ITBA Maps

Objetivo

Implementar un algoritmo para, dadas las coordenadas de dos puntos en Capital Federal encontrar la mejor forma de ir de uno al otro usando transporte público.

Requerimientos

Implementación

Se recibirá de la cátedra una base de un proyecto que contará con una clase con dos métodos:

```
List<BusInPath> findPath(double fromLat, double fromLng, double
toLat, double toLng);

List<PlaceLocation> findPlaces(String searchTerm);
```

Las clases *BusInPath* y *PlaceLocation* se encontrarán en el código.

El método `findPath` recibe dos puntos geográficos correspondientes a CABA y devuelve, si logra encontrar un camino, una lista con los colectivos que un usuario puede tomarse para ir de origen a destino. El camino elegido debe minimizar un criterio elegido. Algunas opciones para este criterio son la cantidad de combinaciones, la cantidad de paradas por las que se pasa, la distancia caminada, la distancia recorrida arriba del colectivo o alguna fórmula más sofisticada que combine algunas de estas variables y otras. Este criterio debe estar detallado en el informe y se evaluará que el criterio elegido coincida con el comportamiento del programa.

Para ello se puede obtener los datos de las paradas de los colectivos desde la siguiente página de la ciudad de Buenos Aires:
https://data.buenosaires.gob.ar/dataset/colectivos-paradas

Una vez que se tengan los datos se deberá armar un grafo. Para esto es importante pensar en qué representa un nodo y qué una arista (por ejemplo, un nodo podría ser una esquina, una región del mapa, un colectivo, una manzana, un barrio, etc). Para el armado del grafo no se debe tardar más de unos minutos como máximo y debe realizarse una única vez al iniciar el programa.

A partir de que el grafo esté armado, los llamados a *findPath* deberán responder en menos de 2 segundos el camino más corto usando transporte público entre los puntos de origen y destino. Para hacerlo se propone implementar el algoritmo de *Dijkstra* que permite encontrar ese camino.

Por otro lado, el método `findPlaces` recibe un *String* de búsqueda y debe devolver la ubicación y nombre de los puntos de interés que más se adecúen a la búsqueda (máximo 10 lugares). Estos puntos pueden tomarse de cualquier dataset que tenga nombre y ubicación geográfica de lugares en CABA, una opción es el siguiente dataset de espacios culturales: https://data.buenosaires.gob.ar/dataset/espacios-culturales Se debe analizar cual de los métodos para la búsqueda en texto visto en clase es más adecuado e implementarlo.

Interfaz visual

El código provisto por la cátedra levanta un servidor web que escucha en el puerto 4567 y expone dos métodos, los dos mencionados anteriormente. Este servidor web es consumido desde la web http://itbamaps.com/

La página permite seleccionar dos puntos en un mapa interactivo haciendo click en el mapa o buscando puntos de interés con texto. Al apretar el botón de *Buscar* se hará un request al servidor corriendo local que deberá responder con los transportes a usar para el trayecto, los cuales aparecerán en la pantalla.

Cualquier cosa que se quiera modificar sobre el código provisto se puede hacer, pero siempre deberá seguir funcionando con la página web.

Consideraciones

No se pueden usar librerías externas salvo las usadas para el parseo de los archivos de entrada y las que ya se encuentran en el pom provisto por la cátedra.

Para leer los datos de las paradas de colectivo y puntos de interés se puede utilizar cualquier método, pero se recomienda descargar los CSV de la página del la ciudad y parsearlo usando alguna librería como OpenCSV o Apache Commons CSV.

Criterio de evaluación

Se podrán obtener hasta 5 puntos por una versión funcional del algoritmo de Dijkstra que retorne un recorrido en colectivo que pueda ser razonablemente considerado como bueno (es decir, que no proponga combinar entre 3 colectivos o hacer un recorrido muy grande si hay una opción directa). Algunos de estos recorridos deberán hacer combinación de colectivos, por lo que se debe probar que ande bien eso.

Además, se pueden sumar puntos por:

- 3 puntos por una versión funcional de la búsqueda con texto de puntos de interés.
- 1,5 puntos por también incluir subtes. Buscar los recorridos en la página de datos de la ciudad.
- 1,5 puntos por tener cobertura de tests de unidad sobre todas las partes críticas del código (el algoritmo de búsqueda de colectivos usando Dijkstra y el algoritmo de búsqueda en texto).
- 1 punto por detallar la complejidad de las partes algorítmicas del código (dijkstra + búsqueda en texto)

Si se suman más de 10 puntos en total la nota final será 10.

Entrega

El trabajo se podrá realizar en grupos de hasta 3 alumnos (también pueden ser de 2 o individuales).

La entrega deberá incluir:

- Un zip con el código.
- Un pequeño documento (1-2 carillas) con una explicación breve del código, el criterio elegido para seleccionar el camino más corto, cuáles son las clases importantes que contienen la lógica principal y cuáles de los puntos opcionales de la entrega se realizaron. En caso de haber calculado la complejidad incluirlo en ese documento.

La fecha de entrega es el domingo 11 de octubre a última hora. Si no se llega a esa fecha, se podrá entregar la semana siguiente (18 de octubre) con una penalidad de 2 puntos.

Se deberá enviar la entrega por Campus en la actividad creada para tal fin.