Hoja de trabajo No. 10

Realizar: Programa de implementación de grafos para algoritmo de Floyd.
Realizarse: parejas**.**

Objetivos:
a. Implementación de un grafo en Java.
b. Implementación del algoritmo de Floyd (camino más corto entre cualquier par de vértices del grafo) en Java.
c. Usar los resultados del algoritmo de Floyd para calcular el centro del grafo, en Java.
d. Opcional: usar el módulo de Python NetworkX para el algoritmo de Floyd.

Programas a realizar:
1. Implementar grafos en Java. Basta con implementar grafos con dirección.
2. Se recomienda (pero no es obligatorio) usar la implementación de un grafo por medio de una Matriz de Adyacencia. (Solo para grafos con dirección).
3. Implementar el algoritmo de Floyd para encontrar la ruta entre cualquier par de nodos del grafo. El resultado del camino más corto debe indicar el peso de la ruta y los nodos por los que pasa la ruta más corta entre cualquier par de nodos.
4. El programa permite leer el grafo desde un archivo y posteriormente permitir agregar o eliminar arcos desde la pantalla.
5. Su programa también calcula el centro del grafo.

Problema[1]**:**
A logistics company that ships goods to different locations via a network of warehouses needs to optimize its supply chain to reduce transportation costs and improve delivery times. The company currently uses a simple routing system based on the

[1] Elaborado con la ayuda de mi amigo chatGPT-3. Refinado para este curso.

shortest distance between two points[2]. However, this approach does not take into account traffic congestion, weather conditions, or other factors that may impact the delivery time and cost.

¡Aquí es donde entra usted! No fue en valde estudiar tantas cosas en Algoritmos y Estructura de Datos, principalmente el algoritmo de Floyd para calcular la ruta más corta entre cualquier par de nodos. Using Floyd's algorithm, the company was able to create a more optimized routing system that takes into account various factors such as traffic congestion, weather conditions, and delivery schedules. The algorithm allows the company to quickly determine the shortest path between any two points in the network, factoring in all relevant variables.

In order to use Floyd's algorithm to optimize the supply chain taking into account distance and weather conditions, we can use a weighted graph where each node represents a location in the supply chain (e.g. warehouse, distribution center, etc.) and the edges between nodes represent the distance between them, taking into account the impact of weather conditions on travel time.

For example, we could use historical data on travel times between locations during different weather conditions (such as rain, snow, or wind) to estimate the impact of weather on travel time. We could then assign a weight to each edge based on the distance between the two locations and the estimated impact of weather on travel time.

It is important to note that the weights assigned to each edge would need to be updated regularly to reflect changes in weather conditions and traffic patterns. The company could use real-time data on traffic and weather conditions to adjust the weights in the adjacency matrix on a regular basis, ensuring that the routing system is always optimized for the current conditions.

Para lograr esto, su programa debe leer un archivo txt que representa el grafo, llamado **logistica.txt** Cada línea tiene, el nombre de la ciudad1, nombre de la ciudad2 y los tiempos en horas para llegar de la ciudad1 a la ciudad2. Los tiempos son: a) con clima normal, b) con lluvia, c) con nieve, d) con tormenta.

Un ejemplo del contenido del archivo **logistica.txt** (nota: el encabezado no esta en el archivo)

Ciudad1 Ciudad2 tiempoNormal tiempoLluvia tiempoNieve tiempoTormenta
BuenosAires SaoPaulo 10 15 20 50
BuenosAires Lima 15 20 30 70
Lima Quito 10 12 15 20

Notar que no se pone espacio en blanco en el nombre de las ciudades, por ejemplo en BuenosAires, para facilitar su programa.

El archivo logistica.txt no es proporcionado con la tarea. Usted puede hacerlo con los datos que desee. Los auxiliares usarán el que ellos han elaborado para calificar su programa.

Luego de leer el archivo, se construye el grafo y se aplica el algoritmo de Floyd para calcular la distancia más corta, de acuerdo con el tiempo con clima normal entre cualquier par de ciudades y cuál es la ciudad que queda en el centro del grafo. Se debe mostrar la matriz de adyacencia que representa el grafo (o la representación que usted haya implementado).

Las opciones de su programa, en una repetición, son:

1. Preguntar el nombre de la ciudad origen y ciudad destino. El programa mostrará el valor de la ruta más corta y los nombres de las ciudades intermedias por donde pasa esa ruta.
2. Indicar el nombre de la ciudad que queda en el centro del grafo. (Ver documento adjunto, con el algoritmo para calcular el centro del grafo).
3. Modificar el grafo indicando algunos de los siguientes datos:
    a. hay interrupción de tráfico entre un par de ciudades
    b. se establece una conexión entre ciudad1 y ciudad2 (recordar ingresar todos los tiempos). Por defecto se usa el tiempo con clima normal en esta nueva conexión.

[2] Se recuerda de nuestro algoritmo Dijkstra?

c. Se indica el clima (normal, lluvia, nieve o tormenta) entre un par de ciudades.

Al fin de estas modificaciones se recalculan las rutas más cortas y el nuevo centro del grafo.

4. Finalizar el programa.

Tareas:

a. Implementar grafo en Java.
b. Construir el programa que utilice la implementación del grafo y realice el algoritmo de Floyd. Con base a los resultados del algoritmo de Floyd se calcula el centro del grafo.
c. Usar JUnit para probar por lo menos los métodos para agregar nodos, agregar y eliminar arcos y el algoritmo de Floyd.
d. Usar un sistema de control de versiones, como Git para registrar el trabajo realizado.
e. Hacer el diagrama UML de clases que se emplean en su programa.
f. Colocar el programa Java. Incluir el diagrama UML de clases, las pruebas unitarias y el enlace al control de versiones del programa.

Opcional:

Puede implementar el mismo programa en Python, usando el módulo NetworkX. Son puntos extras que puede aplicar en la hoja de trabajo en que haya sacado menos nota (o no haya entregado). Los puntos que se le pondrán son un máximo de 30 puntos.

Calificación: su programa debe funcionar para ser calificado.

Aspecto	Puntos
Implementación del Grafo en Java	15
Implementación del algoritmo de Floyd en Java	20
Implementación del cálculo del centro del grafo.	15
Programa principal que usa el grafo y el algoritmo de Floyd	20
Diagrama UML de clases	10
Control de versiones	5
Pruebas unitarias de los métodos del grafo con JUnit	15
TOTAL:	100
Opcional:	
Programa implementado en Python con módulo NetworkX	30

Referencias:

1. El algoritmo de Floyd lo puede encontrar en la plataforma del curso.
2. Adjunto a esta hoja se encuentra el algoritmo para calcular el centro de un grafo.