Name: Praveen Kumar K
Date: 04/01/2022
Subject: AI

EXP2: Implementation of Graph Coloring Problem

Given a graph color its edges such that no two adjacent have the same color using minimum number of colors

Code:

```
class Graph:
def __init__(self, edges, n):
      self.adjList = [[] for _ in range(n)]
      for (src, dest) in edges:
            self.adjList[src].append(dest)
            self.adjList[dest].append(src)
def colorGraph(graph, n):
      result = {}
      for u in range(n):
            assigned = set([result.get(i) for i in graph.adjList[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(f'Color assigned to vertex {v} is {colors[result[v]]}')
if __name__ == '__main__':
colors = ['', 'BLUE', 'GREEN', 'RED', 'YELLOW', 'ORANGE', 'PINK',
'BLACK', 'BROWN', 'WHITE', 'PURPLE', 'VOILET']
edges = [(0, 1), (0, 4), (0, 5), (4, 5), (1, 4), (1, 3), (2, 3), (2, 4)]
n = 6
graph = Graph(edges, n)
colorGraph(graph, n)
```

OUTPUT:

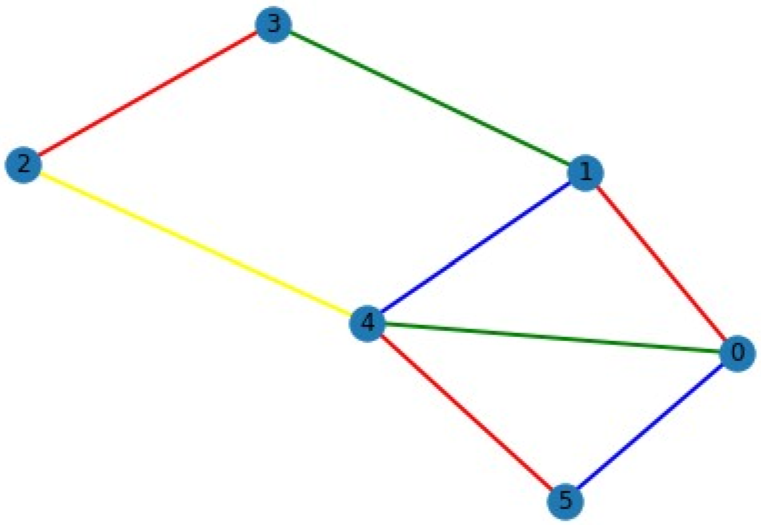
```
Color assigned to vertex 0 is BLUE
Color assigned to vertex 1 is GREEN
Color assigned to vertex 2 is BLUE
Color assigned to vertex 3 is RED
Color assigned to vertex 4 is RED
Color assigned to vertex 5 is GREEN
```

Edge coloring:

```
import matplotlib.pyplot as plt
import networkx as nx
from matplotlib.patches import Polygon
import numpy as np
G = nx.Graph()
colors = {0:"red", 1:"green", 2:"blue", 3:"yellow"}
G.add_nodes_from([0,1,2,3,4,5])
G.add_edges_from([(0, 1), (0, 4), (0, 5), (4, 5), (1, 4), (1, 3), (2, 3),
(2, 4)])
nodes = list(G.nodes)
edges = list(G.edges)
color_lists = []
color_of_edge = []
some_colors = ['red','green','blue','yellow','brown','violet','pink']
for i in range(len(nodes) + 2):
      color_lists.append([])
      color_of_edge.append(-1)
def getSmallestColor(ls1,ls2):
      i = 1
      while(i in ls1 or i in ls2):
            i = i + 1
      return i
i = 0
for ed in edges:
      newColor = getSmallestColor(color_lists[ed[0]],color_lists[ed[1]])
      color_lists[ed[0]].append(newColor)
      color_lists[ed[1]].append(newColor)
      color_of_edge[i]=newColor
      i = i + 1
# Makin graph again
G = nx.Graph()
for i in range(len(edges)):
      G.add_edge(edges[i][0],edges[i]
[1],color=some_colors[color_of_edge[i]-1])

colors = nx.get_edge_attributes(G,'color').values()
nx.draw(G, edge_color=colors, with_labels=True, width=2)
plt.show()
```

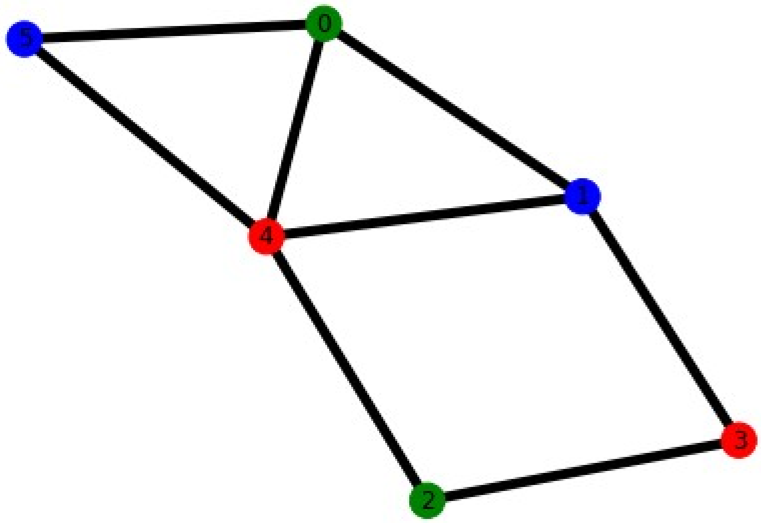
Graph:



Vertex Coloring:

```
import matplotlib.pyplot as plt
import networkx as nx
G = nx.Graph()
colors = {0:"red", 1:"green", 2:"blue", 3:"yellow", 4:"black",
5:"violet", 6:"brown", 7:"pink", 8:"orange",9:"white",10:"purple"}
G.add_nodes_from([0,1,2,3,4,5])
G.add_edges_from([(0, 1), (0, 4), (0, 5), (4, 5), (1, 4), (1, 3), (2, 3),
(2, 4)])
d = nx.coloring.greedy_color(G, strategy = "largest_first")
node_colors = []
for i in sorted (d.keys()):
      node_colors.append(colors[d[i]])
nx.draw(G, node_color = node_colors, with_labels = True, width = 5)
plt.show()
```

Graph:



Result:

Edge and vertex coloring problem which are together known as graph coloring problem solved and visualized in an optimized way using greedy approach.