Tarea 3 Optimización de Flujo de Redes

Evely Gutiérrez Noda

4 de abril 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 vez en la clase se realizaron experimentos con el algoritmo de **Floyd-Warshall**.

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 creó,para trabajar con el algoritmo **Floyd-Warshall**.

Se creó la **clase Grafo**, donde se definieron los parámetros n (nodos),x, y (vértice x y vértice y respectivamente), un conjunto E, donde mas tarde se almacenará cada par de vértices que serán 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 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, y además se agregaron los algoritmos de **FloydWarshall** para caminos cortos y **FordFulkerson**. Se realizaron algunos cambios en el código utilizado en la Tarea 2 [[3]] para poder utilizar estos algoritmos, ya que anteriormente no se trabajaba con un arreglo donde se guardaran los nodos vecinos, y este arreglo fue necesario crearlo para poder utilizar estos algoritmos.

El algoritmo **FloydWarshall** es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo en-

cuentra el camino entre todos los pares de vértices en una única ejecución. Esto es semejante a construir una tabla con todas las distancias mínimas entre pares de ciudades de un mapa, indicando además la ruta a seguir para ir de la primera ciudad a la segunda. Este es uno de los problemas más interesantes que se pueden resolver con algoritmos de grafos.

Este algoritmo trabaja comparando todos los posibles caminos a través del grafo entre cada par de vértices, esto lo hace mejorando paulatinamente una estimación del camino más corto entre dos vértices, hasta que se sabe que la estimación es óptima. El algoritmo usa la metodología de Programación Dinámica para resolver el problema. Éste puede resolver el problema con pesos negativos y tiempos de ejecución iguales a $O(V^3 3)$; sin embargo, para ciclos de peso negativo el algoritmo tiene problemas

A continuación se muestra como quedo modificado este algoritmo para poderlo utilizar en el grafo que se crea en el codigo de la Tarea 2 [[3]]

```
def FloydWarshall(self):
      d = {}
      for nodo in range(self.n - 1):
          d[(nodo, nodo)] = 0 # distancia reflexiva es cero
          for (vecino, peso) in self.vecinos[nodo]: # para vecinos, la
               distancia es el peso
              d[(nodo, vecino)] = peso
      for intermedio in self.vecinos:
          for desde in self.vecinos:
              for hasta in self.vecinos:
                  di = None
                  if (desde, intermedio) in d:
                      di = d[(desde, intermedio)]
                  ih = None
                  if (intermedio, hasta) in d:
                      ih = d[(intermedio, hasta)]
                  if di is not None and ih is not None:
                      c = di + ih # largo del camino via "i"
                      if (desde, hasta) not in d or c < d[(desde,
                           hasta)]:
                          d[(desde, hasta)] = c # mejora al camino actual
      return d
```

Referencias

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

[2] Gnuplot, www.gnuplot.info

[3] Tarea2, https://github.com/EvelyGutierrez/Optimizacion-de-flujo-de-redes/blob/master/Tarea2/VersionFinal/Tarea%202.pdf

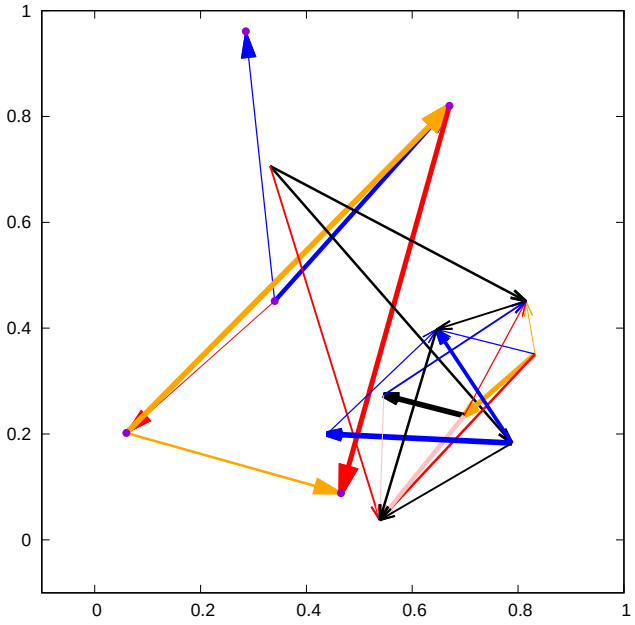

Figura 1: Grafo Dirigido.