Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for a graph coloring problem.

```
// Java program for the above approach

public class MyClassjava {

	// Number of vertices in the graph
	static int V = 4;

	/* A utility function to print solution */
	static void printSolution(int[] color)
	{
		System.out.println(
			"Solution Exists:"
			+ " Following are the assigned colors ");
		for (int i = 0; i < V; i++)
			System.out.print(" " + color[i]);
		System.out.println();
	}

	// check if the colored
	// graph is safe or not
	static boolean isSafe(boolean[][] graph, int[] color)
	{
		// check for every edge
		for (int i = 0; i < V; i++)
			for (int j = i + 1; j < V; j++)
				if (graph[i][j] && color[j] == color[i])
					return false;
		return true;
	}

	/* This function solves the m Coloring
	problem using recursion. It returns
```

```
false if the m colours cannot be assigned,
otherwise, return true and prints
assignments of colours to all vertices.
Please note that there may be more than
one solutions, this function prints one
of the feasible solutions.*/
static boolean graphColoring(boolean[][] graph, int m,
                                          int i, int[] color)
{
      // if current index reached end
      if (i == V) {

            // if coloring is safe
            if (isSafe(graph, color)) {

                  // Print the solution
                  printSolution(color);
                  return true;
            }
            return false;
      }

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

            // Recur of the rest vertices
            if (graphColoring(graph, m, i + 1, color))
                  return true;
            color[i] = 0;
      }
      return false;
}
```

```
    // Driver code
    public static void main(String[] args)
    {

        /* Create following graph and
            test whether it is 3 colorable
            (3)---(2)
            | / |
            | / |
            | / |
            (0)---(1)
            */
        boolean[][] graph = {
            { false, true, true, true },
            { true, false, true, false },
            { true, true, false, true },
            { true, false, true, false },
        };
        int m = 3; // Number of colors

        // Initialize all color values as 0.
        // This initialization is needed
        // correct functioning of isSafe()
        int[] color = new int[V];
        for (int i = 0; i < V; i++)
            color[i] = 0;

        // Function call
        if (!graphColoring(graph, m, 0, color))
            System.out.println("Solution does not exist");
    }
}
```

Python Code :

```
# Python3 program for the above approach

# Number of vertices in the graph
# define 4 4

# check if the colored
# graph is safe or not


def isSafe(graph, color):

    # check for every edge
    for i in range(4):
        for j in range(i + 1, 4):
            if (graph[i][j] and color[j] == color[i]):
                return False
    return True

# /* This function solves the m Coloring
# problem using recursion. It returns
# false if the m colours cannot be assigned,
# otherwise, return true and prints
# assignments of colours to all vertices.
# Please note that there may be more than
# one solutions, this function prints one
# of the feasible solutions.*/


def graphColoring(graph, m, i, color):

    # if current index reached end
    if (i == 4):

        # if coloring is safe
        if (isSafe(graph, color)):

            # Print the solution
            printSolution(color)
            return True
        return False

    # Assign each color from 1 to m
    for j in range(1, m + 1):
        color[i] = j

        # Recur of the rest vertices
        if (graphColoring(graph, m, i + 1, color)):
            return True
        color[i] = 0
    return False

# /* A utility function to print solution */


def printSolution(color):
    print("Solution Exists:" " Following are the assigned colors ")
    for i in range(4):
        print(color[i], end=" ")


# Driver code
if __name__ == '__main__':
    # /* Create following graph and
    # test whether it is 3 colorable
    # (3)---(2)
    # | / |
    # | / |
    # | / |
    # (0)---(1)
    # */
    graph = [
        [0, 1, 1, 1],
        [1, 0, 1, 0],
        [1, 1, 0, 1],
        [1, 0, 1, 0],
    ]
    m = 3 # Number of colors

    # Initialize all color values as 0.
    # This initialization is needed
    # correct functioning of isSafe()
    color = [0 for i in range(4)]

    # Function call
    if (not graphColoring(graph, m, 0, color)):
        print("Solution does not exist")

# This code is contributed by mohit kumar 29
```