Objetivo

El objetivo de este proyecto es poner en práctica los conceptos aprendidos en clase acerca de grafos como estructura de datos.

Las Fuentes de Datos

A continuación, se presenta una descripción de las fuentes de datos que se utilizarán en el proyecto.

Intersecciones de la malla vial de Bogotá y las zonas de Uber

Se representan las intersecciones de la malla vial como los vértices del grafo. Estas intersecciones están representadas en el archivo bogota_vertices.txt con id, longitud, latitud y MOVEMENT_ID (zona Uber):

```
id;long;lat;MOVEMENT_ID
0;-74.08921298299998;4.582989396000016;275
1;-74.08952746199999;4.582560966000016;275
2;-74.09389220199999;4.576679366000008;684
…
```

Grafo de Calles

El grafo de la **malla vial de la ciudad de Bogotá** se debe construir a partir de la carga de las vías relacionadas en el archivo *bogota_arcos.txt separado por espacios*. Estas vías se representan como un conjunto de adyacencias donde el primer campo especifica el id del vértice origen, y los demás campos son los ids de los vértices destino. En el siguiente ejemplo, la primera fila especifica que el vértice 0 está conectado con los vértices 1 y 733.

```
0 1 733
1 49
…
```

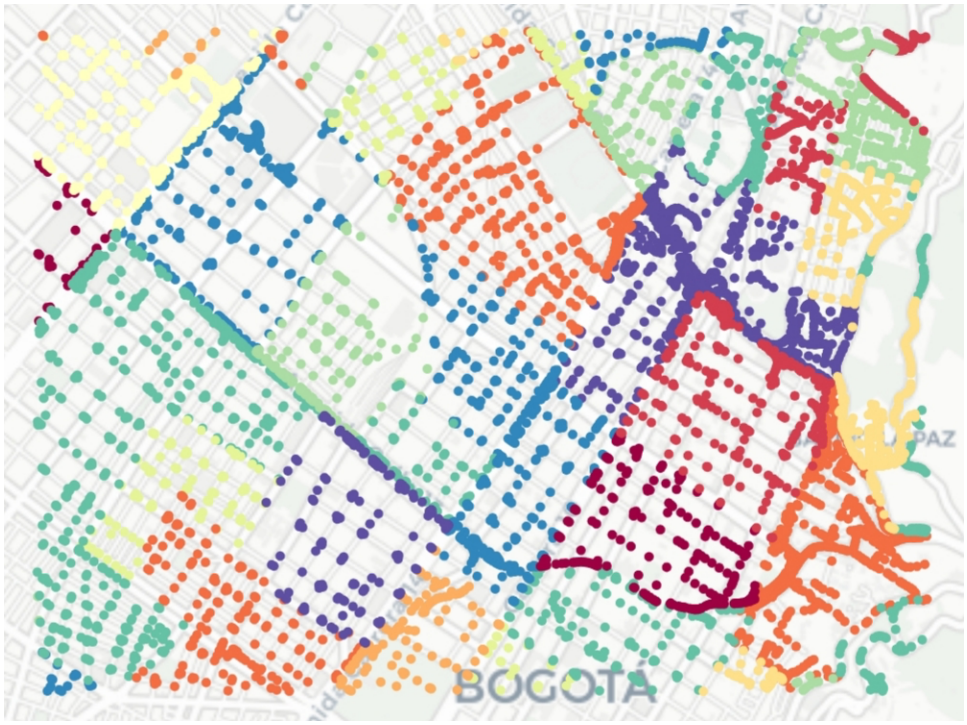

Fig 1. Malla vial del centro de Bogota con vértices coloreados por Zona Uber

Promedio de tiempos de viajes entre zonas Uber

Se van a usar los datos en formato CSV de los tiempos de viaje en Uber por día de la semana en Bogotá solo para el primer trimestre 2018. Los datos se pueden descargar en https://movement.uber.com/?lang=es-CO ó desde Sicua:

- bogota-cadastral-2018-1-WeeklyAggregate.csv

Este archivo tiene el siguiente formato:

```
sourceid,dstid,dow,mean_travel_time,standard_deviation_travel_time
,geometric_mean_travel_time,geometric_standard_deviation_travel_ti
me
706,157,7,771.58,346.93,710.63,1.48
702,197,7,1619.37,434.59,1565.36,1.29
707,147,7,1549.21,536.21,1478.0,1.34
1100,826,3,1481.63,524.21,1409.49,1.36
```

Requerimientos

0. Cargar el Grafo No Dirigido (grafo más grande) de la malla vial de la ciudad completa de Bogotá creado en el taller 7. Informar el total de vértices y el total de arcos que definen el grafo cargado (cada arco debe contarse una única vez). Solo es permitido leer una vez la información de los archivos.

1. Al grafo creado, se debe agregar la información de costo. El grafo va tener 3 tipos de costo en sus arcos:
    a. El primer tipo de costo de un arco es la distancia Harversine (en kilómetros) entre las localizaciones geográficas de los vértices que conecta. Esta distancia ya fue calculada en el taller 7.
    b. El segundo tipo de costo de un arco es el tiempo de viaje entre sus vértices. Este se calcula tomando los promedios de tiempos de viaje (segundos) entre las zonas Uber (MOVEMENT_ID) de sus vértices. Para tal fin, el costo en tiempo entre los vértices de un arco se define de la siguiente forma:
        i. Si los vértices del arco pertenecen a la misma zona Uber, el peso es igual al tiempo promedio de los viajes Uber reportados en el trimestre donde la zona origen y destino es la misma. Si no existen tiempos promedios de viajes Uber para esa misma zona origen y destino, se asignará un costo predefinido de 10 segundos.
        ii. Si los vértices del arco están en diferentes zonas Uber, el peso es igual al tiempo promedio de los viajes Uber reportados en el trimestre desde la zona del vértice origen hasta la zona del vértice destino. Si no existen tiempos promedios de viajes Uber entre la zona origen y la zona destino, se asignará un costo predefinido de 100 segundos.
    c. El tercer tipo de costo es la velocidad del arco, calculada como su distancia dividida por su tiempo.

Requerimientos - Parte Inicial (ambos estudiantes)

2. Completar el grafo con los tres costos para cada arco a partir de los datos de Uber y la malla vial de Bogotá. Crear un archivo JSON para guardarlo y un método que permita cargar el nuevo grafo JSON generado.
3. Dada una localización geográfica con latitud y longitud, encontrar el Id del Vértice de la malla vial más cercano por distancia Haversine. Esta solución se deberá utilizar en algunos requerimientos de la parte A y parte B del proyecto.

Requerimientos - Parte A (estudiante 1 de cada grupo)

4. Encontrar el camino de costo mínimo (menor tiempo promedio según Uber en la ruta) para un viaje entre dos localizaciones geográficas de la ciudad ((lat,long) origen, (lat, long) destino), ingresados por el usuario. Estas localizaciones deben aproximarse a las localizaciones más próximas en la malla vial.
   Respuesta en consola: Muestre en la consola de texto el camino a seguir, informando el total de vértices, sus vértices (Id, latitud, longitud), el costo mínimo (menor tiempo promedio en segundos) y la distancia estimada (sumatoria de distancias Haversine en Km).
   Visualización mapa: Muestre el camino resultante en Google Maps (incluyendo la ubicación de inicio y la ubicación de destino).

5. Determinar los *n* vértices con menor velocidad promedio en la ciudad de Bogotá. Siendo la velocidad promedio de un vértice *v,* el promedio de las velocidades de todos sus arcos. El parámetro *n* es un dato de entrada dado por el usuario.
   Respuesta en consola: Mostrar los *n* vértices resultantes en la consola de texto (su identificador, su ubicación (latitud, longitud), ordenados de menor a mayor por la velocidad promedio del vértice.
   Informar el número de componentes conectados (subgrafos) que se definen entre estos vértices en el grafo original. Por cada componente informar los identificadores de los vértices que la componen.
   Visualización mapa: marque la localización de los *n* vértices resultantes en un mapa en Google Maps usando un color 1. Destaque la componente conectada más grande (con más vértices) usando un color 2. Para esta componente muestre sus vértices y sus arcos.

6. Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Prim, aplicado al componente conectado (subgrafo) más grande de la malla vial de Bogotá.
   Respuesta en consola: Muestre en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos), y la siguiente información del árbol generado: el total de vértices en el componente, los vértices (identificadores), los arcos incluidos (Id vértice inicial e Id vértice final) y el costo total (distancia en Km) del árbol.
   Visualización mapa: Muestre árbol generado resultante en Google Maps: sus vértices y sus arcos.

Requerimientos - Parte B (estudiante 2 de cada grupo)

7. Encontrar el camino de menor costo (menor distancia Haversine) para un viaje entre dos localizaciones geográficas de la ciudad ((lat,long) origen, (lat, long), destino), ingresadas por el usuario. Estas localizaciones deben aproximarse a las localizaciones más próximas en la malla vial.
   Respuesta en consola: Muestre en la consola de texto el camino a seguir, informando el total de vértices, sus vértices (Id, latitud, longitud), el tiempo estimado (la sumatoria de los tiempos de sus arcos) y la distancia Haversine estimada (sumatoria de distancias Haversine en Km).
   Visualización mapa: Muestre el camino resultante en Google Maps (incluyendo la ubicación de inicio y la ubicación de destino).

8. A partir de las coordenadas de una localización geográfica de la ciudad (lat, lon) de origen, indique cuáles vértices son alcanzables para un tiempo T (en segundos) dado por el usuario. La localización de origen debe aproximarse a la localización más próxima en la malla vial.
   Respuesta en consola: Muestre en la consola los identificadores y la ubicación (lat, lon) de los vértices alcanzables en un tiempo T a partir de la localización de origen.

Visualización mapa: Marque la localización de origen en un color 1 y las localizaciones de los vértices alcanzables en un color 2 en Google Maps.

9. Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Kruskal, aplicado al componente conectado (subgrafo) más grande de la malla vial de Bogotá.
   Respuesta en consola: Muestre en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos), y la siguiente información del árbol generado: el total de vértices en la componente, los vértices (identificadores), los arcos incluidos (Id vértice inicial e Id vértice final) y el costo total (distancia en Km) del árbol.
   Visualización mapa: Muestre el árbol generado resultante en Google Maps: sus vértices y sus arcos.

Parte C (trabajo en grupo)

10. Construir un nuevo grafo simplificado No dirigido de las zonas Uber, donde cada zona (MOVEMENT_ID) es representada con un único vértice y los enlaces entre ellas representan su vecindad dentro de la malla vial. Es decir, si existe al menos un arco entre los vértices idA y idB en la malla vial y estos vértices pertenecen a la zonaX y la zonaY de Uber respectivamente, entonces las zonas zonaX y zonaY son vértices del grafo de zonas y estas zonas son adyacentes (vecinas) en este grafo. NO considerar los arcos que definan auto-ciclos entre una misma zona. Para cada zona seleccione una única localización (latitud, longitud) de referencia de manera que pueda luego visualizarse en el mapa. Adicionalmente, el costo del arco entre dos zonas vecinas corresponde al tiempo de viaje promedio reportado por Uber entre dichas zonas, teniendo en cuenta los tiempos promedio para los días (domingo, lunes, …, sábado) en que Uber haya reportado viajes entre dichas zonas. En caso de NO existir ningún tiempo de viajes en los datos Uber entre las dos zonas, asumir un valor predefinido de 200 segundos. Entre dos vértices (zonas) adyacentes solo debe crearse un arco, sin importar que pueda haber muchas intersecciones en común entre dichas zonas.
    Respuesta en consola: Al final de la construcción del grafo de zonas, reportar la cantidad de vértices y arcos (cada arco debe contarse una única vez).
    Visualización mapa: Muestre el grafo resultante en Google Maps: sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia.

11. Calcular el camino de costo mínimo (algoritmo de Dijkstra) basado en el tiempo promedio entre una zona de origen y una zona de destino sobre el grafo de zonas.
    Respuesta en consola: Muestre en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos) y del camino resultante: su secuencia de vértices/zonas (MOVEMENT_ID) y su costo total (sumatoria de tiempos de los arcos en segundos).

Adicionalmente, mostrar el tiempo promedio desde la zona origen hasta la zona destino reportado por el archivo de resolución semanal de Uber (teniendo en cuenta los tiempos promedio para los días (domingo, lunes, …, sábado) en que Uber haya reportado viajes entre dichas zonas).
Esto con el propósito que el usuario pueda comparar el tiempo estimado en el grafo de zonas y el tiempo estimado con los datos de Uber entre la zona origen y la zona destino.
Visualización mapa: Muestre el camino de costo mínimo en Google Maps: sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia.

12. A partir de una zona origen, calcular los caminos de menor longitud (cantidad de arcos) a todas sus zonas alcanzables. De estos caminos, seleccionar el camino más largo (mayor cantidad de arcos); este será el camino desde la zona origen a su zona más distante (teniendo en cuenta el número de arcos) en el grafo de zonas.
Respuesta en consola: Muestre en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos) y del camino resultante más largo: su secuencia de vértices/zonas (MOVEMENT_ID) y su número total arcos. Si hay múltiples caminos con la mayor longitud (número de arcos), mostrar entre estos caminos aquel que llegue al vértice destino con menor MOVEMENT_ID.
Visualización mapa: Muestre el camino más distante desde la zona de origen en Google Maps: sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia.

Restricciones

- Los datos contenidos en los archivos sólo se pueden leer una vez
- Se deberá trabajar en Java 8
- El proyecto se debe implementar en Eclipse
- La entrada/salida de información adicionales se debe realizar por consola
- **No usar las colecciones del API Java.**

Entrega de Diseño (33% Nota del Proyecto)

- Fecha/Hora límite de entrega: **Noviembre 24, 11:59 p.m.**
- Repositorio Bitbucket/GitHub con el nombre de la forma Proyecto_3_201920_sec_Y_team_Z (reemplazar Y por el número de la sección de su curso y Z su número de grupo)
- Verificar que el repositorio tiene configurado como usuarios a los monitores y al profesor con acceso de lectura.
- Entregables:
    - Documento con:

        - Documentación de los requerimientos funcionales (incluyendo los datos de entrada, la descripción, los datos de salida y estimación de complejidad temporal de cada requerimiento funcional). **Nota: Mencionar y justificar las estructuras de datos utilizadas para los requerimientos del proyecto.**
        - Diseño de las Estructuras de Datos a utilizar (diagrama de clases UML, imagen)
        - Diseño de la Solución al proyecto (diagrama de clases UML, imagen)
    - Proyecto Eclipse/Java con:
        - Implementación y Pruebas Unitarias Automáticas de la Estructuras de Datos Grafo No Dirigido (Eclipse/Java).
        - Construcción del grafo de la malla vial completo (con los tres costos). Generación del grafo completo en formato JSON y su respectiva lectura. **Respuesta en consola**: Informar el total de vértices y total de arcos del grafo de la malla vial (contar una única vez cada arco). Informar el total de componentes conectadas en el grafo. Informar el número de vértices de las 5 componentes conectadas más grandes en el grafo (ordenadas de mayor a menor por su número de vértices). **Visualización mapa**: Mostrar el grafo formado por los vértices que estén geolocalizados dentro de la zona delimitada por las siguientes coordenadas: longitud min= -74.094723, longitud max= -74.062707, latitud min= 4.597714 y latitud max= 4.621360, y los arcos que conectan dichos vértices.
        - Hacer el desarrollo del proyecto en la rama (*branch*) master del repositorio. Como parte final de esta entrega, crear la **rama (*branch*) entrega-diseNo**. Verificar que en esta rama queda la copia de los entregables del proyecto. Los entregables en la rama entrega-diseNo No deben actualizarse después de la fecha/hora límite de esta entrega.

Entrega Final (67% Nota del Proyecto)

- Fecha/Hora límite de entrega: **Diciembre 9, 8:00 a.m.**
- Repositorio Bitbucket/GitHub Proyecto_3_201920_sec_Y_team_Z (reemplazar Y por el número de la sección de su curso y Z su número de grupo) creado para la entrega de diseño. El proyecto completo (documentación e implementación) debe estar accesible en la rama (branch) master.
- Entregables:
    - Proyecto Eclipse/Java con:
        - Implementación completa de los requerimientos funcionales
        - Los entregables en la **rama master** No pueden tener fecha de actualización posterior a la fecha/hora límite de esta entrega