SRM INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT OF NETWORKING & COMMUNICATIONS

18CSC305J-ARTIFICIAL INTELLIGENCE

SEMESTER – 6

BATCH-1

REGISTRATION NUMBER	**RA1911003011013**
NAME	**Shivank Arora**

B.Tech- CSE / CC, Third Year (Section: H2)

Faculty In charge: Dr. S. Prabakeran, B.Tech, M.E, PhD
Assistant Professor
School of Computing - Department of Networking and Communications

Year 2021-2022 / Even Semester

I

INDEX

Ex No	DATE	Title	Page No	Marks
1	24/01/2022	Developing agent programs for real-world problems (Graph Coloring Problem)		

Exercise: 2

Date: 24-01-2022

GRAPH COLOURING PROBLEM

Problem Statement: graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. The problem is, given m colors, find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using same color. The other graph coloring problem is ***Edge Coloring*** (No vertex is incident to two edges of same color).

Algorithm :

1.Color first vertex with first color.

2. Do following for remaining V-1 vertices.

a) Consider the currently picked vertex and color it with the lowest numbered color that has not been used on any previously colored vertices adjacent to it.

b) If all previously used colors appear on vertices adjacent to v, assign a new color to it.

Optimization technique : The idea is to assign colors one by one to different vertices, starting from the vertex 0. Before assigning a color, check for safety by considering already assigned colors to the adjacent vertices i.e check if the adjacent vertices have the same color or not. If there is any color assignment that does not violate the conditions, mark the color assignment as part of the solution. If no assignment of color is possible then backtrack and return false.

Algorithm:

1. Create a recursive function that takes the graph, current index, number of vertices, and output color array.
2. If the current index is equal to the number of vertices. Print the color configuration in output array.
3. Assign a color to a vertex (1 to m).
4. 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 next index and number of vertices
5. If any recursive function returns true break the loop and return true.
6. If no recursive function returns true then return false.

Tool : Cloud9 ide and Python 3.9.0

Programming code :

```
class Graph:
      def __init__ (self, edges, N):
        self.adj = [[] for _ in range(N)]
        for (src, dest) in edges:
          self.adj[src].append(dest)
          self.adj[dest].append(src)

def colorGraph(graph):
    result = {}
    for u in range(N):
      assigned = set([result.get(i) for i in graph.adj[u] if i in result])
      color = 1
      for c in assigned:
        if color != c:
            break
        color = color + 1
      result[u] = color
    for v in range(N):
        print("Vertex Color", v, "is",colors[result[v]])
    print("\n")
    for v in range(N):
        print("Edge Color", v, "is",colors[result[v]+3])

if __name__ == '__main__':
    colors= ["", "YELLOW", "RED", "BLUE", "ORANGE", "GREEN", "PINK", "BLACK"
, "BROWN", "WHITE", "PURPLE", "VIOLET","BLUE"]
    edges = [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5),(5,6),(6,7),(7,0)]
    N = 8
    graph = Graph(edges, N)
    colorGraph(graph)
```

Output screen shots:

```
Vertex Color 0 is YELLOW
Vertex Color 1 is RED
Vertex Color 2 is YELLOW
Vertex Color 3 is RED
Vertex Color 4 is YELLOW
Vertex Color 5 is RED
Vertex Color 6 is YELLOW
Vertex Color 7 is RED


Edge Color 0 is ORANGE
Edge Color 1 is GREEN
Edge Color 2 is ORANGE
Edge Color 3 is GREEN
Edge Color 4 is ORANGE
Edge Color 5 is GREEN
Edge Color 6 is ORANGE
Edge Color 7 is GREEN
```

Result : A unique color was successfully assigned to each vertex and edge of the graph.