

# Tarea No. 1 de Optimización de Flujo de Redes

Evelly Gutiérrez Noda, Matrícula 1935050

February 15, 2018

## 1 Introducción

En el siguiente reporte se abordará el tema estudiado en la clase de Optimización de Flujo de Redes impartida por la Dr. Elisa. En dicha clase estudiamos sobre la construcción de grafos utilizando lenguaje Python para la programación cuando queremos crear un grafo, además utilizamos la herramienta Gnuplot, la cual vinculamos con Python para representar el grafo antes programado.

## 2 Teoría sobre los Grafos

Primeramente, conocimos un poco sobre los grafos, los cuales son un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas que pueden ser orientados o no. También sabemos que un grafo se representa mediante una serie de puntos (nodos o vertices) conectados por líneas (aristas).(ver figura 1)

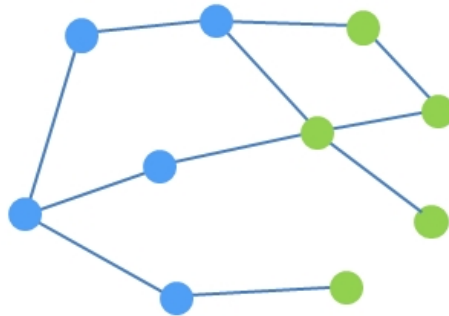


Figure 1: Ejemplo de Grafo

Existen distintos tipos de grafos como por ejemplo:

- **Grafo simple:** Es el grafo donde una arista cualquiera es la única que une dos vértices específicos.
- **Multigrafo:** Es el que acepta más de una arista entre dos vértices. Los grafos simples son una subclase de esta categoría de grafos. También se les llama grafos en general.
- **Pseudografo:** Son aquellos que incluyen algún lazo.

- **Grafo orientado:** Es el grafo dirigido. Son grafos en los cuales se ha añadido una orientación a las aristas, representada gráficamente por una flecha.
- **Grafo etiquetado:** Grafos en los cuales se ha añadido un peso a las aristas (número entero generalmente) o un etiquetado a los vértices.
- **Grafo aleatorio:** Grafo cuyas aristas están asociadas a una probabilidad.
- **Hipergrafo:** Grafos en los cuales las aristas tienen más de dos extremos, es decir, las aristas son incidentes a 3 o más vértices.
- **Grafo infinito:** Son aquellos grafos con un conjunto de vértices y aristas de cardinal infinito.
- **Grafo plano:** Los grafos planos son aquellos cuyos vértices y aristas pueden ser representados sin ninguna intersección entre ellos.
- **Grafo regular:** Un grafo es regular cuando todos sus vértices tienen el mismo grado de valencia.

### 3 Descripción del trabajo en clase

En la clase aprendimos como programar en Phyton un grafo a partir de una cantidad de nodos **n**, tomando por ejemplo una cantidad de nodos **n = 10**, luego utilizando un ciclo **for** creamos un conjunto de nodos con valores aleatorios, los imprimimos en la consola y luego los almacenamos en un archivo de texto llamado **nodos.dat**. Para ello utilizamos las sentencias de código siguientes en la consola de phyton:

---

```
from random import random
x = random()
y = random()
n = 10
for nodo in range(n):
    print(random(),random())
with open("nodos.dat","w") as archivo:
```

▷ Variables para almacenar los nodos

▷ Cantidad de nodos

▷ Ciclo para imprimir los 10 nodos

▷ Trabajo con archivo nodos

---

Luego de tener los nodos creados, construimos las aristas correspondientes de nodo a nodo. Estas aristas las almacenamos en otro fichero llamado **aristas.dat**. Este trabajo se realizó en la consola de phyton, donde utilizamos dos ciclos **for**, uno para recorrer los nodos guardados anteriormente y el otro para agregar un nodo tras del otro siguiendo una condición especificada en un **if**. A continuación esta el codigo utilizado para este trabajo.

---

```
with open("aristas.dat","w") as aristas:
    for (x1,y1) in nodos:
        for (x2,y2) in nodos:
            if random() < 0.1:
                print(x1,y1,x2,y2,file= aristas)
```

▷ Trabajo con archivo aristas

▷ Iteración sobre los primeros pares de coordenadas

▷ Iteración sobre los segundos pares de coordenadas

▷ Condición para graficar

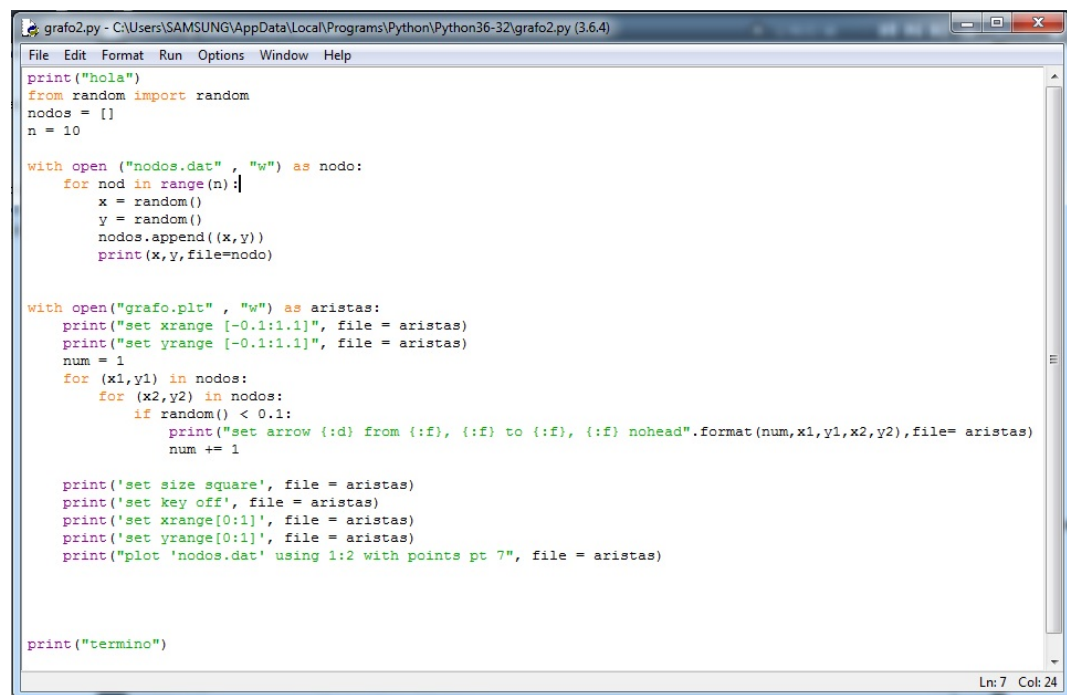
▷ Imprime en consola las coordenadas

---

Luego del trabajo en consola del Phyton, aprendimos a trabajar vinculando el Phyton con Gnuplot, por medio de un editor de Phyton, donde combinamos los códigos de cada programa con una secuencia lógica, para luego abrirlo desde el Gnuplot y de este modo quedaría representado el grafo antes programado (ver figura 2).

Esta representacion del grafo en el editor de Gnuplot se hace utilizando los comandos de Gnuplot **"set arrow ..."** para representar las aristas como "flechas", en este comando tambien definimos cuales serian los nodos a unir y graficar y tambien se puede especificar el estilo con que se va a graficar, en la clase lo hicomos con el estilo que trae por defecto el comando.

Para representar el grafico de puntos desde el archivo **nodos.dat**, por medio de este editor de Gnuplot con comandos de Phyton, usamos **"plot 'nodos.dat' using 1:2"**.



```

grafo2.py - C:\Users\SAMSUNG\AppData\Local\Programs\Python\Python36-32\grafo2.py (3.6.4)
File Edit Format Run Options Window Help
print("hola")
from random import random
nodos = []
n = 10

with open ("nodos.dat" , "w") as nodo:
    for nod in range(n):
        x = random()
        y = random()
        nodos.append((x,y))
        print(x,y,file=nodo)

with open("grafo.plt" , "w") as aristas:
    print("set xrange [-0.1:1.1]", file = aristas)
    print("set yrange [-0.1:1.1]", file = aristas)
    num = 1
    for (x1,y1) in nodos:
        for (x2,y2) in nodos:
            if random() < 0.1:
                print("set arrow {:d} from {:f}, {:f} to {:f}, {:f} nohead".format(num,x1,y1,x2,y2),file= aristas)
                num += 1

    print('set size square', file = aristas)
    print('set key off', file = aristas)
    print('set xrange[0:1]', file = aristas)
    print('set yrange[0:1]', file = aristas)
    print("plot 'nodos.dat' using 1:2 with points pt 7", file = aristas)

print("termino")
Ln: 7 Col: 24

```

Figure 2: Editor de Gnuplot para trabajar con Phyton

Como los valores de los nodos los creamos aleatorios cada vez que se corre el programa en el editor de Gnuplot, se crean nodos diferentes, que arrojaron la creación de diferentes grafos, a partir de varios valores de  $n = 5, 6, 10, 100$  que definen la cantidad de nodos.(ver figura 3)

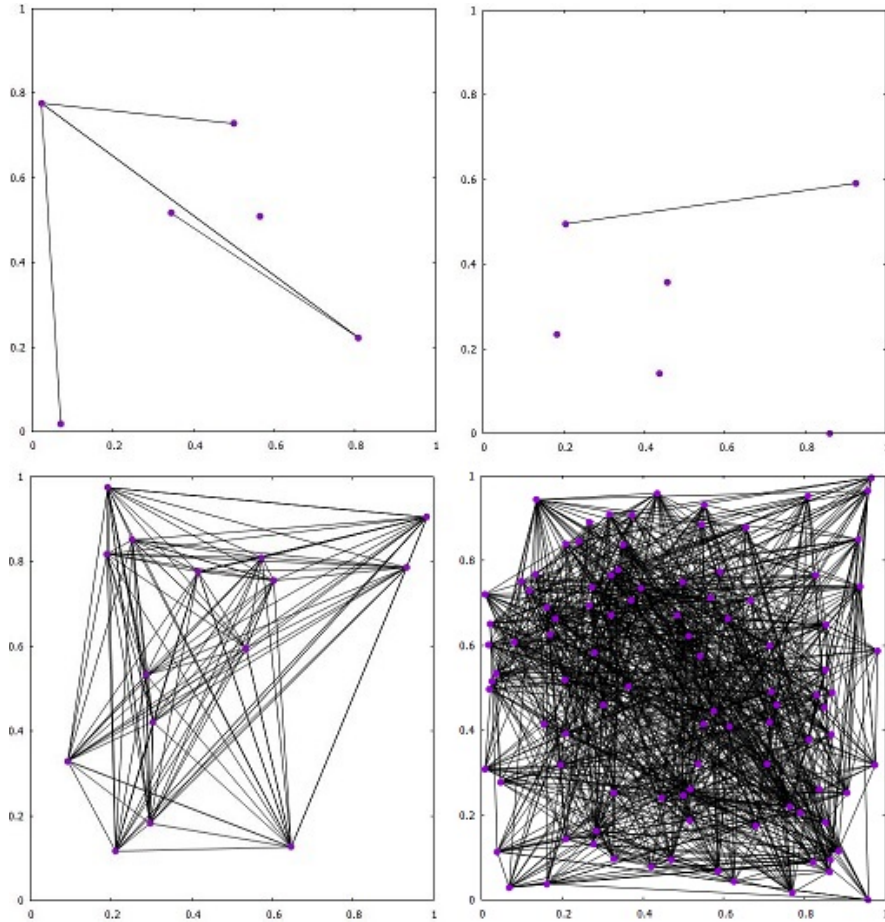


Figure 3: Grafos resultantes de lo aprendido en clase

## 4 Tarea Extraclase

Hasta aquí fue lo aprendido en clase, luego de forma investigativa individual, se hicieron varias modificaciones al grafo programado, cambiando la forma de los nodos, ya sean triángulos, cuadrados y círculos, y cambiando de color y grosor las aristas que unen los nodos. También se le puso nombres a los ejes del plano XY y nombre al grafo a representar. (ver figura 4)

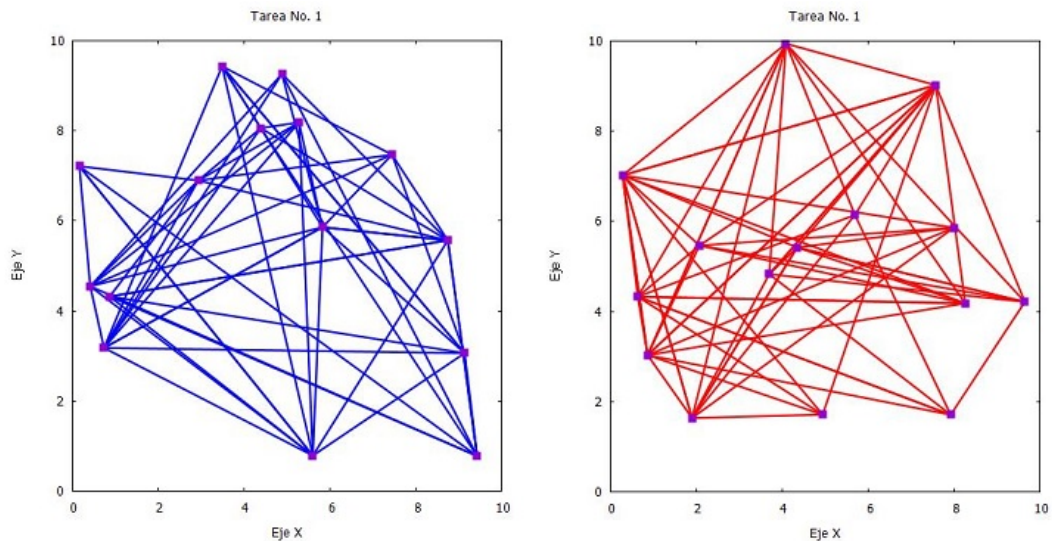


Figure 4: Cambios de grosor y color de aristas

Para colocar el nombre al eje X y Y, se utilizó el comando `set xlabel "eje x"` y `set ylabel "eje y"` respectivamente, para el título del grafo se utilizó el comando `set title (t) "Título"`.

El cambio de color y de grosor de las aristas fue utilizando los comandos `linetype n (lt n)` y `linewidth g (lw g)` respectivamenete.

Con esto se puede ver la aplicación del uso de grafos para poder resolver o representar problemas de flujo en la vida real, ya sea de transporte, servicio de red o cualquier problema en el cual se desee conocer la vía más económica para resolverlo.

Un ejemplo hipotético podría ser para describir o representar las Líneas del metro que existen en Monterrey, utilizando lo aprendido, representar de diferentes colores el flujo de cada línea del metro, y los nodos serían las estaciones correspondientes. Quedarían las líneas rojas para representar la línea del metro 1, y la azul, para la línea del metro 2.(ver figura 5)

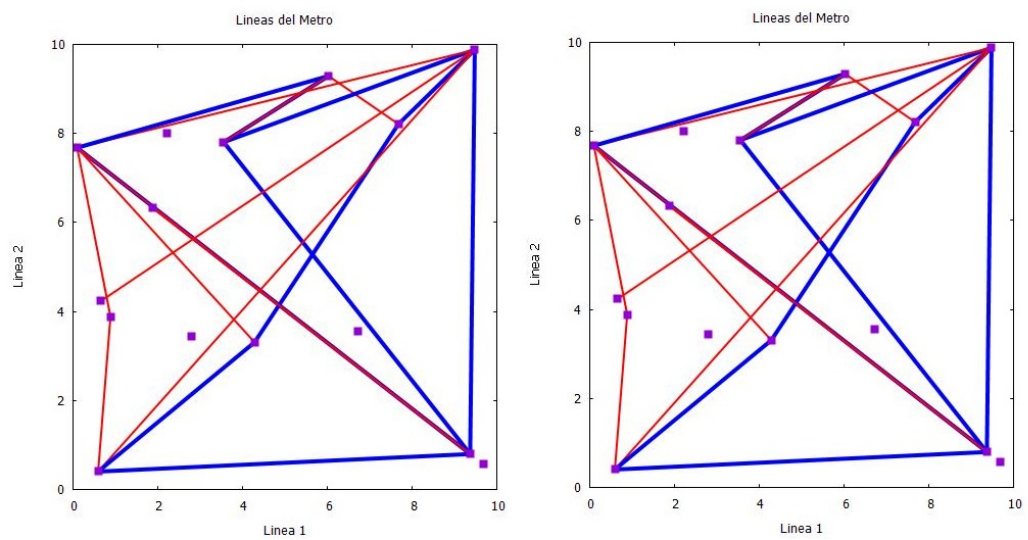


Figure 5: Líneas del Metro