Title: Graph Coloring Problem

**Ex. No.:** 2 **Reg. No.:** RA2011051010006

**Date:** 25/01/23 **Name:** Vittesh Raghavan

# Aim:

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

**Procedure/Algorithm:**

**m Coloring Problem using Backtracking:**

# 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 the 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 the 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

# Program:

# 

**Manual Output: Manual calculation for the example you have taken:**

The backtracking approach to solving the graph coloring problem can be to assign the colors one after the other to the various vertices. The coloring will start with the first index only but before assigning any color, we would first check if it satisfies the constraint or not (i.e. no two adjacent vertices have the same color). If the current color assignment does not violate the condition then add it into the solution else, backtrack by returning false.

# Define a recursive function that takes the current index (i), several vertices, and the color array for output.

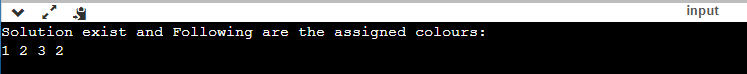
# Check if the current color is safe i.e. no two adjacent vertices have the same color. Then print the current color configuration and break the loop.

# Assign a color to the vertex. The range of assigned colors is from 1 to m.

# Now, for every color assigned, call the recursive function.

# If the recursive call returns true then the coloring is possible. So, break the loop and return true. Else, backtrack.

# Screenshot of output: Actual Output you get after executing your program:

****

**Result:**

M Graph Problem was executed and solved successfully.