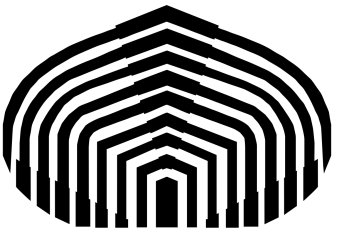

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Laboratorio de Algoritmos y Estructuras de Datos III (CI 2693)
Prof. Fernando Torre

Proyecto de Laboratorio 1: TADs Grafo, Grafo No Dirigido y Grafo Dirigido

Elaborado por:
Manuel Faria. 15-10463
Juan Oropeza. 15- 11041

Sartenejas, octubre de 2018

Cómo correr el programa

Para la ejecución del programa se creó un cliente para la prueba de las funcionalidades del programa. Para ejecutar dicho cliente, primero deberá compilarlo. Para ello, ejecute en el directorio donde se encuentran los archivos el siguiente comando:

$$\$\ javac\ ClientProgram.java$$

Una vez compilado, existen dos formas de correr el programa:

1. **Comenzar con un grafo vacío:** En este caso, deberá especificar mediante el cliente, el tipo de grafo (Dirigido o No Dirigido) y los tipos de datos que se almacenarán en los vértices y lados (Booleanos, Strings o Doubles). Para comenzar con esta funcionalidad, ejecute:

   $$\$\ java\ ClientProgram$$

2. **Comenzar cargando un grafo desde un archivo:** Para esta funcionalidad, deberá tener en el mismo directorio un archivo con el formato previamente especificado en el enunciado del proyecto. Ya con esto, ejecute el comando:

   $$\$\ java\ ClientProgram < nombreDeArchivo >$$

   donde $< nombreDeArchivo >$ representa la ruta hacia el archivo que se desea cargar. El programa mostrará una lista con las opciones que se pueden ejecutar sobre dicho grafo.

Detalles de implementación

El esquema de clases usado en el proyecto fue el planteado inicialmente en el enunciado del proyecto. Sin embargo, a este esquema se le agregaron una interfaz y tres clases para la conversión de los datos almacenados en vértices y lados.

El grafo se almacena como se indicó en el enunciado del proyecto, con la representación de lista de adyacencias. A esto se le añadió una lista de lados, para poder almacenar los datos de dichos lados. Cada lado posee dos atributos de tipo string que almacena el identificador de cada uno de los vértices sobre los cuales el lado incide.

Como ya fue mencionado, debido a la problemática que se presentó al momento de generar un cliente que pudiera manipular distintos tipos de datos sobre los vértices y lados, se decidió generar una interfaz y tres clases adicionales. Dicha interfaz solo posee un método llamado transform, cuya finalidad es transformar el input de usuario o línea de archivo, ambos de tipo string, al tipo de dato que haya sido seleccionado.

Las 3 clases generadas, corresponden a los transformadores de los tres tipos de datos que son admitidos: Booleano, String y Double. Cada una de ellas implementan la interfaz previamente analizada, y sus respectivos métodos transform se encargan de convertir los inputs en el tipo de dato especificado. En el caso particular del transformador Double, se lanza una excepción de tipo NumberFormatException si el input no puede ser transformado a Double. Dicha excepción es manejada por el método *loadGraph* cuando se está cargando un grafo desde un archivo y por el método *dataInput* cuando se está utilizando mediante la implementación de un grafo vacío.

Es importante mencionar que gracias al uso de la lista de lados la complejidad de muchos de los métodos que se implementaron disminuyó significativamente. Se puede notar que el uso de estructuras de almacenamiento generalmente es sumamente útil.

Del mismo modo, la modularización pertinente del código en la clase $ClientProgram$ redujo en una cantidad sustancial el número de líneas de la implementación de este proyecto.

Para la implementación de los métodos *vertices, lados, incidentes, sucesores y predecesores* no se utilizaron las funciones *toString* por cuestión de comodidad

visual. Nótese que todas estas funciones, devuelven una lista de vértices o lados, y cuando existe una gran cantidad, es prácticamente imposible visualizarlos todos por pantalla. Es por ello, que se realizó una implementación en formato de tabla que facilita la visualización de los datos de vértices o lados según corresponda. Si se desea visualizar la ejecución de las funciones *toString* de los objetos de tipo vértice o de tipo lado (arco o arista), se debe proceder a la ejecución de las funciones *obtenerVertice, obtenerArco* o *obtenerArista*. En dichas funciones si fue utilizada la función *toString* de cada una de los respectivos tipos de datos puesto que se asegura que el resultado es único, y por ende, es legible en pantalla.

Como casos de prueba se utilizaron los casos de prueba de la carpeta *graphs* resultando exitosos en un 100%. Los archivos de dicha carpeta se encuentran guardados con el siguiente formato de nombre $graphVEGv-e.txt$ donde, V representa el tipo de dato de los vértices, E el tipo de dato de los lados, G el tipo de grafo, v el número de vértices y e el número de lados.