Metro_Madrid

April 30, 2021

1 Informática - Curso 2020/2021

1.1 Práctica 4 - Metro

1.1.1 Fecha de entrega:

Objetivos: clases, uso de ficheros, recursión En esta práctica haremos uso de clases para representar la red de metro de Madrid. Actualmente, la red de metro de Madrid se compone de 302 estaciones, 12 líneas convencionales y el ramal que une Ópera y Príncipe Pío. Además, existen tres líneas de metro ligero que suman un total de 27,78 km y cuentan con 38 estaciones.

De las 302 estaciones actuales, por 201 pasa una línea, en 27 transbordan dos líneas, en 10 tienen parada tres líneas y en la estación de Avenida de América coinciden cuatro líneas.

2 Parte 1

2.1 Clase Line

Escribe una clase `Line` para representar las líneas de metro de Madrid. Una línea de metro viene definida por una lista de nombres de estaciones y los tiempos del trayecto entre estaciones consecutivas. La creación de objetos de la clase `Line` inicializa la información de dichos elementos.

Esta clase debe definir al menos los métodos siguientes:

- `contains_station(e)`: devuelve el valor `True` si la línea contiene una estación de nombre `e` y `False` en caso contrario.
- `previous_e(e)` y `next_e(e)`: devuelven la estación anterior y la estación siguiente a la dada, respectivamente. Si la estación `e` no pertenece a la línea, entonces el método genera una excepción de la clase `LineException` con el mensaje adecuado. Si la estación `e` es cabecera de línea, entonces `previous_e()` genera una excepción de la clase `LineException` con el mensaje adecuado. Análogamente, si la estación `e` es la última de la línea, entonces `next_e()` genera una excepción de la clase `LineException` con el mensaje adecuado.
- `cost_origin2destination(start, finish)`: calcula la cantidad de segundos que se tarda en recorrer el trayecto en la línea desde la estación `start` hasta la estación `finish`. Si alguna de las estaciones `start` o `finish`no pertenece a la línea, entonces el método genera una excepción de la clase `LineException` con el mensaje adecuado.

A continuación se muestra la representación textual de los objetos de la clase `Line`:

```
>>> line1 = Line(['Bilbao','Tribunal','Gran Vía','Sol'], [130, 200, 150])
>>> print(line1)

Bilbao --> Tribunal --> Gran Vía --> Sol
```

Nota: Para simplificar la implementación de las clases, no consideramos la circularidad de algunas líneas de metro.

3 Parte 2

3.1 Clase Metro

Escribe una clase `Metro` para representar la red de metro de Madrid. Los objetos de la clase `Metro` almacenan la información de las líneas que forman la red junto con los transbordos entre líneas.

- La red de líneas de metro se puede representar mediante un diccionario cuyos elementos son de la forma:

```
line_name : line_object
```

donde `line_name` es el nombre de una línea dentro de la red y `line_object` es un objeto de la clase `Line`.

- La información sobre los transbordos se puede representar mediante un diccionario cuyas claves son todas las estaciones de la red y los valores asociados son las listas de los nombres de línea en que aparece cada estación.

La creación de objetos de la clase `Metro` inicializa la información a una red vacía de líneas y un conjunto vacío de transbordos. Esta clase debe definir al menos los métodos siguientes:

- `add_line(line_name, line)`: se encarga de añadir a la red de metro la línea `line` con el nombre `line_name`. También debe añadir los transbordos existentes entre líneas que comparten estaciones. Para añadir los transbordos debes usar el método `add_connections()` que se describe a continuación. Si en la red ya existe una línea con el nombre `line_name`, entonces el método `add_line()` genera una excepción de la clase `MetroException` con el mensaje apropiado.

- `add_connections(line_name, line)`: se encarga de actualizar la información de los transbordos a partir de las estaciones de la línea indicada, que ha sido previamente añadida a la red de metro por el método `add_line()`.

- `load_metro(file_name)`: es un método estático que recibe el nombre `file_name` de un fichero con los datos de la red de metro y devuelve un objeto de la clase `Metro`. Cada línea de metro se describe en el fichero con dos líneas de texto: la primera es el nombre (`Línea 3: Villaverde Alto - Moncloa`) y la segunda especifica las estaciones y los tiempos en segundos entre estaciones. A continuación se muestra un fragmento con la información que aparece en el fichero:

```
...
Línea 3: Villaverde Alto - Moncloa
Villaverde Alto->87->San Cristobal->8-> ... ->99->Argüelles->120->Moncloa
Línea 4: Argüelles - Pinar de Chamartín
Argüelles->39->San Bernardo->46->Bilbao->14-> ... ->77->Pinar de Chamartín
Línea 5: Alameda de Osuna - Casa de Campo
Alameda de Osuna->37->El Capricho->88->Canillejas->... ->74->Casa de Campo
Línea 6: Circular
Laguna->64->Carpetana->21->Oporto-> ... ->Lucero->98->Laguna
...
```

- `get_connections(e, line_name)`: devuelve una lista con los nombres de las líneas de la red con las que podemos conectar haciendo transbordo desde la estación `e` de la línea cuyo nombre es `line_name`. Si la línea `line_name` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Análogamente, si la estación `e` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. A continuación se muestra un ejemplo de su ejecución:

```
>>> print(m.give_connections('Bilbao','Línea 1: Pinar de Chamartín - Valdecarros'))
>>> print(m.give_connections('Sol','Línea 1: Pinar de Chamartín - Valdecarros'))
>>> print(m.give_connections('Iglesia','Línea 1: Pinar de Chamartín - Valdecarros'))

['Línea 4: Argüelles - Pinar de Chamartín']
['Línea 2: Las Rosas - Cuatro Caminos', 'Línea 3: Villaverde Alto - Moncloa']
[]
```

- `cost_origin2destination_transfer(start, finish)`: calcula la cantidad mínima de segundos que se tarda en realizar un trayecto que comienza en la estación `start` y termina en la estación `finish`. Si alguna de las estaciones `start` o `finish`no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. **Nota: El número máximo de transbordos permitido es uno.** Para simplificar el cálculo asumimos constante el tiempo empleado en todos los transbordos (300 segundos que incluyen el tiempo medio de espera al segundo tren). Si es imposible desplazarse desde una estación a otra haciendo como máximo un transbordo, el método `cost_origin2destination_transfer()` devuelve `None`.

4 Parte 3

Deseamos almacenar información sobre el estado de las estaciones. El estado de una estación en una línea puede ser abierto o cerrado. Inicialmente, cuando se crean los objetos de la clases `Line` y `Metro`, las estaciones están abiertas. Por otra parte, también queremos mantener información sobre el estado deas vías, ya que es posible que la circulación se interrumpa en un tramo de una línea.

Implementa en la clase `Metro` los métodos siguientes: * `close_station(line_name, e)`: se encarga de cerrar la estación een la línea de metro cuyo nombre es `line_name`. Si la línea `line_name` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Si la estación `e` no pertenece a la línea, entonces el

método genera una excepción de la clase `LineException` con el mensaje adecuado. Por último, si la estación `e` ya se encuentra cerrada, el método no tiene ningún efecto.

- `open_station(line_name, e)`: se encarga de abrir la estación een la línea de metro cuyo nombre es `line_name`. Si la línea `line_name` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Si la estación `e` no pertenece a la línea, entonces el método genera una excepción de la clase `LineException` con el mensaje adecuado. Por último, si la estación `e` ya se encuentra abierta, el método no tiene ningún efecto.

- `close_section(line_name, start, finish)`: se encarga de cerrar el tramo de la línea de metro cuyo nombre es `line_name` que comienza en la estación `start` y termina en la estación `finish`. Si la línea `line_name` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Si alguna de las estaciones `start` o `finish`no pertenece a la línea, entonces el método genera una excepción de la clase `LineException` con el mensaje adecuado. Por último, si la línea ya tiene otro tramo cortado, el método no tiene ningún efecto. **Nota:** El cierre de un tramo supondrá el cierre de las estaciones interiores al mismo, pero no el las correspondientes a su extremo. Sin embargo, esas estaciones cerradas no pueden ser reabiertas más que reabriendo el tramo.

- `open_section(line_name)`: se encarga de abrir el tramo cerrado de la línea de metro cuyo nombre es `line_name`. Si la línea `line_name` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Si la línea no tiene ningún tramo cortado, el método no tiene ningún efecto.

Esta implementación implica modificar las clases `Line` y `Metro` tanto para almacenar el estado de las estaciones y los tramos, como para incluir su estado en el cálculo del tiempo que se emplea en realizar un trayecto. Por ejemplo, no es posible realizar un viaje en una línea entre dos de sus estaciones si existe un tramo cerrado en dicha línea entre ambas estaciones. De forma similar, es conveniente que el estado de las estaciones y los tramos aparezca cuando se muestra la información de la red de metro. A continuación tienes la información que obtiene un usuario cualquiera acerca de la línea 3 de metro, en la que se ha cerrado el tramo entre Villaverde Alto y San Cristóbal.

```
Línea 3: Villaverde Alto - Moncloa
Villaverde Alto (CORTE) -/- (CORTE) San Cristobal  --> ... --> Argüelles --> Moncloa
```

5 Parte Opcional

Implementa en la clase `Metro` el método siguiente:

- `cost_origin2destination_transferN(start, finish, n)`: calcula la cantidad mínima de segundos que se tarda en realizar un trayecto que comienza en la estación `start` y termina en la estación `finish` efectuando una cantidad máxima `n` de transbordos. Si alguna de las estaciones `start` o `finish`no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Si es imposible desplazarse desde una estación a otra haciendo como máximo `n` transbordos, el método `cost_origin2destination_transferN()` devuelve `None`, lo que incluye el caso de trayectos imposibles por el cierre de estaciones y/o tramos.

[ ]: