Universidad de los Andes

Ingeniería de Sistemas y Computación
ISIS1206 – Estructura de Datos
Taller 8

Taller 8: Grafos

Objetivos

- Estudiar, implementar, probar y documentar la estructura de datos grafo dirigido
- Utilizar adecuadamente herramientas para el desarrollo de software en equipos

Lectura Previa

Estudiar la teoría de los grafos dirigidos con peso. Consultar las secciones 4.1 a 4.4 del libro guía *Algorithms* de Sedgewick y Wayne.

Lo que su grupo debe hacer (parejas)

Parte 1 – Trabajo en casa

1. Cree en bitbucket un repositorio llamado T8_201190. Al momento de crearlo recuerde la URL que se muestra en la parte superior derecha de la página de bitbucket: Por ejemplo
   Repository_url = https://login-usuario@bitbucket.org/login-usuario/T8_201910.git
   donde login-usuario corresponde a su login en Bitbucket.
2. Cree el README del repositorio donde aparezcan los nombres y códigos de los miembros del grupo de trabajo.
3. Realice el procedimiento para crear el directorio en su computador de trabajo para que relacione este directorio con el repositorio remoto T8_201910. El trabajo debe repartirse entre ambos estudiantes del grupo.
4. Implemente la estructura de datos Grafo(), la cual representa un grafo No Dirigido con pesos. El API que debe implementar está compuesto, al menos, por las siguientes operaciones:

Operación	Descripción
`Graph ( )`	Crea un grafo No dirigido sin vértices y sin arcos
`int V()`	Número de vértices
`int E()`	Número de arcos. Cada arco No dirigido debe contarse una única vez.
`void addVertex( K idVertex, V infoVertex)`	Adiciona un vértice con un Id único. El vértice tiene la información InfoVertex

Operación	Descripción
`void addEdge(K idVertexIni, K idVertexFin, A infoArc )`	Adiciona el arco No dirigido entre el vertice IdVertexIni y el vertice IdVertexFin. El arco tiene la información infoArc.
`V getInfoVertex(K idVertex)`	Obtener la información de un vértice
`void setInfoVertex(K idVertex, V infoVertex)`	Modificar la información del vértice idVertex
`A getInfoArc(K idVertexIni, K idVertexFin)`	Obtener la información de un arco
`void setInfoArc(K idVertexIni, K idVertexFin, A infoArc)`	Modificar la información del arco entre los vértices idVertexIni e idVertexFin
`Iterator<K> adj(K idVertex)`	Retorna los identificadores de los vértices adyacentes a idVertex

5. Diseñe escenarios de prueba para probar la estructura.
6. Construya sets de pruebas (es decir casos de prueba en JUnit) para verificar y validar la implementación del API.

Parte 2 – Trabajo en clase

La malla vial de una ciudad (representada como un grafo, donde los vértices son las intersecciones de las calles y los arcos las calles) puede ser utilizada en una gran cantidad de aplicaciones para navegación terrestre. El objetivo de esta parte del trabajo es construir un **Grafo No Dirigido** que represente la malla vial de Washington D.C., tomado como base la información del archivo *Central-WashingtonDC-OpenStreetMap.xml* del portal OpenSteetMap (https://www.openstreetmap.org/). A continuación, se muestra un ejemplo de uno de los archivos:

1. Se representarán las **intersecciones de la malla vial** como los **vértices del grafo**. Estas intersecciones están representadas en el archivo XML con la etiqueta **node**

```
<node id="49716126" lat="38.8963324" lon="-77.0380200" version="7" timestamp="2015-06-06T23:50:51Z" changeset="31779161" uid="152074" user="beweta"/>
```

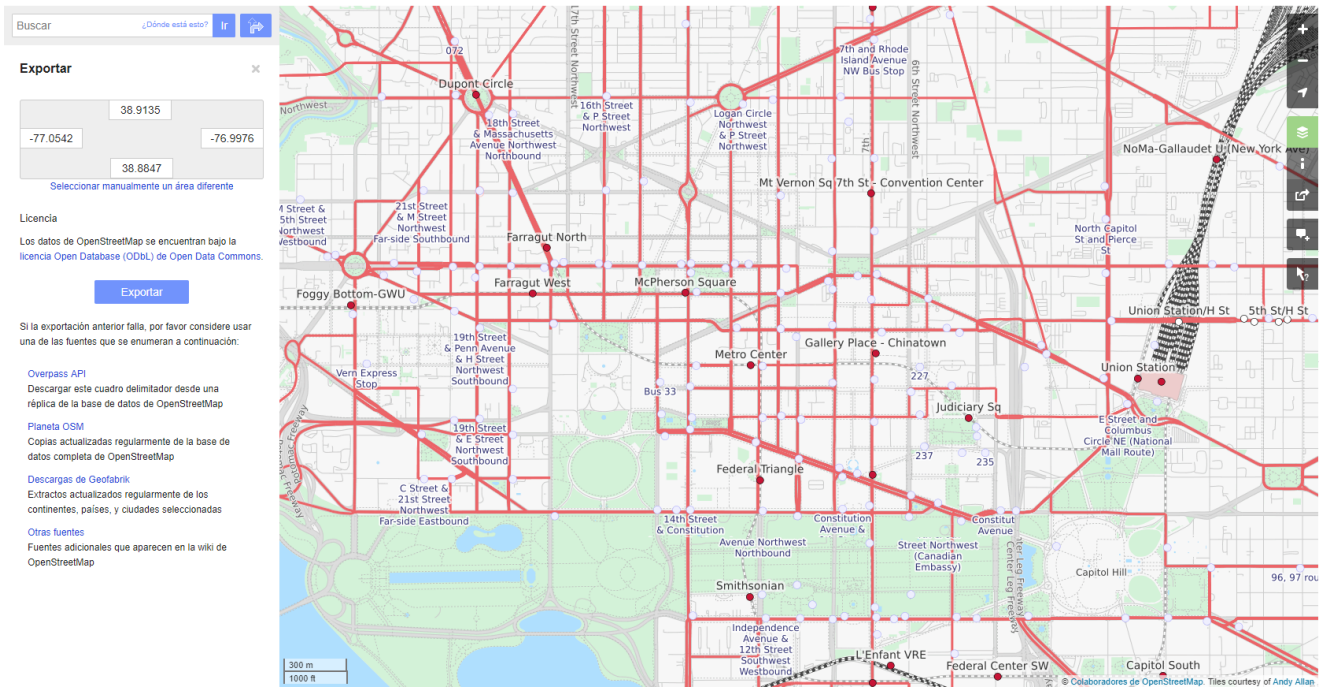
   De cada intersección se debe almacenar solamente tres datos: el identificador único de la intersección (id), latitud (lat) y longitud (lon). Cada vértice es de tipo intersección.

2. El grafo de la **malla vial de la ciudad de Washington D.C.** se debe construir a partir de la carga de las vías relacionadas en el archivo XML que se genera en el portal**.** Estas vías se almacenan bajo la etiqueta **way** del archivo y solamente se deben considerar aquellos **way** que adicionalmente contienen el atributo "**highway**"; de esta etiqueta se deberá extraer el id, y los vértices que la conforman (representados por la etiqueta **nd**). La conexión entre cada par de vértices de una vía se va considerar válida en sus dos sentidos (doble vía) y cada conexión debe contener su distancia harvesiana entre sus dos vértices.

```
<way id="6051287" version="13" timestamp="2017-07-15T20:27:20Z" changeset="50313523" uid="2601744" user="sctrojan79">
    <nd ref="49716661"/>
    <nd ref="3010704392"/>
    <nd ref="2350027403"/>
```

```
    <tag k="access" v="private"/>
    <tag k="highway" v="service"/>
    <tag k="name" v="South Capitol Street"/>
    <tag k="sidewalk" v="no"/>
    <tag k="tiger:cfcc" v="A41"/>
    <tag k="tiger:county" v="District of Columbia, DC"/>
    <tag k="tiger:name_base" v="Capitol"/>
    <tag k="tiger:name_direction_prefix" v="S"/>
    <tag k="tiger:name_direction_suffix" v="SE"/>
    <tag k="tiger:name_type" v="St"/>
    <tag k="tiger:reviewed" v="no"/>
    <tag k="tiger:zip_left" v="20003"/>
    <tag k="tiger:zip_right" v="20003"/>
  </way>
```

El grafo debe ser almacenado utilizando listas de adyacencia.



Sugerencia 1: Utilizar el API SAX (https://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html) para Java.

3. Reportar el número de vértices intersección y el número de arcos del grafo inicial construido en el documento Leame.txt.

4. Defina un esquema JSON para persistir el último grafo; este grafo extendido se va a utilizar en el proyecto 3.

5. Cargue el grafo persistido en el paso 4, leyendo el archivo JSON definido por usted y verifique que

el grafo quedó bien creado en memoria.

6. Grafique con ayuda de Google Maps el grafo resultante de la carga de la zona delimitada por las siguientes coordenadas: Longitud min= -77.0542, Longitud max= -76.9976, Latitud min= 38.8847 y Latitud max: 38.9135. Pintar los nodos intersección (con círculos), y los arcos (que conectan los nodos).

   Sugerencia 2: Utilizar el API JxMaps (https://www.teamdev.com/jxmaps) ó mapbox (https://docs.mapbox.com/android/java/overview/#installation) para Java

   Restricción de uso: El uso de JxMaps tiene una restricción de tiempo en la licencia. En la página web (https://www.teamdev.com/jxmaps#evaluate) un estudiante puede solicitar una licencia de proyecto académico, gratuita, pero esta licencia tiene una restricción de uso de **30 días**.

   Importante: Como el taller 8 y el proyecto 3 van a tener el requerimiento de despliegue de mapas de Google Maps, hay que administrar el tiempo de su licencia para que alcancen a hacer los 2 desarrollos. Para los grupos se les recomienda primero solicitar la licencia de uno de los estudiantes y si se requiere luego solicitar la licencia del segundo estudiante. Para los grupos de 1 solo estudiante se recomienda hacer el mejor uso posible de su licencia en los **30 días que tienen de licencia**.

7. Construir un **Grafo No Dirigido** que represente la malla vial de Washington D.C., tomado como base la información del archivo *map.xml* del portal OpenSteetMap (https://www.openstreetmap.org/) y genere el archivo JSON correspondiente.

Entrega

1. Para hacer la entrega del taller usted debe agregar a su repositorio los usuarios de los monitores y su profesor, siguiendo las instrucciones del documento “Guía Creación de Repositorios para Talleres y Proyectos.docx”.
2. Entregue su taller por medio de BitBucket. Recuerde, si su repositorio No tiene acceso al profesor ni a los monitores, su taller no será calificado por más de que lo haya desarrollado.