**Artificial Intelligence Lab**

**LAB 2 – Developing agent programs for real world problems**

**Aditya Kumar**

**RA1911030010100**

**O2 Section**

**Date – 12/1/22**

**Problem Statement:**

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

**Algorithm:**

1. Color first vertex with first color

Loop for remaining (v-1) particles

1. Consider currently picked vertex and color it with lowest nnumbered color that has not been used on any previously colored vertices adjacent to it.
2. If all previously used colors appead on vertices adjacent to v, assign a new color
3. Repeat for all following edges.

**CODE:**

**For edges:**

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([1,2,3,4,5])

G.add\_edges\_from([(1,2), (1,3), (2,4), (3,5), (4,5)])

nodes = list(G.nodes)

edges = list(G.edges)

color\_lists = []

color\_of\_edge = []

some\_colors = ['red','green','blue','yellow']

for i in range(len(nodes) + 1):

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

#iterate over edges

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)):

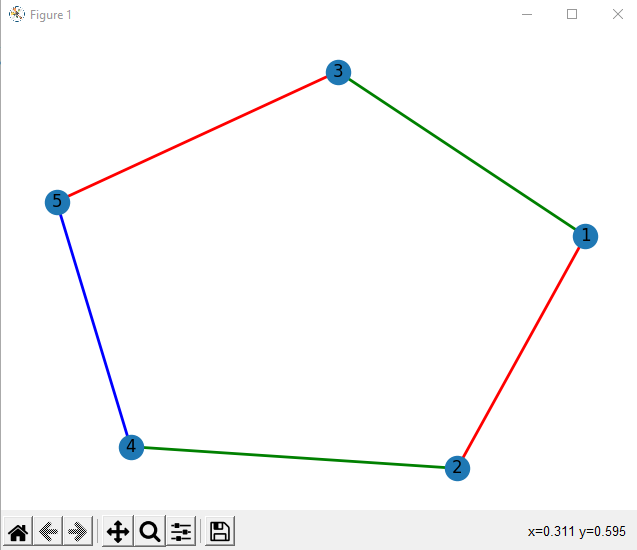
G.add\_edge(edges[i][0],edges[i][1],color=some\_colors[color\_of\_edge[i]-1])

colors = nx.get\_edge\_attributes(G,'color').values()

nx.draw(G, edge\_color=colors, with\_labels=True, width=2)

plt.show()

**Output:**



**For Face:**

import networkx as nx

G = nx.Graph()

colors = {0:"red", 1:"green", 2:"blue", 3:"yellow"}

G.add\_nodes\_from([1,2,3,4,5])

G.add\_edges\_from([(1,2), (1,3), (2,4), (3,4), (4,5)])

nodes = list(G.nodes)

edges = list(G.edges)

some\_colors = ['red','green','blue','yellow']

no\_of\_faces = len(edges)+2-len(nodes)-1

def regionColour(regions):

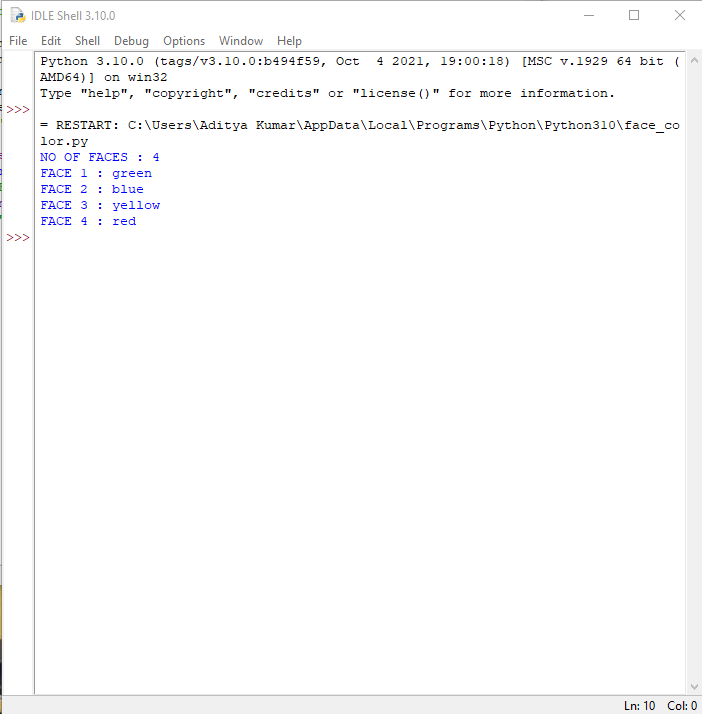
print("NO OF FACES : "+str(regions))

for i in range(1,regions+1):

print(f"FACE {i} : "+some\_colors[i%4])

regionColour(4)

**Output:**



**For Vertex:**

import matplotlib.pyplot as plt

import networkx as nx

G = nx.Graph()

colors = {0:"red", 1:"green", 2:"blue"}

G.add\_nodes\_from([1,2,3,4,5])

G.add\_edges\_from([(1,2), (1,3), (2,4), (3,5), (4,5)])

d = nx.coloring.greedy\_color(G, strategy = "largest\_first")

node\_colors = []

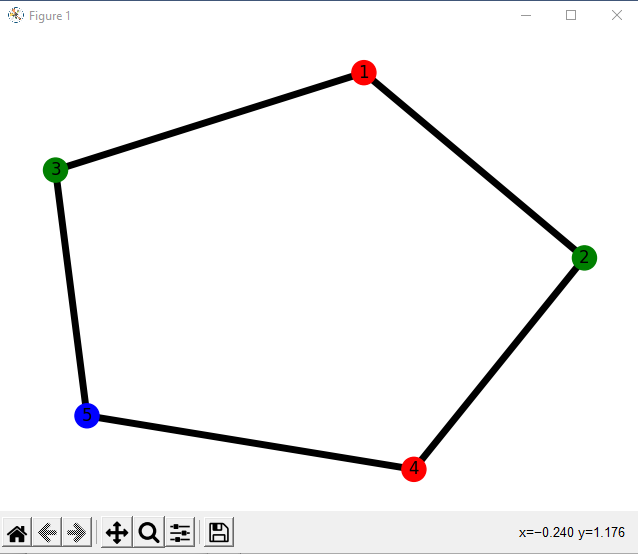
for i in sorted (d.keys()):

node\_colors.append(colors[d[i]])

nx.draw(G, node\_color = node\_colors, with\_labels = True, width = 5)

plt.show()

**Output:**



**Result:**

Hence the graph coloring problem (edge,vertex,face) was solved and was visualized.