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

Representación de grafos

1. Descripción de la actividad

El objetivo de este laboratorio es el de la implementación de la representación de grafos como lista de adyacencias. Para ello se va a realizar una librería de grafos, que llamaremos *libGrafoKt*. La idea de la librería es la de contener representaciones de grafos y algoritmos sobre grafos. En esta semana se van agregar las representaciones sobre grafos. A medida que se avance en el curso se agregarán algoritmos sobre grafos.

La librería tiene una clase abstracta `Lado`, de la cual van a derivar las clases `Arista` y `Arco`. La clases `AristaCosto` y `ArcoCosto` son subclases que tienen como característica adicional que contienen el costo del lado. La Figura 1 muestra la jerarquía de clases de los diferentes tipos de lados que puede contener un grafo.

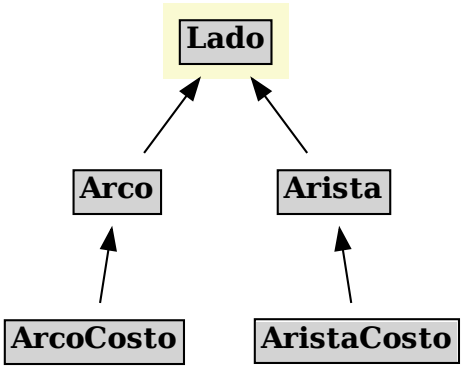

Figura 1: Jerarquía de las clases que representan a los lados en un Grafo. El amarillo claro indica que la clase `Lado` es una clase abstracta.

Se crea una interfaz `Grafo`, de la cual se derivan las clases `GrafoDirigido`, `GrafoNoDirigido`, `GrafoDirigidoCosto` y `GrafoNoDirigidoCosto`. La idea es que con una interfaz `Grafo`, podemos usar un objeto de tipo grafo en los algoritmos en los cuales el grafo de entrada puede un grafo no dirigido o digrafo. La Figura 2 muestra la jerarquía de las clases que representan grafos.

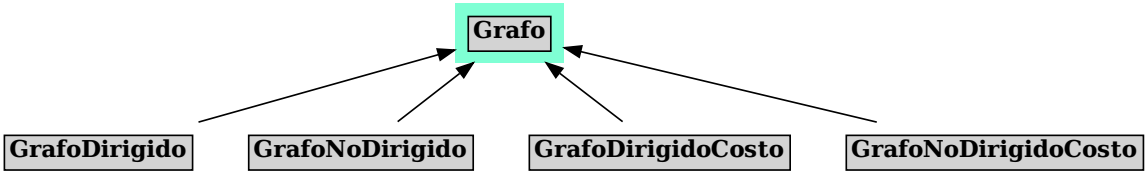

Figura 2: Jerarquía de las clases que representan a los Grafo. La clase `Grafo` es una interfaz.

Se le proporcionará un código base de *libGrafoKt*, el cual debe completar. A continuación se describirá los archivos que componen el código base de *libGrafoKt*:

Lado.kt : Clase abstracta de implementa los lados de un grafo.

Arco.kt : Clase que contiene la implementación de los lados de un digrafo.

Arista.kt : Clase que contiene la implementación de los lados de un grafo no dirigido.

ArcoCosto.kt : Clase que contiene la implementación de los lados de un digrafo con costo.

AristaCosto.kt : Clase que contiene la implementación de los lados de un grafo no dirigido con costo.

Grafo.kt : Interfaz que contiene la implementación de un grafo.

GrafoDirigido.kt : Clase que contiene la implementación de un digrafo como lista de adyacencia.

GrafoNoDirigido.kt : Clase que contiene la implementación de un grafo no dirigido como lista de adyacencia.

GrafoDirigidoCosto.kt : Clase que contiene la implementación de un digrafo como lista de adyacencia usando la clase ArcoCosto.

GrafoNoDirigidoCosto.kt : Clase que contiene la implementación de un grafo no dirigido como lista de adyacencia usando la clase AristaCosto.

Makefile : Archivo para la compilación del código fuente de *libGrafoKt*.

Se quiere crear un programa cliente que pruebe la librería *libGrafoKt*. Por ello, la librería y el programa cliente estarán contenidos en la carpeta *pruebaEstructuraGrafo*. A continuación se describirá los archivos que están contenidos en la carpeta *pruebaEstructuraGrafo*:

Main.kt : Este archivo contiene el programa cliente que demuestre el correcto funcionamiento de la librería *libGrafoKt*.

grafoEjemplos : Carpeta con dos grafos de prueba. A continuación se describen los dos grafos:

- **grafoCosto.txt** : Archivo con el formato de los datos de un grafo en donde se indican los costos de los lados. Puede interpretarse como un grafo no dirigido o digrafo.
- **grafoSinCosto.txt** : Archivo con el formato de los datos de un grafo sin costo en los lados. Puede interpretarse como un grafo no dirigido o digrafo.

Makefile : Archivo para la compilación del código fuente del programa cliente y de la librería *libGrafoKt*.

pruebaGrafo.sh : Archivo *script* ejecutable que llama de manera correcta al programa cliente.

El *script* `pruebaGrafo.sh` se ejecuta mediante la siguiente línea de comandos:

```
>./pruebaGrafo.sh -g[d|n|c|p] archivo_con_grafo
```

Donde:

`archivo_con_grafo:` archivo con los datos de un grafo en el formato correcto.

`-g:` Indica el tipo de grafo que es `archivo_con_grafo`. Las cuatro opciones posibles:

- `d:` grafo dirigido.
- `n:` grafo no dirigido.
- `c:` grafo dirigido con costos.
- `p:` grafo no dirigido con costos.

Una llamada de línea de comando del programa que es válida, sería:

```
>./pruebaGrafo -gc grafoEjemplos/grafoCosto.txt
```

El formato del archivo que contiene la descripción de un grafo, con o sin costo en los lados, se indica en el constructor de las clases: GrafoDirigido.kt, GrafoNoDirigido.kt, GrafoDirigidoCosto.kt y GrafoNoDirigidoCosto.kt

2. Detalles de la implementación

La librería *libGrafoKt* debe ser implementada como un paquete de Kotlin. El paquete se llama `ve.usb.libGrafo` por lo que todo el código de la librería debe estar contenido en el directorio *ve/usb/libGrafo*. Al compilar la librería, se crea un archivo *jar* llamado libGrafoKt.jar. Este archivo contiene a la librería y es el que debe ser importado por los programas clientes que quieran usar la librería.

El programa cliente Main.kt, ubicado en *pruebaEstructuraGrafo*, debe contener el código que implementa pruebas a algunas de las funciones de *libGrafoKt* y funciona ejecutando la línea de comandos presentada anteriormente. El programa recibe la línea de comando con que es llamado y debe procesarla. En caso de que haya algún error en la llamada del programa, se debe indicar el mismo al usuario.

Puede hacer uso de las clases de la librería de Kotlin para su implementación. Cada una de las operaciones en las clases tiene una breve descripción la misma. Esa descripción debe borrada de su código de entrega. En su lugar deben colocar para cada de las operaciones una documentación que incluya: *descripción*, *parámetros*, *precondiciones*, *postcondiciones* 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. La implementación debe seguir la guía de estilo de Kotlin indicada en clase.

3. Condiciones de entrega

Los códigos del laboratorio y la declaración de autenticidad debidamente firmada, deben estar contenidos en un archivo comprimido, con formato *tar.xz*, llamado *LabSem1_X_Y.tar.xz*, donde X y Y son el número de carné de los estudiantes. La entrega del archivo *LabSem1_X_Y.tar.xz*, debe hacerse por medio de la plataforma *Classroom* antes de las 12:00 pm del día viernes 03 de junio de 2022.