Title: Developing agent programs for real world problems
(Graph Coloring)

Ex. No.:02 **Reg. No.:** RA2011003011334

Date: 01/02/2023 **Name: Rishabh Singh Sahil**

Aim:

To color the regions of the given map such that no two adjacent states have the same color. The states are the variables and the colors are the domains.

Program:

```
colors = ['Red', 'Blue', 'Green']
states = ['wa', 'nt', 'sa', 'q', 'nsw', 'v']
neighbors = {}
#adjacent pairing neighbors of different states
neighbors['wa'] = ['nt', 'sa']
neighbors['nt'] = ['wa', 'sa', 'q']
neighbors['sa'] = ['wa', 'nt', 'q', 'nsw', 'v']
neighbors['q'] = ['nt', 'sa', 'snw']
neighbors['nsw'] = ['q', 'sa', 'v']
neighbors['v'] = ['sa', 'nsw']
colors_of_states = {}def promising(state, color): #function to check a promising color - returns a
promising color
for neighbor in neighbors.get(state):
color_of_neighbor = colors_of_states.get(neighbor)
if color_of_neighbor == color: #same color (of neighbor and state) -> rejected
return False
return True
#if not same -> color accepted
def get_color_for_state(state): #promising color is assigned to the state
for color in colors:
if promising(state, color):
return color
def main():
for state in states:
colors_of_states[state] = get_color_for_state(state)
print(colors_of_states)
main()
```

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

- Graph coloring is a problem in graph theory where the goal is to assign colors to the vertices of a graph such that no two adjacent vertices have the same color. One way to manually calculate a proper coloring for a graph is to use a backtracking algorithm. The steps are as follows:

- Assign the first color to the first vertex and color all the vertices that are not adjacent to it with the same color.

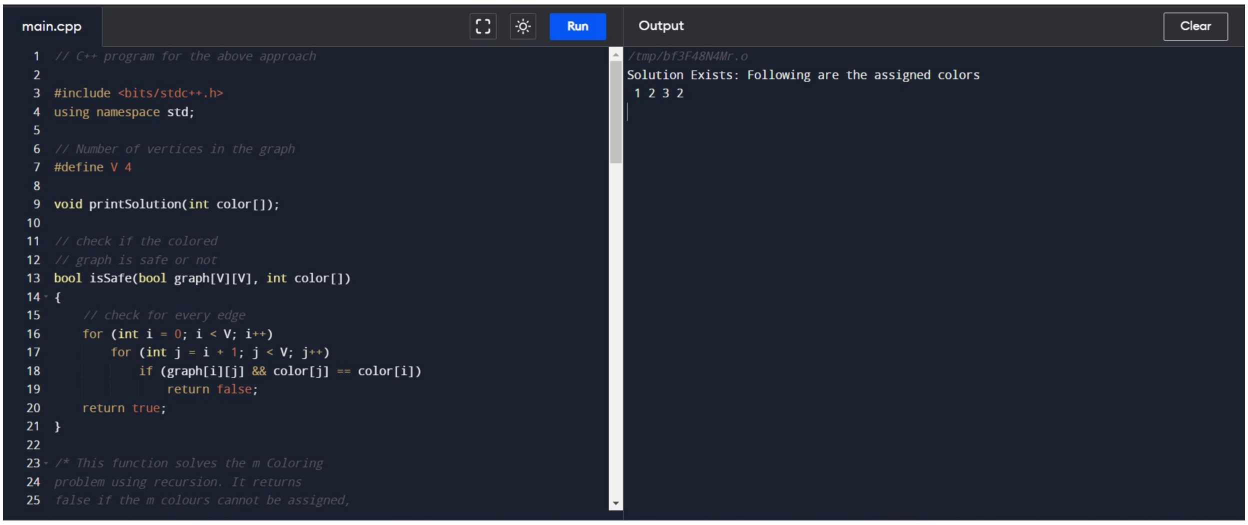
- Repeat the above steps for each uncolored vertex, but use a different color.

- If a vertex cannot be assigned a color, backtrack to the previous vertex and try a different color for it.

- Repeat the above steps until all vertices have been colored or it is determined that the graph cannot be properly colored with the available colors.

- Note: The number of colors needed to properly color a graph is known as its chromatic number.

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



Result:

Implemented Graph coloring problem successfully.

Date:

01-Feb-2023

Experiment - II

Aim:

To color the region of the given map such that no two adjacent state have the same color the State are the variables and the color are the domains

Algorithm:

Start:

colors = ['Red'; 'Blue', 'Green']

States = ['wa', 'nt', 'sa', 'q', 'nsw', 'v']

neighbours = {}

neighbour = ['wa'] = ['nt', 'sa']

neighbour ['nt'] = ['wa', 'sa', 'q']

neighbour ['sa'] = ['wa', 'nt', 'q', 'nsw', 'v']

neighbour ['q'] = ['nt', 'sa', 'snw']

neighbour ['nsw'] = ['q', 'sa', 'v']

neighbour ['v'] = ['sa', nsw']

colour States = {}

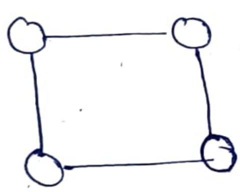
Input: {0, 1, 1, 1}
{1, 0, 1, 0}
{1, 1, 0, 1}
{1, 0, 1, 0}

output:

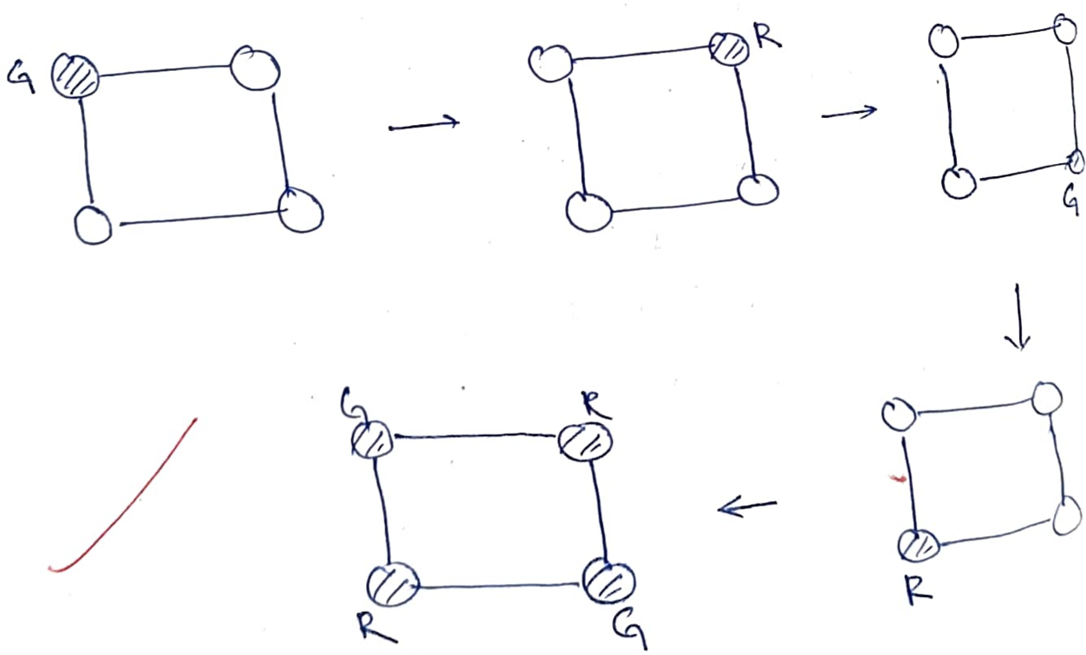
Solution Exists :

Assigned color : 1 2 3 2

Manual Calculation:



→ Given Graph (G)



Time Complexity : $O(V^2 + E)$

```
def promising (state, color):
    for neighbour .get (state):
        color-of-neighbour = color-of-state (neighbour)
        if color-of-neighbour == color:
            return false.

def main():
    for state in states:
        colors-of-state [state] = get-color-for-state

    print (colors-of-states)
main()
```

End:

Result:

Hence, Successfully implemented Map coloring problem.

main.cpp

Run

```cpp
// C++ program for the above approach

#include <bits/stdc++.h>
using namespace std;

// Number of vertices in the graph
#define V 4

void printSolution(int color[]);

// check if the colored
// graph is safe or not
bool isSafe(bool graph[V][V], 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,
```

Output

Clear

```
/tmp/bf3F48N4Mr.o
Solution Exists: Following are the assigned colors
 1 2 3 2
```