Objetivo

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

Contexto

Debido al crecimiento continuo de las ciudades en población, infraestructura y servicios, las organizaciones y agencias que intervienen en su administración y funcionamiento tienen como preocupación contar con información actualizada en los aspectos que afectan la vida de sus ciudadanos: educación, salud, transporte, vivienda, infraestructura, entretenimiento, seguridad, economía, entre otras. Esta información permite mantener informados a sus ciudadanos en sus actividades comunes y a los administradores y autoridades locales tomar decisiones que mejoren la calidad de vida de sus ciudadanos.

El transporte es una de las problemáticas importantes en una ciudad. En particular, el transporte público debe ofrecer soluciones eficientes para transportar un alto volumen de la población, con buena calidad, para así aumentar la productividad de la ciudad y la calidad de vida de sus ciudadanos. Además, un buen servicio de transporte público puede incentivar la reducción del transporte privado.

La malla vial de una ciudad (representada como un grafo, donde los vértices son las intersecciones de las calles y los arcos las calles) son utilizadas en una gran cantidad de aplicaciones para navegación terrestre. Sin embargo, estas mallas viales no son fáciles de conseguir o de construir.

El tema del proyecto está relacionado con el manejo de información de la malla vial de la ciudad de Chicago (USA) y la generación de información para los ciudadanos en cuanto al servicio de transporte público. La malla vial debe ser modelada como un dígrafo el cual debe ser construido tomando como base la información suministrada por los servicios de taxi.

En este proyecto, vamos a construir una aplicación que le permita entender a los administradores y autoridades de la ciudad de Chicago un conjunto de consultas importantes sobre el servicio de taxis.

Las Fuentes de Datos

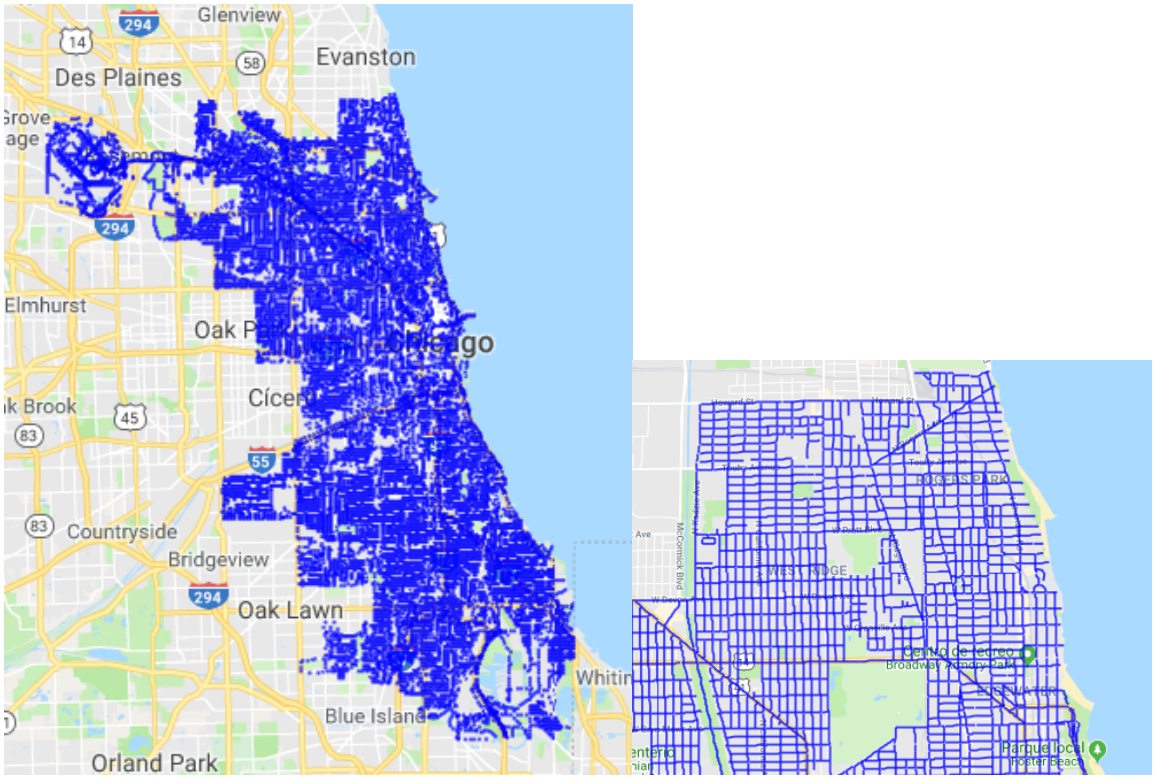
A continuación, se presenta una descripción de las fuentes de datos que se utilizarán en el proyecto.

- **Archivos de Servicios:** *taxi-trips-wrvz-psew-subset-small.json, taxi-trips-wrvz-psew-subset-medium.json, y taxi-trips-wrvz-psew-subset-large.json*

Cada fuente de datos contiene el detalle de un subconjunto de servicios de taxi (carreras) de periodos de longitud de tiempo diferentes. Cada servicio de taxi (carrera) se describe a partir de 23 datos entre los cuales se tienen: el identificador del taxi, el identificador del servicio, la empresa de afiliación del taxi, la ubicación geográfica de recogida y de terminación, la zona de la ciudad de recogida y de terminación, la fecha y la hora estimada de recogida, la fecha y hora estimada de terminación, su duración (en segundos), su distancia (millas), su costo total (US$), entre otros datos. Consulte el sitio Web https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew para conocer el detalle de la información que se incluye en cada servicio.

- **Archivos de Calles**: StreetLines.csv

Esta fuente de datos contiene el detalle del conjunto de las principales calles de la ciudad de Chicago. Cada calle se describe a partir de siete (7) datos entre los cuales se tienen: el identificador del nodo, nombre de la calle, tipo de la calle, sentido de la calle, calles de dirección única, geometría de la ruta. Este último aparte contiene todos los puntos (latitud, longitud) que permiten reconstruir la totalidad de la vía.



Tomado de: Chicago Data Portal
https://data.cityofchicago.org/Transportation/Street-Center-Lines/6imu-meau/data

Carga de Información

Para responder a los requerimientos presentados más adelante, usted deberá cargar la información del archivo .JSON generado por ustedes en el taller 8, el cual representa el grafo

de la ciudad de Chicago; así como el archivo de calles StreetLines.csv. Solo es permitido leer una vez la información de los archivos.

Requerimientos

0. Permita cargar el grafo dirigido de la malla vial de la ciudad de Chicago, creado en el taller 8.

1. Mostrar en la consola de texto la información del vértice más congestionado en Chicago (aquel que contiene la mayor cantidad de servicios que salen y llegan a él): su (latitud, longitud), total servicios que salieron y total de servicios que llegaron.

Visualización: marque la localización del vértice resultante en un mapa en Google Maps. Consultar la información de cómo graficar en Google Maps en los siguientes enlaces:

- https://developers.google.com/maps/
- https://developers.google.com/maps/web-services/client-library
- https://github.com/googlemaps/google-maps-services-java

2. Calcule los componentes fuertemente conexos presentes en el grafo. Asígnele un color a los vértices que componen un componente. Retorne una lista en donde en cada nodo, se tiene la información de un componente conexo (color, número de vértices que lo componen). Muestre en la consola de texto el total de componentes identificadas y la información de la lista de componentes conexo.

Visualización: Muestre la componente fuertemente conexa más grande (es decir, con más vértices del grafo) en un mapa en Google Maps. Pinte los puntos centro de cada vértice de la componente resultante y los arcos que los conectan en el mapa.

3. Visualización: A partir del grafo cargado al inicio y de los componentes conectados encontrados en el punto 2, genere un mapa coloreado de la red vial de Chicago utilizando Google Maps, de la siguiente forma:

a. Para cada vértice del grafo, calcule el radio del circulo de acuerdo a la "densidad" de servicios que salen y llegan a dicho vértice. Es decir que cada nodo tiene asociado un porcentaje de los servicios que salen y llegan, del total de servicios procesados. Tenga en cuenta que cada servicio se contabiliza en el vértice donde inicia y en el vértice donde termina. El tamaño del circulo a dibujar es proporcional a dicho porcentaje. El color del vértice es el color de la componente a la que pertenece.

b. El arco debe ser del color del componente conectado al cual pertenece el vértice del grafo donde inicia el servicio.

4. Encontrar el camino de costo mínimo (menor distancia) para un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Inicie el camino en el vértice del

grafo más cercano a su punto de inicio y finalícela en el vértice más cercano del grafo a su punto de destino. Muestre en la consola de texto el camino a seguir, informando sus vértices, el tiempo estimado, la distancia estimada y el valor estimado a pagar.

Visualización: muestre el camino resultante en Google Maps (incluyendo el punto original de inicio y el punto original de destino).

5. Dado un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Aproxime los puntos de inicio y fin a los vértices más cercanos en el grafo y encuentre los caminos de mayor y menor duración entre dichos puntos. Muestre en la consola de texto los dos caminos, y para cada uno de ellos: sus vértices, el tiempo estimado, la distancia estimada y el valor estimado a pagar.

Visualización: Muestre los caminos generados en Google Maps.

6. Dado un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Indique si existen caminos entre ambos puntos, en los que no deba pagar peaje. Si dichos caminos existen, retorne una lista con todos ellos, ordenadas de menor a mayor por tiempo y de mayor a menor por costo.

Visualización: Muestre el camino sin peajes de menor tiempo y el camino sin peajes de mayor costo en Google Maps.

Restricciones

- Los datos contenidos en los archivos sólo se pueden leer una vez
- Se deberá trabajar en Java 7
- 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.**