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

Proyecto 1: Grafo

Autores:
Br. Álvaro Ávila - 15-10103.
Br. Luis Paredes - 11-10740.

Noviembre, 2018

Ejecución

```
# Compilar desde el directorio src/
cd src/
javac *.java

# Ejecutar el cliente (desde src/)
java ClienteGrafo               # Para crear un nuevo grafo
java ClienteGrafo <grafo.txt>   # Para importar grafo
```

Detalles de implementación

Para la estructura del directorio que contiene al proyecto se optó por usar una disposición simple donde todas las clases se encuentran en el directorio *src/*.

Las clases incluidas son las especificadas en el enunciado del proyecto, con la adición de la clase *Utilidades* que, como su nombre lo indica, contiene métodos de uso común que son consumidos por las clases principales.

Adicionalmente a esto se agregaron 3 archivos *.txt* para realizar pruebas de forma manual —lo ideal hubiera sido usar JUnit pero por limitaciones de tiempo no se escribieron pruebas de unidad.

```
.
├── Proyecto 1.pdf
├── Informe Proyecto 1 Algoritmos 3.pdf
├── README.md
└── src
    ├── Vertice.java
    ├── Lado.java
    ├── Arco.java
    ├── Arista.java
    ├── ClienteGrafo.java
    ├── Grafo.java
    ├── GrafoDirigido.java
    ├── GrafoNoDirigido.java
    ├── Utilidades.java
```

```
    ├──── GrafoDirigido.txt
    ├──── GrafoNoDirigido.txt
    └──── GrafoError.txt
```

Vertice, *Arco* y *Arista*

Se siguió cabalmente la especificación dada para estos 3 TADs. Asimismo, se introdujeron los atributos necesarios para almacenar los datos de cada instancia.

GrafoDirigido y *GrafoNoDirigido*

Para los métodos de estas clases se siguió la especificación planteada en el enunciado del proyecto, mientras que para representar los atributos de cada grafo se optó por usar la siguiente especificación:

- ➔ ***vertexCount***: entero, representa el número de vértices.
- ➔ ***edgeCount***: entero, representa el número de lados (aristas o arcos).
- ➔ ***vertices***: *LinkedHashMap* (diccionario) con claves de tipo cadena de texto y valores de tipo *Vertice*.
- ➔ ***edges***: *LinkedHashMap* (diccionario) con claves de tipo cadena de texto y valores de tipo *Vertice*.

La escogencia del tipo diccionario para la representación de los vértices (y lados) del grafo fue natural ya que es conveniente poder acceder a la información de cualquier vértice (lado) a partir de consultar una clave —el *id* del vértice (lado)— en tiempo constante.

La escogencia de *LinkedHashMap* en vez de otros tipos de diccionarios vino dada por ser la opción que mejor combina versatilidad y eficiencia.[1]

Utilidades

Contiene dos métodos que cumplen la función de verificar que el archivo introducido exista y tenga el formato correcto.

Se decidió colocar estos métodos en una clase aparte puesto que son consumidos tanto por la clase *ClienteGrafo*, como por las clases *GrafoDirigido* y *GrafoNoDirigido*.

ClienteGrafo

Esta clase contiene a la rutina principal de la aplicación y brinda acceso a las funcionalidades de las demás clases.

La lógica principal consiste en mostrar una cónsola que se encarga de procesar los comandos luego de haber creado o importado un grafo al inicio de la ejecución.

Un aspecto interesante de la cónsola es que permite al usuario ingresar la instrucción en conjunto con los argumentos correspondientes en una sola llamada.

Esto es posible gracias a que se implementó un *parseador* que se encarga de analizar el comando introducido y escoger automáticamente el método que se debe ejecutar (o informar al usuario que hay un error de sintaxis).

La idea de usar un parseador surgió como solución para implementar el cliente con la menor cantidad de código posible. Sin embargo, es de notar que fue necesario agregar condicionales extra, debido a la necesidad de identificar algunos tipos de datos y la limitación de no poder usar las Generics en la corrida, debido a la eliminación de tipos parametrizados que aplica Java luego de compilar.[2]

Referencias

1. https://stackoverflow.com/questions/40471/differences-between-hashmap-and-hashtable/42622789#42622789
2. https://docs.oracle.com/javase/tutorial/java/generics/erasure.html