



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

U.A.: Computational Geometry

Profesora: PALMA OROZCO ROSAURA

GRUPO: 3CV1

PRÁCTICA 2: 3-coloración

ALUMNO:

VLADIMIR AZPEITIA HERNÁNDEZ

ANÁLISIS

La aplicación que se desarrollará será en consola y lo único que mostrará desde una interfaz de usuario será el resultado, es decir el grafo ya coloreado con los tres colores.

La representación planar se va leer desde un archivo de texto, el cual contendrá la matriz de adyacencia del grafo. Es decir, es una matriz de $n \times n$ (donde n es el número de nodos o vértices del grafo) la cual contendrá 0's y 1's. Si hay un uno, significa que los índices i, j son una arista del grafo que une los vértices con la etiqueta del índice.

Los colores se leerán desde consola, estos pueden ser escritos en inglés o con su valor hexadecimal en rgb.

Algoritmo de coloración:

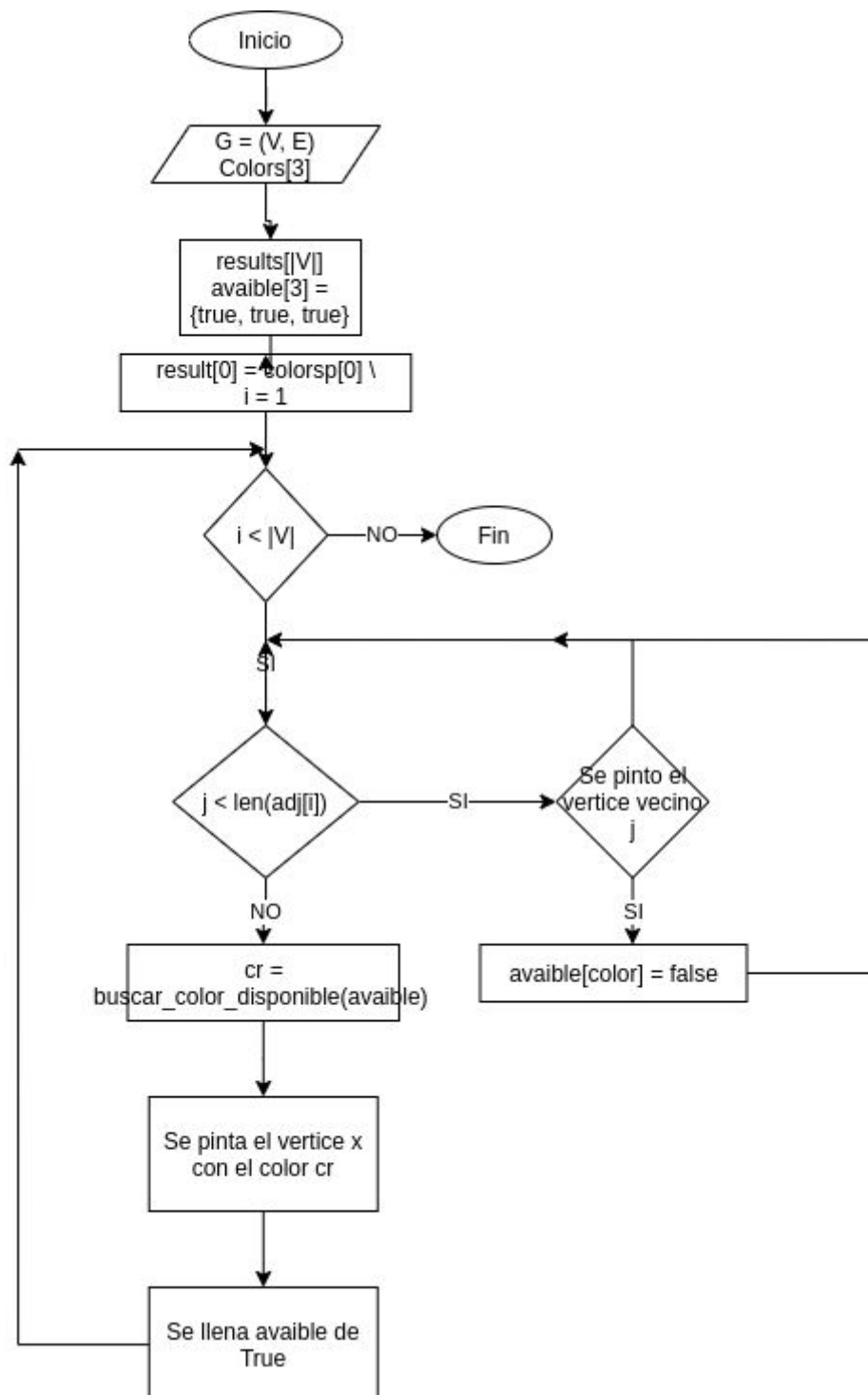
Dado un Grafo $G = (V, E)$ y una lista de colores "colors".

1. Se colorea el primer vértice con el primer color de la lista colorsr.
2. Luego se hace lo siguiente para los vértices $V-1$ restantes.

a) Considere el vértice seleccionado actualmente y coloréalo con el color con el número más bajo que no se haya utilizado en ningún vértice coloreado anteriormente adyacente a él. Si todos los colores utilizados anteriormente aparecen en los vértices adyacentes, asígnele un nuevo color.

ALGORITMO

Diagrama de flujo



IMPLEMENTACIÓN

Se le solicita al usuario indicar el nombre del archivo donde se encuentra la matriz de adyacencia del grafo, y también se le piden los 3 colores con los cuales se pintara el grafo. Después se transforma la matriz de adyacencia en un objeto de tipo grafo.

Posteriormente se llama a la función `colorear_grafo(G, colors)`, la cual recibe como parámetros el grafo que se desea colorear y la lista de colores

```
print("PRACTICA 02: ALGORITMO DE 3-COLORACION")

print("Introduce el nombre del archivo que contiene la matriz de
adyacencia del grafo planar: ")

filename = str(input())

colors = []

print("Ingresa los colores: (máximo 3)")

for i in range(3):

    colors.append(str(input()))

with open(filename, 'r') as f:

    adjacency_matrix = numpy.matrix([[int(num) for num in line.split('
')] for line in f])

G=nx.from_numpy_matrix(adjacency_matrix)

colorear_grafo(G, colors)
```

Implementación del algoritmo para colorear.

```
def colorear_grafo(G, colors):

    for node in G.nodes():

        G.nodes[node]['color'] = 'white' # Vertice NO COLOREADO

    G.nodes[0]['color'] = colors[0]

    print("se pinto el vertice 0 con " + colors[0])

    available = {}

    for c in colors:

        available[c] = True

    for x in range(1, G.number_of_nodes()):

        for y in G[x]:

            if (G.nodes[y].get('color') != 'white'):

                available[G.nodes[y].get('color')] = False

        cr = 0

        for z in range(len(colors)):

            if(available[colors[z]]):

                cr = z

                break

        G.nodes[x]['color'] = colors[cr]

        print("se pinto el vertice " + str(x) + " con " + colors[cr])

    for c in colors:

        available[c] = True

    nodeColors = [G.nodes[node].get('color') for node in G.nodes()]

    nx.draw_planar(G, with_labels = True,

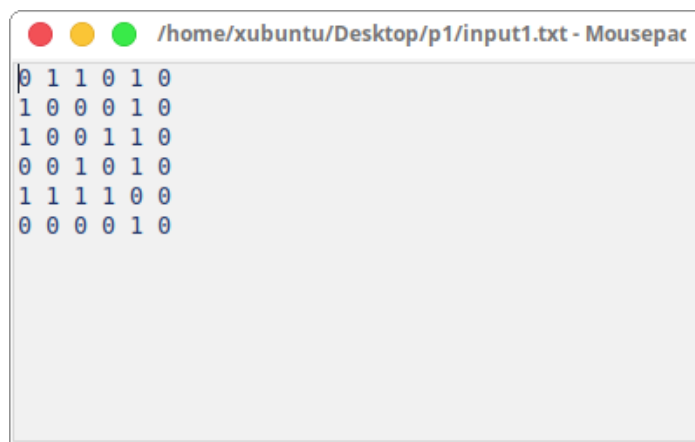
                    font_weight="bold",

                    node_color = nodeColors) # Asigamos el color de los vertices

    plt.show() # Abrir visualizador de matplotlib
```

EJEMPLO DE EJECUCIÓN

Archivo grafo1..txt



```
0 1 1 0 1 0
1 0 0 0 1 0
1 0 0 1 1 0
0 0 1 0 1 0
1 1 1 1 0 0
0 0 0 0 1 0
```

Ejecutando la aplicacion en consola

```
kubuntu@kubuntu-pc:~/Desktop/geometryp2$ /usr/bin/python3 /home/kubuntu/Desktop/geometryp2/p2.py
PRACTICA 02: ALGORITMO DE 3-COLORACION
Introduce el nombre del archivo que contiene la matriz de adyacencia del grafo planar:
grafo1.txt
Ingresa los colores: (máximo 3)
red
pink
yellow
se pinto el vertice 0 con red
se pinto el vertice 1 con pink
se pinto el vertice 2 con pink
se pinto el vertice 3 con red
se pinto el vertice 4 con yellow
se pinto el vertice 5 con red
```

Resultado

