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, los asuntos relacionados con la regulación del transporte público; en especial las infracciones de tránsito; ya que, a través de estas las unidades administrativas buscan regular el comportamiento del transporte tanto público como privado.

El tema del proyecto está relacionado con el sistema de infracciones en movimiento implementado en la ciudad de Washington D.C. (USA). Esta ciudad está a la vanguardia en el registro de información en diferentes aspectos de su funcionamiento (portal oficial de datos *Open Data DC* http://opendata.dc.gov/). Para el análisis del sistema de infracciones utilizaremos como fuente de información el portal oficial de Open Data Web de consulta: http://opendata.dc.gov/datasets?q=moving%20violations.

En este proyecto, vamos a construir una aplicación que le permita gestionar datos de la malla vial de la ciudad de Washington D.C. (USA) utilizando la información del repositorio de infracciones móviles. La malla vial debe ser modelada como un grafo no dirigido el cual debe ser construido tomando como base la información suministrada.

Las Fuentes de Datos

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

Archivos de Infracciones

Los datos se pueden descargar del aula SicuaPlus Unificada en formato CSV (*Comma separated values*). Un archivo CSV es un archivo de texto donde cada línea tiene sus campos de información separados por punto y coma (;). Tenga en cuenta que estos **nuevos datos** incluyen el número de infracción al interior del archivo (OBJECTID_1 el cual es irrelevante) y la ubicación geográfica (LAT, LONG) de cada infracción.

A continuación, se muestra un ejemplo de uno de los archivos:

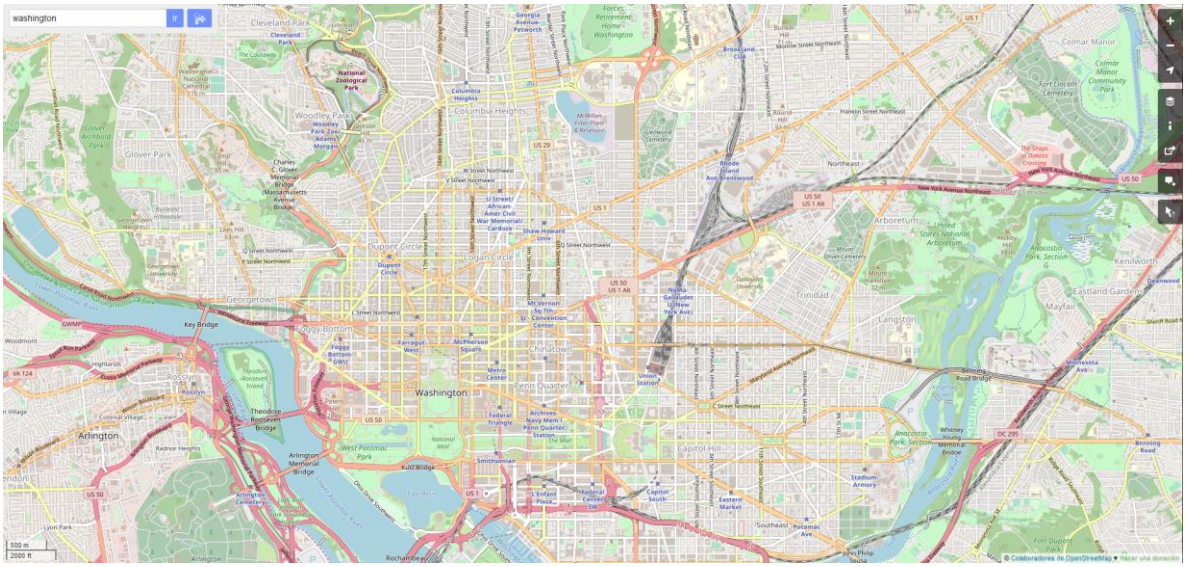
```
OBJECTID_1;OBJECTID;ROW_;LOCATION;ADDRESS_ID;STREETSEGI;XCOORD;YCOORD;TICKETTYPE;FINEAMT;TOTALPAID;PENALT
Y1;PENALTY2;ACCIDENTIN;AGENCYID;TICKETISSU;VIOLATIONC;VIOLATIOND;ROW_ID;LAT;LONG
1;15304300;;DC295            SW            .7            MILES            S/O            EXIT            1
S/B;810381;6993;398406,31000000000;127737,13000000000;Moving;100;0;0;;No;25;2018-12-
01T08:25:00.000Z;T119;SPEED 11-15 MPH OVER THE SPEED LIMIT;;38,81741557000;-77,01835340300
2;15304301;;NEW            YORK            AVE            W/B            @            NEW            JERSEY
NW;808095;6936;398718,38250000000;137500,57230000000;Moving;150;0;0;;No;25;2018-12-
01T21:24:00.000Z;T113;FAIL TO STOP PER REGULATIONS FACING RED SIGNAL;;38,90536926500;-77,01477813300
3;15304302;;NEW            YORK            AVE            W/B            @            NEW            JERSEY
NW;808095;6936;398718,38250000000;137500,57230000000;Moving;150;0;0;;No;25;2018-12-
02T11:15:00.000Z;T113;FAIL TO STOP PER REGULATIONS FACING RED SIGNAL;;38,90536926500;-77,01477813300
```

Los atributos son:

- OBJECTID_1: Número de la infracción en el archivo. Nueva propiedad.
- OBJECTID: Identificador único de la infracción.
- ROW_:
- LOCATION: Dirección en formato de texto.
- ADDRESS_ID: ID de la dirección.
- STREETSEGID: ID del segmento de la calle.
- XCOORD: Coordenada X donde ocurrió (No corresponde a longitud geográfica).
- YCOORD: Coordenada Y donde ocurrió (No corresponde a latitud geográfica).
- TICKETTYPE:
- FINEAMT: Cantidad a pagar por la infracción USD.
- TOTALPAID: Cuanto dinero efectivamente pagó el que recibió la infracción en USD.
- PENALTY1: Dinero extra que debe pagar el conductor.
- PENALTY2: Dinero extra que debe pagar el conductor.
- ACCIDENTINDICATOR: Si hubo un accidente o no.
- AGENCYID:
- TICKETISSUEDATE: Fecha cuando se puso la infracción.
- VIOLATIONCODE: código de la infracción.
- VIOLATIONDESC: descripción textual de la infracción.
- ROW_ID:
- LAT: Corresponde a latitud geográfica. Nueva propiedad.
- LONG: Corresponde a longitud geográfica. Nueva propiedad.

Grafo de Calles

Se tomará como fuente de datos el portal OpenStreetMap (https://www.openstreetmap.org/). A partir de esta fuente de datos (formato XML) se construyeron los grafos No Dirigidos del Centro (downtown) y de la ciudad completa de Washington D.C. como resultado del taller 8. Estos grafos se guardaron en formato JSON.



Tomado de: OpenStreetMap
https://www.openstreetmap.org/#map=14/38.9070/-77.0151

Requerimientos

0. Cargar el Grafo No Dirigido (grafo más grande) de la malla vial de la ciudad completa de Washington D.C. (formato JSON), creado en el taller 8. Informar el total de vértices y el total de arcos que definen el grafo cargado. Solo es permitido leer una vez la información de los archivos.

1. Al grafo creado, se debe agregar la información de cada una de las infracciones de todos los meses del año (nuevos archivos *.csv). Para este fin, ubique el vértice más cercano a la ubicación geográfica de la infracción y sobre este vértice almacene la información de la infracción que considere relevante. Utilice la **distancia harvesiana** (en kilómetros) entre dos ubicaciones geográficas para calcular la distancia entre la ubicación de un vértice del grafo y la ubicación de una infracción.
Informar el total de infracciones cargadas y el número de las infracciones de cada mes.

Requerimientos - Parte A (estudiante 1 de cada grupo)

2. Encontrar el camino de costo mínimo (menor cantidad de infracciones en la ruta) para un viaje entre dos ubicaciones geográficas (latitud, longitud), escogidas aleatoriamente al interior del grafo.

Respuesta en consola: Muestre en la consola de texto el camino a seguir, informando sus vértices (Id, Ubicación Geográfica), el costo mínimo (menor cantidad de infracciones), y la distancia estimada (en Km).
Visualización mapa: Muestre el camino resultante en Google Maps (incluyendo la ubicación de inicio y la ubicación de destino).

3. Determinar los *n* vértices con mayor número de infracciones en la ciudad de Washington D.C. *n* es un dato de entrada dado por el usuario. Adicionalmente identificar las componentes conectadas (subgrafos) que se definan únicamente entre estos *n* vértices. Las componentes conectadas (subgrafos) solo pueden usar arcos del grafo original que conecten estos *n* vértices.
   Respuesta en consola: Mostrar los *n* vértices resultantes en la consola de texto (su identificador, su ubicación (latitud, longitud), y el total de infracciones) ordenados de mayor a menor por el número de infracciones.
   Informar el número de componentes conectadas (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 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.

Requerimientos - Parte B (estudiante 2 de cada grupo)

4. Encontrar el camino más corto (menor número de vértices) para un viaje entre dos ubicaciones geográficas (latitud, longitud), escogidas aleatoriamente al interior del grafo.
   Respuesta en consola: Muestre en la consola de texto el camino a seguir, informando el total de vértices, sus vértices (Id, Ubicación Geográfica) y la distancia estimada (en Km).
   Visualización mapa: Muestre el camino resultante en Google Maps (incluyendo la ubicación de inicio y la ubicación de destino).

5. A partir de las coordenadas de un área de interés de la ciudad (LonMin, LatMin) y (LonMax, LatMax) y dos valores enteros (N >= 2 y M >= 2), definir una cuadricula regular de N columnas (incluyendo LonMin y LonMax) por M filas (incluyendo LatMin y LatMax). Las intersecciones de la cuadrícula así definida contienen N x M ubicaciones geográficas separadas de forma uniforme en el área de interés (incluyendo sus límites).
   Aproximar estas N x M ubicaciones a los vértices más cercanos en el grafo (usar la distancia harvesiana). El resultado de la aproximación va dar un conjunto de máximo N x M vértices en el grafo.
   Respuesta en consola: Muestre en la consola de texto el número de vértices en el grafo resultado de la aproximación. Mostar el identificador y la ubicación geográfica de cada uno de estos vértices.

Visualización mapa: Marque las ubicaciones de los vértices resultantes de la aproximación de la cuadrícula en Google Maps.

Parte C (trabajo en grupo)

6. Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Kruskal, aplicado a la componente conectada (subgrafo) más grande encontrada en el punto 3.
   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: 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.

7. Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Prim, aplicado a la componente conectada (subgrafo) más grande encontrada en el punto 3.
   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: 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.

8. Calcular los caminos de costo mínimo (algoritmo de Dijkstra) con criterio distancia que conecten los vértices resultado de la aproximación de las ubicaciones de la cuadricula N x M encontrados en el punto 5.
   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 de cada camino resultante: su secuencia de vértices (identificadores) y su costo (distancia en Km).
   Visualización mapa: Muestre los caminos de costo mínimo en Google Maps: sus vértices y sus arcos. Destaque el camino más largo (en distancia) usando un color diferente.

9. Encontrar el camino más corto (con criterio menor número de infracciones en la vía y menor cantidad de vértices) para un viaje entre dos ubicaciones geográficas (latitud, longitud), escogidas aleatoriamente al interior del grafo.
   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 camino resultante: su secuencia de vértices (identificadores), el total de infracciones y la distancia calculada (en Km).

Visualización mapa: Muestre el camino resultante en Google Maps: sus vértices y sus arcos.

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: **Mayo 6, 11:59 p.m.**
- Repositorio Bitbucket/GitHub con el nombre de la forma Proyecto_3_201910_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 las Estructuras de Datos (Eclipse/Java).
        - Lectura/carga de los datos para un semestre dado por el usuario (mostrar la evidencia que los datos fueron leídos/cargados)
        - 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: **Mayo 20, 6:00 a.m.**
- Repositorio Bitbucket/GitHub Proyecto_3_201910_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