Universidad Simón Bolívar
Departamento de Computación y
Tecnología de la Información
CI-2693: Laboratorio de Algoritmos III

Prof. Fernando Torre Mora

Proyectos 2 y 3 Enero – Marzo 2020

Planificador de Rutas

1 Introducción

Diego acaba de mudarse a Caracas. Diego no tiene carro y no puede caminar mucho por razones médicas, por lo que debe moverse en transporte público. Sin embargo, no logra entender cómo funciona, por lo que siempre llega tarde. Diego necesita saber cómo llegar de un punto A a un punto B.

El transporte público en Caracas actualmente enfrenta una crisis que disminuye la frecuencia de los vehículos. Siendo del interior, a Diego no le gusta esperar en la parada. La crisis también resulta en que muchas de las líneas que salen en los mapas en realidad no funcionan, por lo que debe ser posible limitar la búsqueda sólo a las que sí. Sin embargo, estas líneas se podrían reactivar en cualquier momento.

Diego sueña con una aplicación que tan solo le diga qué línea tomar y dónde bajarse para poder llegar a tiempo; usted ha sido contactado para realizar este desarrolla. Se necesita encontrar la mejor rutas entre dos paradas, sin embargo, ya que Diego tiene confusión sobre qué ruta es "la mejor", se le pide que realice dos aplicaciones:

- **Proyecto 2:** Debido a la crisis que aqueja al sector, deberá buscar minimizar el número de transbordos entre las líneas operativas. Para comparación, deberá indicar la ruta a seguir cuando todas las líneas están operativas.
- **Proyecto 3:** Dado que la ruta con menos transbordos no siempre es la mejor, deberá calcular la ruta más rápida considerando el costo de cada parada y tramo. Para comparación, deberá indicar el tiempo que tomaría la ruta con menos transbordos.

2 Requerimientos de la implementación

2.1 Proyecto 2

Para el Proyecto 2, usted debe desarrollar un programa que, dado un mapa, líneas operativas y los nombres de dos paradas, indique las líneas que se deben tomar, y dónde realizar la transferencia entre ellas.

El programa se debe poder ejecutar desde la cónsola con el siguiente comando:

```
java PlanearTransbordos <mapa> <líneas> <pIni> <pFin>
```

donde

- <mapa> es el grafo de todas las paradas y las líneas que pasan por ellas
- <líneas> es la lista de líneas a considerar
- <pIni> es el nombre de la parada inicial
- <pFin> es el nombre de la parada final

Deberá almacenar el grafo completo, y realizar una copia generando el grafo inducido sólo por las líneas del archivo de líneas. Usando un algoritmo de recorrido de grafos, encuentre una ruta desde pIni hasta pFin en ambos grafos. Para minimizar el número de transbordos, verifique las rutas alternas usando backtracking. Algunas señales de que es necesario backtracking incluyen: que una línea aparezca dos veces (por ejemplo: de A a B usando la línea 1, de B a C usando línea 2, y de C a D usando línea 1), que se use una línea contenida enteramente en otra (suponga que la línea 2 sólo va de B a C), o que se requiera hacer transbordos aunque pIni y pFin sean de la misma línea.

Suponga que todas las paradas de una línea se encuentran operativas. Suponga que subgrafo dado por cada línea siempre es conexo. Puede suponer que los nombres de las paradas son únicos.

Si no es posible ir desde pIni hasta pFin con las líneas consideradas, debe indicarse que no existen rutas por transporte público entre esas dos paradas.

Este programa deberá ignorar los tiempos.

2.2 Proyecto 3

Para el Proyecto 3, usted debe desarrollar un programa que, dado un mapa, líneas operativas y los nombres de dos paradas, indique las líneas que se deben tomar, y dónde realizar la transferencia entre ellas.

El programa se debe poder ejecutar desde la cónsola con el siguiente comando:

```
java MejorRuta <mapa> <líneas> <pIni> <pFin>
```

Deberá recorrer el grafo inducido usando un algoritmo de costo mínimo informado como A* ó IDA*. El costo de un vértice se suma cada vez que se hace transferencia, y el de una arista cada vez que este se recorre.

3 Formato de Archivo

Se proporcionan siempre dos archivos: uno conteniendo el mapa de paradas y líneas, representado como un grafo, y uno conteniendo la lista de líneas a considerar.

En el archivo del mapa, cada parada se representa como un vértice y cada línea se representa como una serie de tramos del mismo tipo. Cada tramo es una arista del grafo. Si dos líneas van entre las dos mismas paradas, habrá dos aristas conectándolas.

El archivo líneas es simplemente una lista lineal de líneas

3.1 Archivo de Mapa

El formato del archivo que contiene el mapa es el siguiente:

```
o
n
m
p1 np1 lonp1 latv1 pp1
p2 np2 lonp2 latv2 pp2
⋮
vn nvn xvn yvn pvn
ut1 vt1 lt1 pt1
ut2 vt2 lt2 pt2
```

⋮

ut_m vt_m tt_m pt_m

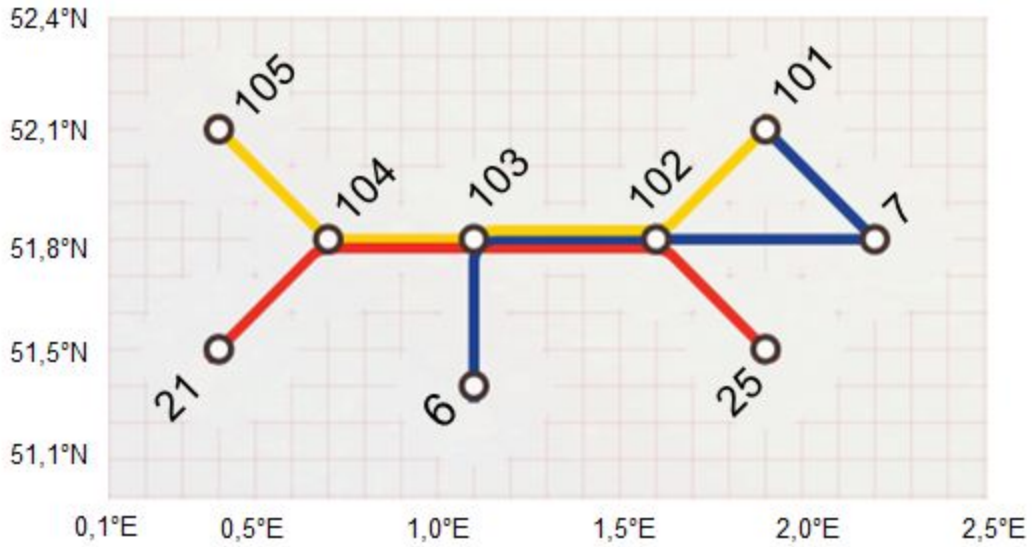
donde

- o es una letra (“d” ó “n”) que indica si los tramos se pueden conducir en un solo sentido (“d”) o en ambos sentidos (“n”)
- n es el número de paradas
- m es el número de tramos
- p_i es el número de una parada
- np_i es el nombre de la parada p_i
- $lonp_i$ es la longitud de la parada p_i
- $latv_i$ es la latitud de la parada p_i
- pp_i es el tiempo que se espera al intentar abordar en la parada p_i
- ut_i es el número de la primera parada del tramo t_i
- vt_i es el número de la segunda parada del tramo t_i
- lt_i es la línea que travieza el tramo t_i
- pt_i es el tiempo que toma atravesar el tramo t_i con un vehículo de la línea l.

El tiempo de espera en una parada sólo aplica si es la primera parada del recorrido, o si se está realizando un transbordo en esa parada; así, el tiempo total de un recorrido es la suma de los tiempos de los tramos, los tiempos de las paradas donde se hace transbordo y la estación inicial.

3.2 Ejemplo de Mapa

Suponga el siguiente subgrafo del Sistema de Transporte de Londres



El contenido del archivo sería:

```
n                                       105 104 Yellow 6
9                                       21 104 Red 6
12                                      104 103 Yellow 10
105 Warwick 0.4 52.1 3                  104 103 Red 10
21 Westbourne 0.4 51.5 6                103 6 Blue 40
104 Paddington 0.7 51.8 18              103 102 Yellow 12
103 BakerStreet 1.1 51.8 10             103 102 Red 15
6 RegentsPark 1.1 51.4 10               103 102 Blue 19
102 KingsCross 1.6 51.8 36              102 101 Yellow 7
101 CaledonianRoad 1.9 52.1 3           102 7 Blue 28
25 Moorsgate 1.9 51.5 6                 102 25 Red 11
7  Highbury&Islington  2.3  51.8        101 7 Blue 7
10
```

3.3 Archivo de líneas

El formato del archivo de líneas es simplemente una lista separada por saltos de línea. Puede suponer que todas las líneas que aparecen en el mismo, aparecen en el mapa. Puede haber líneas que aparezcan en el mapa, pero no en el archivo de líneas

3.4 Ejemplo de líneas

```
Yellow
Red
Blue
```

4 Formato de Salida

Su salida sólo debe indicar en qué estación abordar, cuál línea, y en qué estación hacer transbordo. Se recomienda el siguiente formato:

```
Tome la linea LINEA desde ID1 PARADA1 hasta ID2 PARADA2
```

Recuerde que, si no existe forma de llegar desde la parada inicial hasta la parada final (por ejemplo, el grafo inducido es disjunto), deberá imprimir que no existen rutas de transporte público entre ellas

4.1 Ejemplo Proyecto 2

Para ilustrar, suponga que se desea viajar desde la estación 21 (Westbourne) hasta la Estación 101 (CaledonianRoad). Solo hay una línea que sale de Westbourne, pero no llega hasta Caledonian Road. Debido a que los algoritmos de recorrido de grafo no distinguen entre líneas, es muy probable que la primera respuesta recomiende el arco Amarillo entre 104 y 103, el arco azul entre 103 y 102 y, finalmente, el arco Amarillo entre 102 y 101. Dado a que esto es un transbordo más de lo necesario, debe realizarse backtracking. Supongamos que el backtracking elije simplemente el siguiente arco que fue agregado primero: descartar el arco 102 – 101 hará que se considere el camino 102 – 7 – 101. Suponga que el camino finalmente escogido incluye 102 – 7 – 101.

Un formato de impresión posible:

```
java PlanearTransbordos Londres.txt Lineas.txt Westbourne CaledonianRoad
Tome la linea Red desde 21 Westbourne hasta 102 KingsCross
Tome la linea Blue desde 102 KingsCross hasta 101 Caledonian Road
Transbordos totales: 1

cuando todas las rutas están operativas:
Tome la linea Red desde 21 Westbourne hasta 102 KingsCross
Tome la linea Blue desde 102 KingsCross hasta 101 Caledonian Road
Transbordos totales: 1
```

4.2 Ejemplo Proyecto 3

Al considerar los costos, en cambio, se verá que los trenes de la línea amarilla tienen prioridad sobre cualquier otra en el tramo 103 – 102, por lo cual deben usarse para este tramo. También ayuda el hecho de que el tiempo de espera en Baker Street es mucho menor que en King's Cross. El tiempo total, por lo tanto es de 84 minutos:

- 6 minutos para abordar en 21 Westbourne
- 6 minutos de 21 Westbourne a 104 Paddington
- 10 minutos de 104 Paddington a 103 Baker Street
- 10 minutos de espera para hacer el transbordo en 103 Baker Street
- 12 minutos de 103 Baker Street a 102 King's Cross
- 7 minutos de 102 King's Cross a 101 Caledonian Road

Lo cual se puede imprimir de la siguiente manera:

```
java MejorRuta  Londres.txt Lineas.txt Westbourne CaledonianRoad
Tome la linea Red desde 21 Westbourne hasta 103 BakerStreet
Tome la linea Yellow desde 103 BakerStreet hasta 101 CaledonianRoad
Tiempo total: 84 minutos

Minimizando transbordos:
Tome la linea Red desde 21 Westbourne hasta 102 KingsCross
Tome la linea Blue desde 102 KingsCross hasta 101 Caledonian Road
Tiempo total: 108 minutos
```

5 Documentación

Todo el código debe estar debidamente documentado en formato Javadoc. Las clases deben resumir el propósito y estructura de las mismas. Se deben indicar una descripción del método, la descripción de los parámetros de entrada y salida, aplicando el estándar para la documentación de código en JAVA. Si hay alguna decisión de diseño que debe justificar (por ejemplo, preferir una clase `java.util` por encima de otra), deberá hacerlo en el javadoc del método o clase pertinente.

6 Entrega

Debe entregar el código correspondiente al **proyecto 2** antes del lunes, 23 de marzo (lunes de semana 12) en un archivo comprimido (.zip, .tgz, etc.) libre de archivos intermedios o ejecutables. Deberá subirlo al Moodle de la materia en la sección marcada como "Proyecto 2". Sólo deberá efectuar una entrega por grupo.

Debe entregar el código correspondiente al **proyecto 3** antes del lunes, 30 de marzo (lunes de semana 13) en un archivo comprimido (.zip, .tgz, etc.) libre de archivos intermedios o ejecutables. Deberá subirlo al Moodle de la materia en la sección marcada como "Proyecto 3". Sólo deberá efectuar una entrega por grupo.

Sus implementaciones deben ser razonablemente eficientes. Su implementación debe incluir manejo de excepciones. Puede usar las librerías de JAVA que considere útiles.

El día de la entrega de cada proyecto deberán entregar una "Declaración de autenticidad de la entrega" firmada por los autores del proyecto. Esto significa que debe firmar una Declaración para el Proyecto 2, y otra para el Proyecto 3. El no cumplimiento de estos requisitos resultará en que su trabajo no sea evaluado.

7 Evaluación

En la evaluación del proyecto se tomará en cuenta aspectos como la documentación, el estilo de programación, la modularidad y mantenibilidad del código, la eficiencia en tiempo de ejecución y memoria, el uso de herencia, el manejo de excepciones, el buen uso de las librerías y la robustez.

Se asignan, para cada proyecto:

- 5 puntos por código
    - 1 punto por la generación del grafo inducido
    - 2 puntos por su algoritmo de recorrido de grafo
    - 2 puntos por su backtracking (en el caso del proyecto 2) o heurística (en el caso del proyecto 3)
- 12 puntos por ejecución
    - 2 puntos por reportar una ruta válida cuando en efecto existe
    - 2 puntos por reportar que no existe una ruta cuando en efecto no existe
    - 3 puntos por poder dar la ruta con los transbordos transbordos (en el caso del proyecto 2) o tiempo total (en el caso del proyecto 3) que, en efecto, sea el mínimo
    - 2 puntos por imprimir correctamente el número de transbordos (en el caso del proyecto 2) o tiempo total (en el caso del proyecto 3)
    - 1 punto por imprimir en el formato correcto
    - 2 punto por realizar la comparación adecuadamente
- 3 puntos por documentación
    - 1 punto por documentar de forma adecuada y completa sus clases
    - 1 punto por documentar de forma adecuada y completa sus métodos
    - 1 punto por explicar su backtracking (en el caso del proyecto 2) o heurística (en el caso del proyecto 3)