Tarea 2 Optimización de Flujo de Redes

Evely Gutiérrez Noda

3 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.

Grafo Simple

Grafo Dirigido

Grafo Ponderado

Es el grafo al cual se le atribuye a cada arista un número específico, llamado ponderación o coste, y se obtiene así un grafo ponderado.

En un grafo ponderado se le llama peso de un camino, a la suma de los pesos de las aristas o arcos que lo forman. Ademas se puede determinar caminos mas cortos o mas largos entre dos vertices obteniendo el camino de menor peso entre los vertices o nodos.

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 creo el parametro **destino**, donde se almacenará el nombre del archivo donde se guardan los nodos y aristas creados, este fragmento de codigo 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 vaco
        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 éste 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 cada función 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 # se le 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 esta trabajando, se utiliza la palabra reservada **self**.

Luego se crea una función para guardar los nodos en un archivo determinado utilizando una estructura parecida a la antes explicada, donde se pasa por parametro 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 definido anteriormente y un color.

Función Conecta

```
def conecta(self, prob):
      for nodo in range(self.n - 1):
          for otro in range(nodo + 1, self.n):
              if random() < prob:
                  peso = choice(pesos)
                  color = choice(colores)
                  if peso > 0:
                      self.E.append((nodo, otro, peso, color))
                      print(peso)
```

Estas variables color y pesos, se definen antes de crear la clase Grafo, y se le asignan varios valores a cada uno, para que a la hora de crear las aristas estas tengan distintos colores y distintos pesos. Estas variables se crean por medio de estas lineas de código.

```
colores = ["black", "blue", "pink", "orange", "red"]
```

```
pesos = [1,2,3,4, 5, 6, 7, 8]
```

Referencias

[1] Python, **http://www.python.org/**, 19 de Febrero de 2018

[2] Gnuplot, **http://www.gnuplot.info/**, 19 de Febrero de 2018