Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI2693 - Laboratorio de Algoritmos y Estructuras III
Trimestre Abril-Julio 2022

Proyecto 2

1. Descripción de la actividad

Este problema *está basado en un problema de la ICPC* [1]. Tom es aficionado al béisbol y es fanático del equipo de la ciudad *CaracasSur*, llamado *Los Leones*. El equipo de *Los Leones* juega esta semana en la ciudad y Tom quiere ir al estadio de béisbol a ver el juego. El objetivo de este proyecto es ayudar a Tom a encontrar *la ruta por la cual pueda llegar desde su casa al estadio en el menor tiempo posible*. Las calles de *CaracasSur* están conectadas por intersecciones. Estos caminos son bidireccionales, por lo tanto es posible recorrerlos en dirección de ida y vuelta. Entre cada par de intersección solo hay un calle. Tom vive en una casa en una intersección y el estadio se encuentra en otra intersección. En la ciudad de *CaracasSur* neva en estos días. Recorrer una calle i se realiza en un tiempo t_i cuando la calle esta libre de nieve. Si la calle tiene algo de nieve, entonces el tiempo de recorrer la calle aumenta dependiendo de la cantidad de nieve que tenga. Cada calle está identificada por un número entero, comenzando desde uno.

Una vez que comienza a nevar, la alcaldía de *CaracasSur* aplica una programación para la limpieza de algunas de las i calles. El tiempo de la programación *tiene como referencia el inicio de la nevada*. Se tiene que la limpieza de una calle comienza en un tiempo t_i y finaliza en un tiempo t_f y se cumple que $t_i < t_f$, donde los tiempos son números enteros no negativos. Durante la limpieza *no puede haber ningún carro en la calle*. Una calle puede ser programada para limpiar varias veces. Ninguno de los tiempos de limpieza se interfieren o solapan. Por ejemplo, podemos decir que la calle 2 está programada para ser limpiada una primera vez comenzando la limpieza en $t_i = 5$ y finalizando en $t_f = 7$. La segunda vez la limpieza comienza en $t_i = 10$ y termina en $t_f = 14$. Un tiempo cualquiera como por ejemplo $t_i = 3$, hace referencia al tiempo 3 que comenzó la nevada. Cuando comienza la limpieza de una calle, los carros que quieren cruzar la calle deben esperar a que se termine de limpiar para cruzarla. Observe que al planificar la ruta, debe tomar en cuenta que un vehículo no puede atravesar una calle mientras se esté limpiando en el intervalo de tiempo $[t_i, t_f)$.

Cada una de las intersecciones está identificada con un número entero comenzando en uno. Cada una de las calles está identificada con un número entero a partir uno. Cada una de las calles i, tiene un tiempo t_i que indica el tiempo que se tarda un vehículo en atravesarla cuando no hay nieve en él. Cuando comienza a nevar, el tiempo de cruzar una calle i es mayor y viene dado por la función:

$$\text{mín}((\lceil (1 + T/100) * t_i \rceil), 100500 * t_i)$$

Donde t_i es tiempo que tarda en atravesar una calle cuando no está nevando, y T es el tiempo que lleva recibiendo nieve la calle i desde la última vez que fue limpiada, y en caso

de no haber sido nunca limpiada, entonces T es el tiempo que la calle lleva recibiendo nieve desde el inicio de la nevada.

Dado un mapa con las intersecciones y calles de *CaracasSur*, dado el cronograma de limpieza de las calles, dada la intersección en donde esta la casa de Tom, y dada la intersección donde está el estadio, se quiere obtener la ruta desde la casa de Tom que permita llegar al estadio en el menor tiempo posible. Se asume que siempre existe una ruta desde la casa de Tom hasta el estadio. Se tiene que la *nevada comienza en el instante que Tom sale de su casa.*

Para la solución del problema debe implementar un programa cliente llamada `RutaEstadio.kt`. Este programa debe ejecutarse con un *shell script*, llamado `runRutaEstadio.sh`. La línea de comando que la ejecuta es la siguiente:

```
>./runRutaEstadio.sh <archivoEntrada>
```

Donde `archivoEntrada` es el nombre del archivo con los datos de entrada. El formato del archivo de entrada es como sigue. La primera línea contiene dos números, el primero es el número n de intersecciones y el segundo el el número m de calles. Luego vienen m líneas con la información de las calles. Cada una de estas líneas contienen tres números enteros separados por un espacio en blanco. La posición i en la que aparece la calle, va a ser el identificador de la calle. Los dos primeros números de la calle i corresponden a las intersecciones que conectan a la calle. El tercer número corresponde al tiempo t_i que un vehículo tarda en atravesar la calle cuando está sin nieve. Luego viene un número entero k, que indica el número de limpiezas de calles que se van hacer en *CaracasSur*, al comenzar la nevada. Las siguientes k líneas tienen tres números enteros separados un espacio en blanco. El primer número corresponde al identificador de la calle, el segundo número es tiempo en el que comienza la limpieza y el tercer número es el tiempo en el que termina la limpieza de la calle. Finalmente viene una línea con dos números enteros separados por un espacio en blanco, en donde el primer número es la intersección donde se encuentra la casa de Tom, y el segundo el de la intersección donde se encuentra el estadio.

La salida esperada tiene dos líneas. En la primera línea tiene la secuencia, separada por un espacio en blanco, de las intersecciones de la ruta de menor tiempo desde la casa Tom hasta el estadio. La segunda línea tiene indica el tiempo mínimo dado por la ruta anterior.

1.1. Ejemplo 1 de entrada y salida

Se tiene como entrada un archivo llamado `prueba1.txt`, cuyo contenido se muestra a continuación:

```
4 3
1 2 10
2 3 10
3 4 10
1
2 10 15
1 4
```

Se tiene que se debe ejecutar el comando:

```
>./runRutaEstadio.sh prueba1.txt
```

La salida correcta es la siguiente:

```
1 2 3 4
38
```

1.2. Ejemplo 2 de entrada y salida

Suponga que se tiene como entrada un archivo llamado `prueba2.txt`, con el siguiente contenido:

```
6 7
1 2 4
1 5 3
2 3 5
2 4 2
3 6 1
4 6 2
5 4 5
9
3 15 17
3 3 5
5 11 12
5 6 8
5 1 3
7 12 14
7 2 4
6 1 2
6 8 9
1 6
```

Si se ejecuta el comando:

```
>./runRutaEstadio.sh prueba2.txt
```

La salida esperada es la siguiente:

```
1 2 4 6
10
```

2. Detalles de la implementación

La idea es que para implementar la solución de las actividad, debe modelar el problema como un grafo y utilizar los algoritmos sobre grafo vistos en clase, para su solución. El cliente

debe hacer uso de la librería *libGrafoKt*. La librería puede modificarse y/o ampliarse para poder resolver el problema planteados. Puede hacer uso de las clases de la librería de Kotlin para su implementación. Todo el código debe estar debidamente documentado. Debe realizar un archivo, en formato Markdown, llamado `Readme.md` que debe contener una explicación detallada de como la actividad planteada fue resuelta. Debe indicar las estructuras de datos y los algoritmos usados para su solución. Sin esta explicación el proyecto puede ser invalidado. La documentación de las operaciones implementadas debe incluir *descripción*, *parámetros* y *tiempo de la operación*. El tiempo de las operaciones debe ser dado en número de lados y/o número de vértices, cuando eso sea posible. Sus implementaciones deber ser razonablemente eficientes.

Debe crear una carpeta llamada *proyecto2_X_Y*, donde donde X y Y son los carné son los integrantes del equipo. Dentro de este directorio debe estar el código de la solución de las actividad. La entrega debe contener el programa cliente, el archivo `Makefile`, el *shell script* ejecutable, la librería *libGrafoKt*, y el archivo `Readme.md`.

Si su entrega no puede compilarse o ejecutarse tiene cero en la nota del proyecto. La plataforma en la que debe ejecutarse el proyecto es GNU/Linux.

3. Condiciones de entrega

Los códigos del proyecto y la declaración de autenticidad debidamente firmada, deben estar contenidos en un archivo comprimido, con formato *tar.xz*, llamado *proyecto2_X_Y.tar.xz*, donde X y Y son los número de carné de los estudiantes. La entrega del archivo *proyecto2_X_Y.tar.xz*, debe hacerse por medio de la plataforma *Classroom* antes de las 11:59 P.M. del día martes 02 de agosto de 2022.

Referencias

[1] International Collegiate Programming Contest, "Past problems," 2022, [Online; accessed 24-Junio-2021]. [Online]. Available: https://icpc.global/worldfinals/problems

Guillermo Palma / gvpalma@usb.ve / Julio de 2022