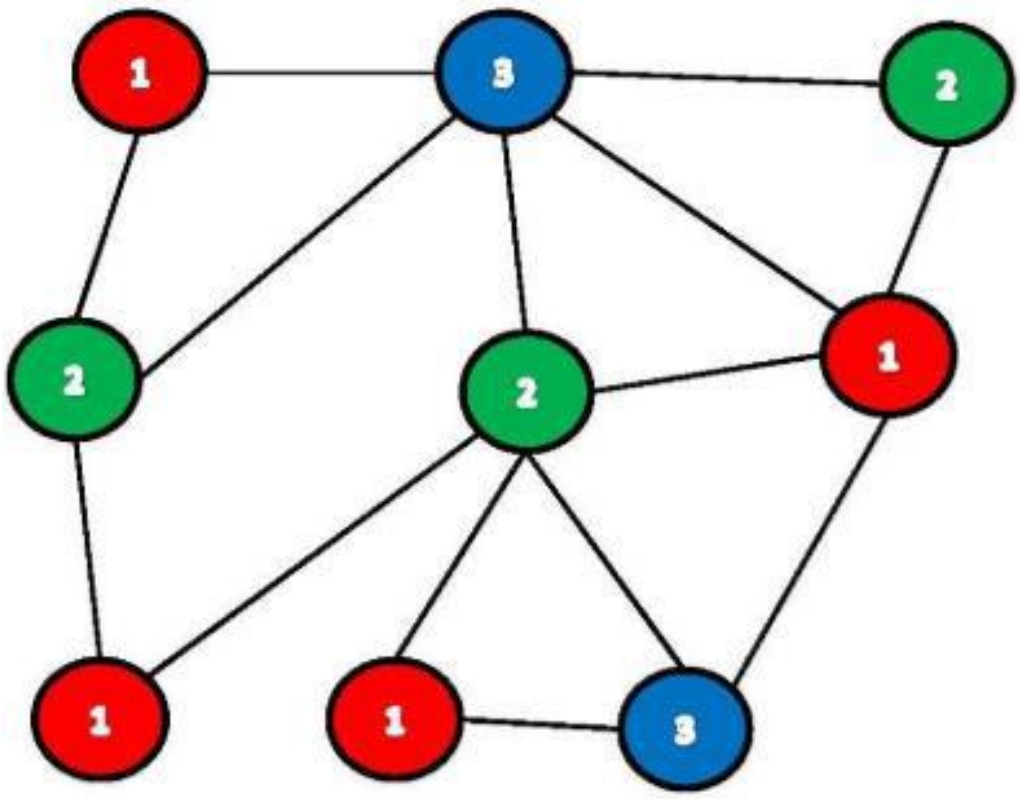
Index

GRAPH COLORING

- Graph coloring is a problem in graph theory that involves assigning colors to the vertices of a graph, subject to certain constraints. The goal is to color the vertices of the graph in such a way that no two adjacent vertices have the same color.
- The problem of graph coloring has a wide range of applications in computer science, operations research, and other fields. For example, it can be used to model scheduling problems, map coloring problems, and register allocation in compilers.
- In addition to its practical applications, graph coloring is also an important area of research in theoretical computer science. There are many interesting and unsolved problems related to graph coloring, such as the chromatic number of a graph (the minimum number of colors needed to color the vertices of a graph), and the complexity of coloring graphs with specific structures or properties.

CONDITION:

- The main condition for graph coloring is that no two adjacent vertices (vertices connected by an edge)should have the same color.
- In other words, if two vertices are connected by an edge, they should be assigned different colors.
- The goal of graph coloring is to assign colors to the vertices in such a way that the fewest number of colors possible are used, while still satisfying this condition. The minimum number of colors required to color the vertices of a graph without any adjacent vertices having the same color is called the chromatic number of the graph.
- Graph coloring can also have additional constraints or conditions, depending on the specific problem being solved. For example, a graph coloring problem might have constraints on the colors that can be used or on the number of colors that can be used for certain groups of vertices.
- Overall, the main condition for graph coloring is that adjacent vertices must be colored differently, and the goal is to minimize the number of colors used to achieve this condition.



Different Approch Used :

1)Greedy Coloring: This is a simple and efficient method for coloring a graph. It starts with an arbitrary vertex and colors it with the first available color. Then it moves to the next uncolored vertex and assigns the first available color that is not already used by its adjacent vertices. This process is repeated until all vertices are colored.

2) Backtracking Coloring: This is a recursive method that tries to find a valid coloring of the graph by exploring all possible combinations of colors. It starts with an arbitrary vertex and assigns a color to it. Then it recursively tries to assign colors to the remaining vertices, checking at each step if the coloring is valid. If a coloring is found to be invalid, the algorithm backtracks and tries a different color for the last vertex until a valid coloring.

3) Dynamic graph coloring is the process of maintaining a valid coloring of a graph when the graph changes over time. This is an important problem in many applications where the graph is constantly changing, such as in network routing, traffic management, and resource allocation.

Best Approach: BACKTRACKING Method

It works by systematically searching through all possible colorings of the graph until a valid coloring is found. Although backtracking can be slow for large and complex graphs, it is guaranteed to find a valid coloring if one exists. Additionally, backtracking can be a good choice when there are constraints on the colors that can be used or when the objective is to find a coloring with a certain property, such as a minimal number of colors

Algorithm:

Choose an uncolored vertex of the graph.

1) For each possible color for the vertex:

a. Check if the color is valid (i.e., does not conflict with any adjacent vertices that have already been colored).

b. If the color is valid, assign the color to the vertex and move on to the next uncolored vertex.

c. If the color is not valid, try the next color.

2) If all possible colors have been tried for a vertex without success, backtrack to the previous vertex and try a different color for that vertex.

3) Repeat steps 1-3 until all vertices have been colored or there are no more possible colorings.

Pseudocode:

```
BacktrackColoring(G, colors):

for each vertex v in G:

set color[v] = UNCOLORED

backtrack(0, G, colors)

backtrack(v, G, colors):

if v == num_vertices(G):

return True

for each color in colors:

if isValidColor(v, color, G):

color[v] = color

if backtrack(v+1, G, colors):

return True

color[v] = UNCOLORED

return False

isValidColor(v, color, G):

for each neighbor u of v:
```

if color[u] == color:

return False

return True

CODE:

```
private static boolean isSafeToColor(int[][] graph, int[] color)
  {
     // check for every edge, if any adjacent edge has same color, return false
     for (int i = 0; i < V; i++)
        for (int j = i + 1; j < V; j++)
           if (graph[i][j] == 1 && color[j] == color[i])
              return false;
     return true;
  }


  private static void printColorArray(int[] color){
     System.out.println("Solution colors are: ")
     for(int i =0;i < color.length; i++){
        System.out.println(color[i]);
     }
  }
```

```
/**
* returns false if the m colors cannot be assigned, else,
* return true and print assignments of colors to all vertices.
* Note: There may be more than one solutions
* The function prints only one of the solutions.
* */
private static boolean graphColoring(int[][] graph, int m, int i, int[] color)
{
    // if current index becomes end
    if (i == V) {
        // check whether its safe to color
        if (isSafeToColor(graph, color)) {
            //If safe, print solution array and return true
            printColorArray(color);
            return true;
        }
        return false;
    }

    // Assign each color from 1 to m
    for (int j = 1; j <= m; j++) {
        color[i] = j;
```

```
            // Check for the rest vertices by recursion
            if (graphColoring(graph, m, i + 1, color))
                return true;

            color[i] = 0;
        }

        return false;
    }
}
```

OUTPUT

```
A Green

B Blue

C Green

D Blue


Process finished with exit code 0
```

TIME COMPLEXCITY:

O (m V) O(m^V) O(mV)

=>O(m^V)

SPACE COMPLEXCITY:

O(M)

Applications of graph coloring

- **Design a timetable.**
- **Sudoku**
- **Register allocation in the compiler**
- **Map coloring**
- **Mobile radio frequency assignment:**

RESULT: Hence the graph coloring problem has been implemented by using Backtracking approach.

References

- https://pencilprogrammer.com/algorithms/graph-coloring-problem/#:~:text=Using%20Backtracking%20Algorithm&text=In%20this%20approach%2C%20we%20color,the%20given%20graph%20are%20colored.

- https://www.gatevidyalay.com/tag/graph-coloring-using-backtracking/