Tarea 2 Optimización de Flujo de Redes

Evely Gutiérrez Noda

4 de marzo de 2018

1. Introducción

En el siguiente reporte se abordará el tema estudiado en la clase de Optimización de Flujo de Redes. En dicha clase continuamos el estudio sobre la construcción de grafos, utilizando lenguaje Python [1] para la programación cuando queremos crear un grafo, además utilizamos la herramienta Gnuplot [2], la cual vinculamos con Python para representar el grafo antes programado.

Esta ves en la clase se estudió el trabajo con grafos Simples, Dirigidos y Ponderados, cuyas definiciones se dan a continuación.

Grafo Simple

Es un grafo donde una arista cualquiera es la única que une dos vértices específicos.

Grafo Dirigido

Son grafos en los cuales se ha añadido una orientación a las aristas, representada gráficamente por una flecha.

Grafo Ponderado

Grafos en los cuales se ha añadido un peso a las aristas (número entero generalmente) o un etiquetado a los vértices.

Un grafo simple dirigido, se puede decir que es ponderado al atribuirle peso a cada arista.

2. Descripción del trabajo en clase

En la tarea anterior ya quedo programado un grafo con ciertas carecterísticas, ahora se utilizará este grafo que se creo, para convertirlo en un grafo ademas de simple, dirigido y ponderado.

Para realizar esta actividad comenzamos el trabajo con la estructura de clases en Python, la cual permitira crear metodos generales donde solamente se le cambia el valor a la cantidad de nodos y la probabilidad con que se uniran. De este modo se podra representar distintos tipos de grafos.

Se creo la **clase Grafo**, donde se definieron los parametros n (nodos),x, y (vertice x y vertice y respectivamente), un conjunto$E$, donde mas tarde se almacenara cada par de vertices que seran unidos por una arista, el color de esta arista y el peso correspondiente a la misma. También se creó el parámetro *destino*, donde se almacenará el nombre del archivo en el cual se guardan los nodos y aristas creados, este fragmento de código se muestra a continuación.

```
class Grafo:
```

```
def __init__(self):
    self.n = None # se crean las variables pero aun no se inicializan
    self.x = dict()
    self.y = dict()
    self.E = []  # conjunto vacio
    self.destino = None
```

Luego dentro de la **clase Grafo**, se comenzó a definir funciones para la programación de un grafo, partiendo de las características del grafo programado en la Tarea 1, pero en este caso estará dividido el código por clases, y dentro de sus clases estarán las funciones. Todo esto con el objetivo de organizar el código y de esta forma hacerlo general para la programación de cualquier grafo que se desee. A continuación se expone un fragmento de alguna de las funciones creadas dentro de la **clase Grafo**, y una breve explicación de cada una de ellas.

```
def creaNodos(self, orden): # creando los nodos
    self.n = orden    # se asigna a n, el valor pasado por parametros
    for nodo in range(self.n):
        self.x[nodo] = random()*10 # da valores aleatorios
        self.y[nodo] = random()*10
```

Función CreaNodos: En esta función, como su nombre lo indica, se van a definir el tamaño de **n**(nodos), y luego se va a dar valores aleatorios a las **x** y **y** correspondientes. Para definir una función se utiliza la palabra reservada **def**, y para hacer referencia a una variable global creada dentro de la clase donde se está trabajando, se utiliza la palabra reservada **self**.

Luego se crea la **función Imprimir** para guardar los nodos en un archivo determinado utilizando una estructura parecida a la antes explicada, donde se pasa por parámetro a la función, el archivo de destino donde se guardará, que es una de las variables que se definieron en la estructura de la **clase Grafo**, la cual es **self.destino = None**. El valor **None**, se utiliza cuando la variable se quiere crear vacía.

También se crea la **función Conecta** para conectar dos nodos por una arista, donde esta arista ahora tendrá un peso y un color definidos anteriormente. Luego se almacenará esta unión de dos nodos con una arista con color y peso, en el conjunto E, definido en la estructura inicial de la **clase Grafo**.

Función Conecta

```
def conecta(self, prob):
      for nodo in range(self.n - 1):
          for otro in range(nodo + 1, self.n):
              if random() < prob: # valor de probabilidad para unir de
                   nodos
                  peso = choice(pesos) # se cambian los pesos definidos
                  color = choice(colores) #se cambian los colores
                  if peso > 0:
                      self.E.append((nodo, otro, peso, color)) # se
                          guarda en el conjunto E el par de nodos que
                          van a ser unidos por una arista con un color y
                          peso
                      print(peso)
```

Estas variables **color** y **pesos**, se definen antes de crear la **clase Grafo**, y se le asignan varios valores a cada una, para que a la hora de crear las aristas éstas tengan distintos colores y distintos pesos. Estas variables se crean por medio de las siguientes líneas de código justo antes de crear la **clase Grafo**.

```
colores = ["black", "blue", "pink", "orange", "red"]
pesos = [1,2,3,4, 5, 6, 7, 8]
```

Al ponerle un peso a cada arista esta operación convierte al grafo programado de grafo simple a grafo ponderado.

Luego se creó la **función Grafica**, donde se recorre el archivo con las aristas que ya se crearon anteriormente entre los nodos, y se plotean en el prorama **Gnuplot**. Pero en este caso, en el **set arrow** (que se explicó en la tarea 1) donde se grafica la arista, esta ves se define que la arista tenga dirección, utilizando el parametro **head**.

Al ponerle dirección a las aristas del grafo, se convierte de grafo simple a grafo dirigido, y como ya tiene peso en las aristas también es un grafo ponderado.

Luego de tener la **clase grafo** creada, con las funciones necesarias, entonces se crea otro fichero donde solamente se llamará a cada función, con los valores que se deseen, para construir el grafo(cantidad de nodos n, la probabilidad con que se conectarán los nodos *prob* y los nombres de archivos donde se almacenarán los nodos y aristas), haciendo de este modo un programa general para crear grafos, partiendo de los parametros que se le deseen pasar, en el ejemplo que viene a continuación, se crea un grafo con $n = 20$ y $prob = 0{,}4$.

```
from grafo2 import Grafo
prueba = Grafo() # creo una variable con la estructura de la clase Grafo
prueba.creaNodos(20)
prueba.imprimir("prueba.txt")
prueba.conecta(0.4)
prueba.grafica("prueba.plot")
```

En las siguientes imágenes se ve el resultado de lo antes estudiado. En la figura 1, se ve un grafo dirigido, y en la figura **??** se ve un grafo ponderado y dirigido, en este caso la ponderación se refleja en el grozor de las aristas.

Referencias

[1] Python Software Fundation, `www.python.org/`

[2] Gnuplot, `www.gnuplot.info`

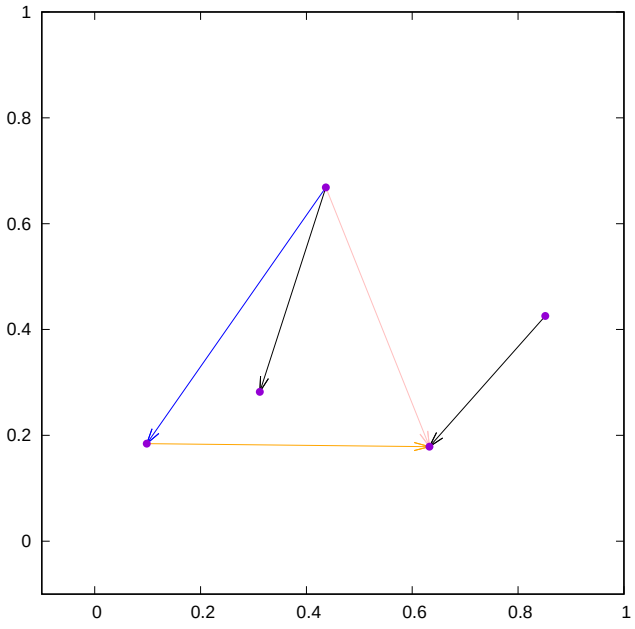

Figura 1: Grafo Dirigido.

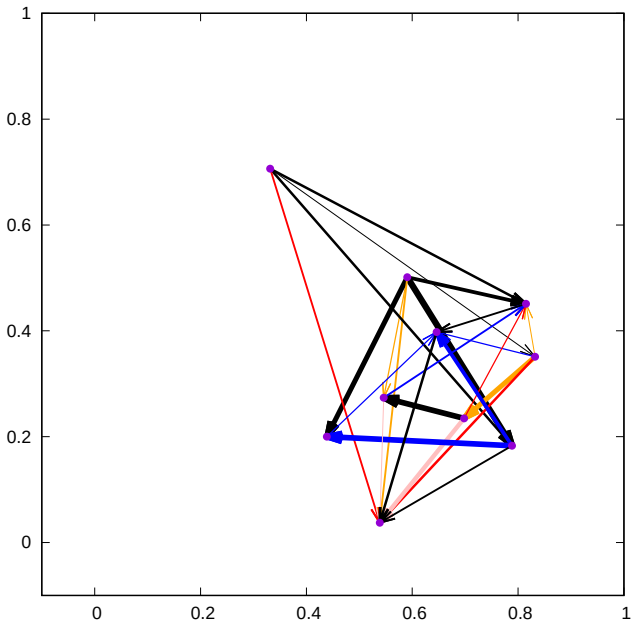

Figura 2: Grafo Ponderado y Dirigido.