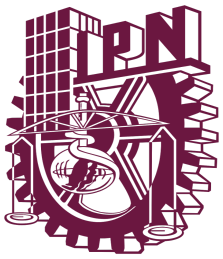


INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A.: Computational Geometry

Profesora: PALMA OROZCO ROSAURA

GRUPO: 3CV1

PRACTICA 1: T4C

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 cuatro 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 nxn (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.

Para colorear el grafo se recorre, con los nodos, y usando el primer color que seleccionó el usuario. Se verifica que este nodo "n" no haya sido coloreado y luego se busca si algunos de los vecinos tiene el mismo color que actualmente está seleccionado; si no hay vecinos con el mismo color se pinta el nodo. Ahora se pasa a usar el siguiente nodo "n"+1 y se vuelve a consultar si no ha sido coloreado o sus vecinos tiene el mismo color y se sigue así hasta terminar de recorrer todos los nodos. Cuando se recorren todos los nodos, ahora se cambia al segundo color que ingresó el usuario y se vuelve a recorrer todos los nodos. Y así hasta terminar con todos los colores o si ya se pintaron todos los nodos.

ALGORITMO

Pseudocódigo

G = (V, E)

caja_de_colores = “color1”, “color2”, “color3”, ”color4”

para cada color **de** caja_de_colores

para cada vértice **de** G

si vértice no ha sido coloreado **entonces**

bandera = 1

para cada nodo vecino **de** vértice

si algun vecino esta pintado de color **entonces**

bandera = 0

fin si

fin para

si bandera es igual 1 **entonces**

se pinta el vertice de color

fin si

fin si

fin para

fin para

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 4 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 los 4 colores.

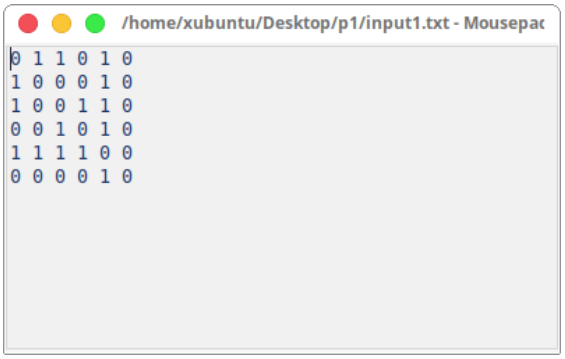
```
print("introduce el nombre del archivo que contiene la
matriz de adyacencia del grafo planar: ")

filename = str(input())

colors = []

print("Ingresa los colores: (máximo 4)")

for i in range(4):

   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 color in colors:
       for node in G.nodes():
            # Si no se ha coloreado
           if (G.nodes[node].get('color') == 'white'):
                 flag = True
                 for x in G[node]:
                       if(G.nodes[x].get('color') == color):
                             flag = False
                  if(flag):
                       G.nodes[node]['color'] = color
   # Dibujar grafo ya coloreado
   nx.draw_planar(G, with_labels = True,
                     font_weight="bold",
                     node_color = nodeColors)
   plt.show() # Abrir visualizador
```

EJEMPLO DE EJECUCIÓN

Archivo input1.txt



Ejecutando la aplicacion en consola

Resultado