Tarea 2

Samuel Alejandro Sánchez Vázquez

5 de marzo de 2018

Objetivo

El siguiente trabajo tiene como objetivo el poder realizar un código utilizando clases con Python[1] en conjunto con Gnuplot[2] para poder hacer grafos simples(no dirigidos), dirigidos y ponderados.

Tipos de grafo

A continuación se mencionarán los diferentes tipos de grafos con los que se trabajaron:

Grafo no dirigido

Un grafo(grafo no dirigido) G consta de un conjunto V de vértices o nodos y un conjunto E de lados, (ramas o aristas) tales que cada lado e ϵ E está asociado a un par no ordenado de vértices.
Podemos observar en la figura 1 un ejemplo de grafo no dirigido

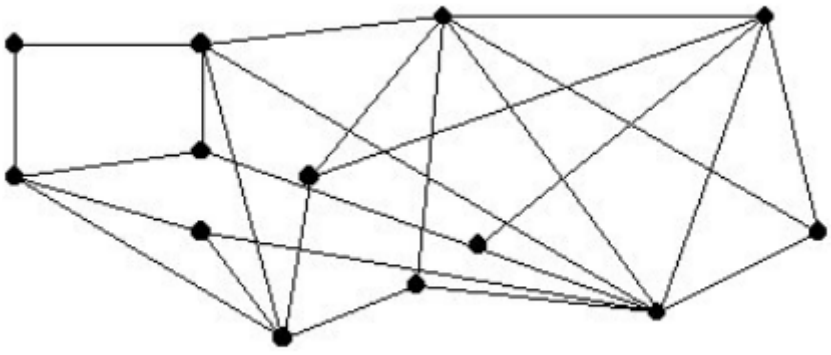

Figura 1

Grafo dirigido

Un grafo dirigido(o digrafo) G consta de un conjunto V de vértices y un conjunto E de lados, tal que e ϵ E está asociado a un par ordenado único de vértices v y w y se escribe e = (v,w).
Podemos observar en la figura 2 un ejemplo de grafo dirigido.

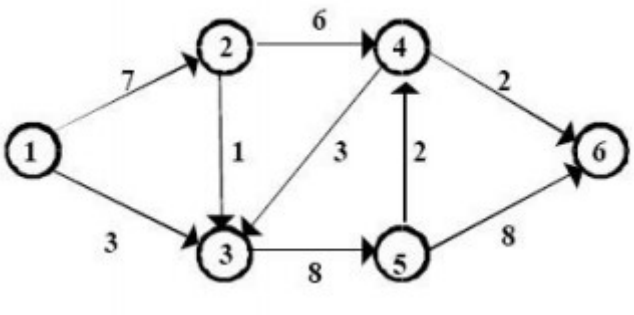

Figura 2

Grafo ponderado

Un grafo ponderado, pesado o con costos es un grafo donde cada arista tiene asociado un valor o etiqueta, para representar el costo, peso, longitud, etc.
Podemos observar en la figura 3 un ejemplo de grafo ponderado.

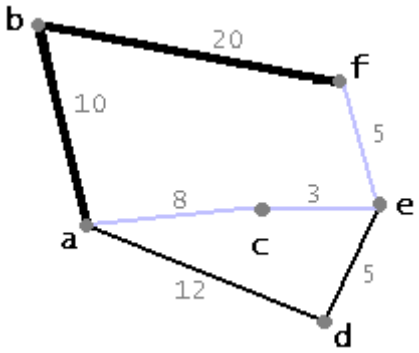

Figura 3

Clases

Una clase es un modelo que define un conjunto de variables(el estado) y apropiados para operar con dichos datos(el comportamiento). Cada objeto creado a partir de la clase se denomina instancia de la clase.
Las clases son un pilar fundamental de la programación orientada a objetos.
Permiten abstraer los datos y sus operaciones asociadas al modo de una caja negra.

Trabajo en clases

Durante la clase anterior(19/02/2018) se nos proporcionó un ejemplo[3] de código de clases en Python[1] con el cual trabajar y así adaptar el código de la tarea 1 en clases.

Aplicación de clases en Python

A continuación se irá explicando cada segmento del código en clases y su uso:

```
class Grafo:
    def __init__(self):
        self.n = None
        self.x = dict()
        self.y = dict()
        self.E = []
        self.destino = None
```

En este segmento de la clase Grafo, se inicializan las variables que se utilizarón más adelante, es decir se crearon las variables sin valores aún.
La variable $self.n$ se refiere a la cantidad de nodos que tendrá el grafo.
Las variables $self.x$ y $self.y$ se refieren a la dirección que tendrá el nodo.
La variable $self.E$ es el conjunto vacío donde se irán almacenando las variables que se ocuparán en cada segmento más adelante.
La variable $self.destino$ es la dirección en conjunto de x, y que tendrán de nodo a nodo el grafo.

```
def generar(self, orden):
        self.n = orden
        for nodo in range(self.n):
            self.x[nodo] = random()
            self.y[nodo] = random()
```

En este segmento se asigna la cantidad de nodos que tendrá el grafo, así como los valores de x, y de cada nodo.

```
def imprimir(self, direccion):
        self.destino = direccion
        with open(self.destino, "w") as archivo:
            for nodo in range(self.n):
                print(self.x[nodo], self.y[nodo], file=archivo)
        print(self.destino)
```

En este segmento se guardan en un archivo los valores de x, y de cada nodo asignado.

```
def conectar(self, probabilidad):
        for nodo in range(self.n - 1):
            for nodo2 in range(nodo + 1, self.n):
                if random() < probabilidad:
                    self.E.append((nodo, nodo2))
            print(len(self.E))
```

En este segmento se da la probabilidad de posibles conexiones entre el nodo 1 y el nodo 2, así sucesivamente, hasta cubrir todos los nodos asignados.

```
def graficar (self, plot, tipo):
        assert self.destino is not None
        with open(plot, "w") as aristas:
            cabecera(aristas)
            num = 1
            lwindex = 0
            for(v, w) in self.E:
                x1 = self.x[v]
                x2 = self.x[w]
                y1 = self.y[v]
                y2 = self.y[w]
                weight = int(random()*5 + 1)
                color = int(random()*10)
                if tipo == 0:
                    print("set arrow {:d} from {:f}, {:f} to {:f}, {:f}
                    nohead lw 1 lc rgb '{:s}'".format(num,x1,y1,x2,y2,
                    colores[color]), file = aristas)

                if tipo == 1:
                    print("set arrow {:d} from {:f}, {:f} to {:f}, {:f}
                    head filled lw 1 lc rgb '{:s}'".format(num,x1,y1,x2,y2,
                    colores[color]), file = aristas)

                if tipo == 2:
                    print("set arrow {:d} from {:f}, {:f} to {:f}, {:f}
                    nohead  lw {:d} lc rgb '{:s}'".format(num,x1,y1,x2,y2,
                    weight,colores[color]), file = aristas)

                if tipo == 3:
                    print("set arrow {:d} from {:f}, {:f} to {:f}, {:f}
                    head filled lw {:d} lc rgb '{:s}'".format(num,x1,y1,x2,
                    y2,weight,colores[color]), file = aristas)
                lwindex += 1
                num += 1
            inf(self.destino, aristas)
```

En este segmento se dan las instrucciones para que el archivo pueda ser leído en Gnuplot[2] para realizar el grafo correspondiente.
Se separo en 4 secciones para la realización de los grafos, para 4 tipos de grafos.
Si *tipo* = 0 es para un grafo simple o no dirigido, en el cual la principal diferencia es que no tiene dirección (nohead) y el grosor de las aristas (lw) se mantiene constante.
Si *tipo* = 1 es para un grafo dirigido, es decir tiene dirección (head), el grosor de las aristas (lw) se mantiene constante.
Si *tipo* = 2 es para un grafo ponderado es decir, no tiene dirección pero el grosor de las aristas, varía.
Si *tipo* = 3 es para hacer un grafo dirigido (head) y ponderado (lw variable para cada nodo).

```
from Grafo2 import Grafo
Grafo = Grafo()
Grafo.generar(6)
Grafo.imprimir("test.txt")
Grafo.conectar(0.5)
Grafo.graficar("test.plot",1)
```

En esta parte del código se resumen todos los segmentos de clase antes mencionados, solo para agregar los valores y características que se quieran para el grafo buscado.

Resultados y Grafos

A continuación se muestran los grafos obtenidos de gnuplot al correr el código:

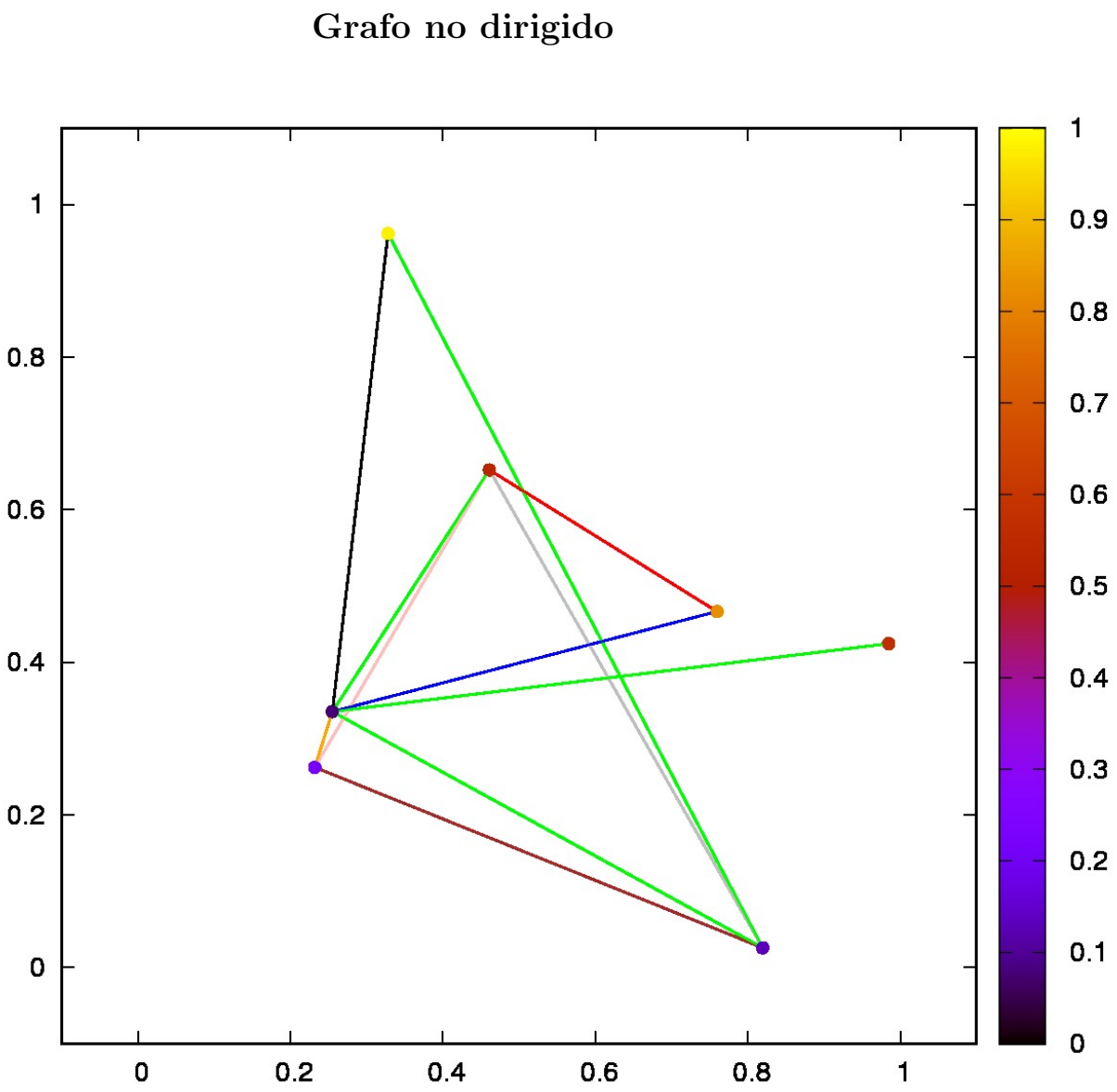

Figura 4

Grafo dirigido

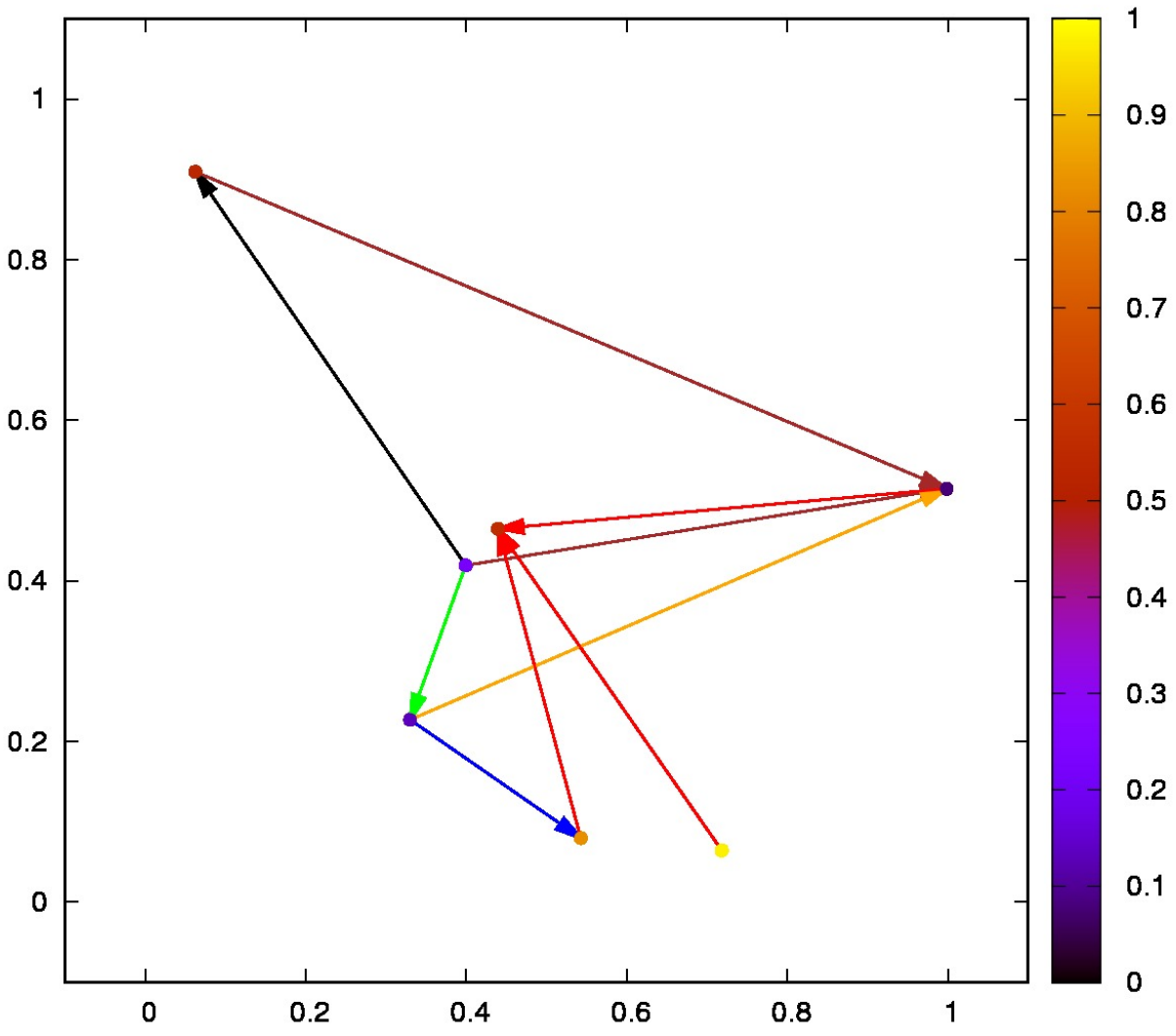

Figura 5

Grafo ponderado

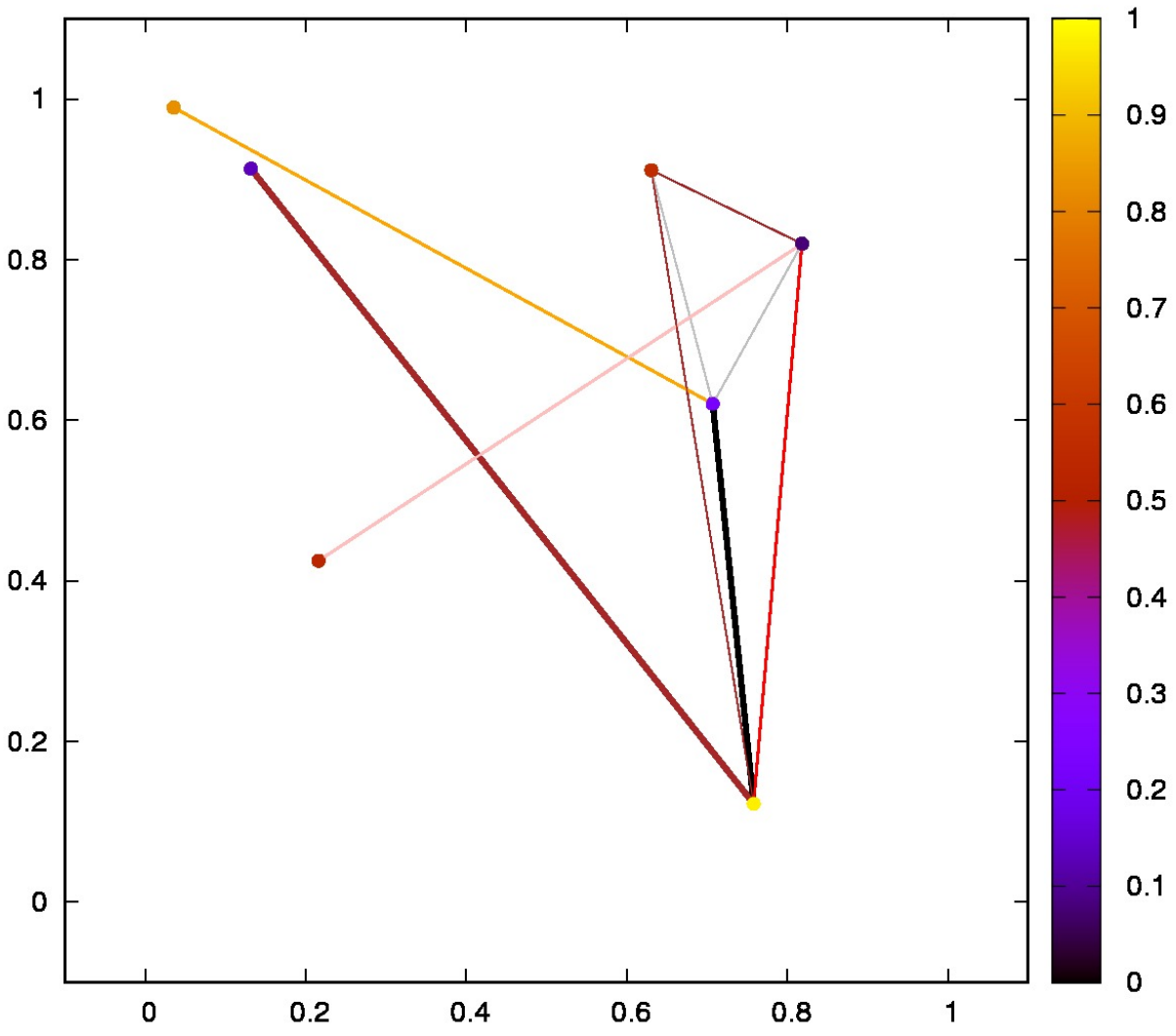

Figura 6

Grafo combinado

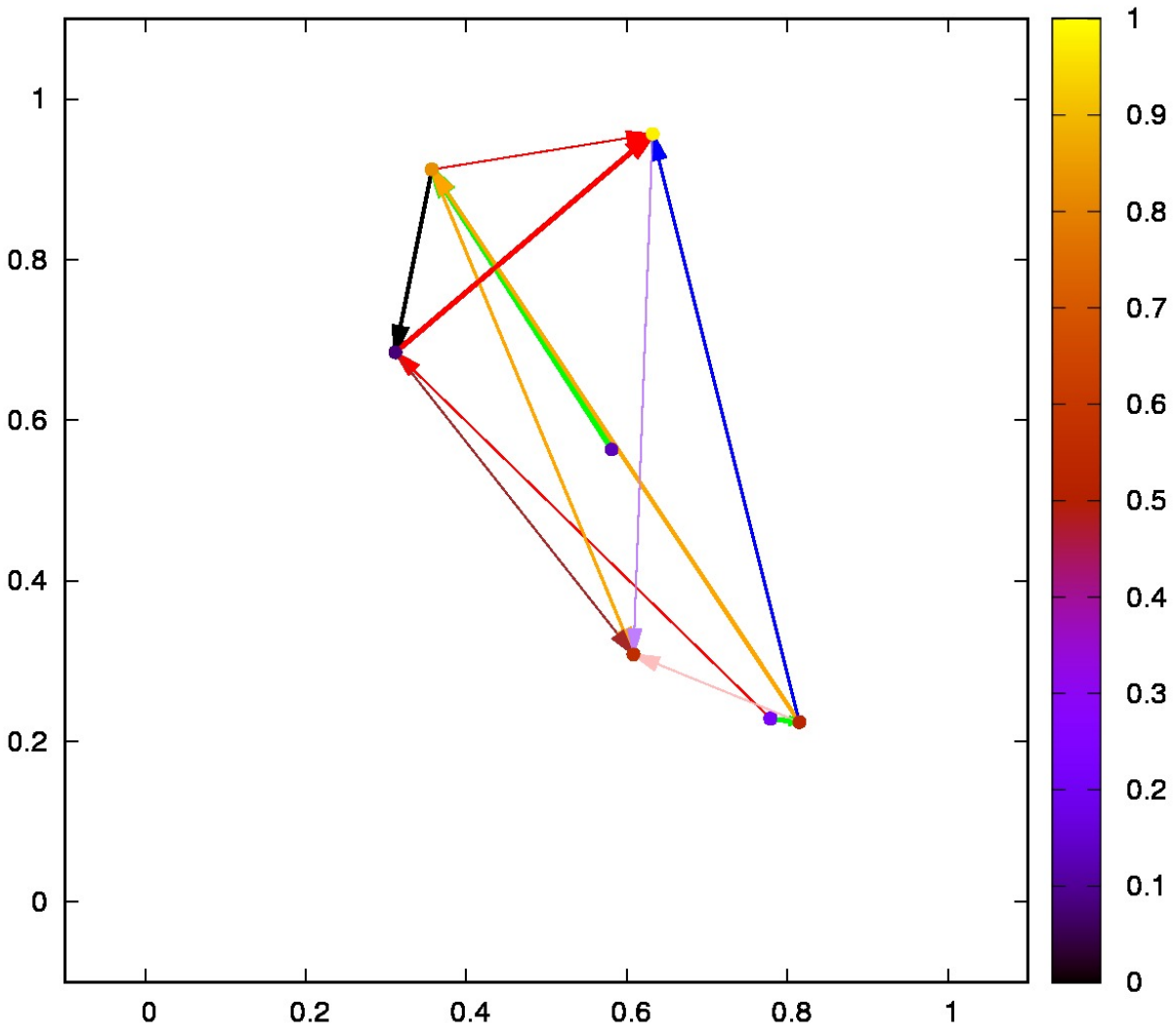

Figura 7

Conclusiones

El trabajar con grafos tiene mucho campo de aplicación así como análisis de datos, tener como herramienta para la generación de archivos como el Gnuplot, es muy útil. El poder conocer y manejar estas herramientas, nos permite un mejor visualización de la información y una mejor toma de decisiones.

Bibliografía

[1] https://www.python.org/about/
[2] http://www.gnuplot.info/
[3] Grafos en Python,
https://elisa.dyndns-web.com/teaching/mat/discretas/md.html
http://163.10.22.82/OAS/estructuras_de_grafos/grafo_no_dirigido.html
http://163.10.22.82/OAS/estructuras_de_grafos/grafo_dirigido.html$
http://163.10.22.82/OAS/estructuras_de_grafos/grafo_ponderado.html
http://docs.python.org.ar/tutorial/2/classes.html