Universidad Simón Bolívar.
Laboratorio de Algoritmos y Estructuras III - CI2693.
Yerimar Manzo: 14-10611.
Jonathan Bautista: 16-10109.

USB

Informe de Proyecto 1
(Implementación de un TAD grafo)

Decisiones de diseño:

Dada la naturaleza de los requisitos para el proyecto en cuestión, decidimos hacer uso de dos listas enlazadas principales. Una lista enlazada de lados (arcos o aristas, en caso de ser dirigido o no dirigido) y una lista enlazada de listas enlazadas para representar todo el grafo en forma de lista de adyacencias. Todo esto con el fin de aprovechar el potencial que poseen tales estructuras de datos, tales como:

- Su facilidad para agregar datos de manera rápida y eficiente.
- No poseen problemas para aumentar su tamaño.
- Su capacidad de representar el grafo de manera manejable y sencilla.
- Todas las funciones que ofrecen sus librerías.

En nuestra implementación, consideramos el primer elemento de cada una de las listas como su vértice representante, así como consideramos al resto de los elementos de la lista como sus vértices adyacentes. En el caso particular del grafo dirigido, estos serían específicamente vértices de llegada, tomando al representante como vértice inicial del arco respectivo. Esto no sucede en el caso del grafo no dirigido, en donde es irrelevante la orientación de las aristas.

La relación entre la lista de arcos y de vértices es especialmente considerada en los procesos de eliminación o adición de elementos al grafo, siempre tomando en cuenta las características particulares de cada tipo (dirigido o no dirigido).

Cada uno de los vértices de las listas poseen sus atributos respectivos, que vienen dados por su propia clase, así como los arcos y aristas, quienes heredan de la clase lado. Para facilidad de comprensión al lector, se denominan a los vértices extremos de una arista como extremo1 o extremo2. Sin embargo, esto no tiene ninguna influencia en la orientación del grafo. Caso contrario de los grafos dirigidos, como explicamos anteriormente, donde importa la identidad del grafo inicial y el grafo final de cada arco, para poder ser colocado en una lista específica. Por supuesto, además de este tipo de identificación, también poseen el resto de los atributos que los identifican como clases vértice y clases lados.

Todos los métodos requeridos fueron implementados e incluso, existen algunos otros que fueron agregados, siendo simples cambios de entrada en funciones ya existentes, pero que pueden expandir las posibilidades de entrada a una misma función que podría requerirlas. Estos métodos extras serán detallados más adelante.

Ejecución Cliente:

Para ejecutar el archivo Cliente.java es necesario ejecutar la línea de código:

java Cliente <nombre archivo txt>

En el cual el nombre del archivo a pasar como argumento es, valga la redundancia, el nombre del archivo que contenga el grafo deseado a cargarse utilizando la convención explicada en el enunciado del proyecto para describirlo. Es decir, la primera línea del archivo debe ser un carácter D o N el cual indica si el grafo es dirigido o no dirigido, la segunda y tercera línea son el número de vértices y lados respectivamente seguido de n líneas que describen toda la información requerida por los vértices y m líneas que tiene toda la información requerida por los lados.

En este caso es fundamental la primera línea ya que ella es la que decide a qué parte del cliente es redirigido el usuario a la hora de ejecutar el programa, ya que existen diferencias de funciones entre los grafos dirigidos y no dirigidos dados por la implementación y por la naturaleza de los mismos. Luego el archivo es cargado y dependiendo de si la información fue bien proveída el programa continua o emite una excepción informando al usuario de cualquier eventualidad que haya ocurrido a la hora de cargar el grafo, éstas excepciones son de distinta índole como que no puede conseguir el archivo proveído o algún dato está erróneo.

En caso de que el archivo haya sido cargado con éxito al usuario se le muestran diversas opciones las cuáles puede realizar, éstas son todas las funciones que posee el TAD del grafo y que dependen como ya dicho anteriormente del tipo del grafo, las mismas están representadas con una letra del alfabeto (en este orden)

y que el usuario debe ingresar seguido de la tecla Enter para ejecutar dicha función. Lo que viene a continuación dependerá de la opción que haya escogido el usuario, pero luego de que dicha acción haya sido realizada exitosamente se procede a mostrarle al usuario otra vez todas las opciones que posee hasta que él mismo decida salir del programa utilizando una opción proveída.

Éstas acciones serían: agregar vértice o lado, eliminar vértice o lado, ver adyacencias, mostrar el grado del grafo o de sus vértices, clonar el grafo actual y crear uno nuevo, imprimir el grafo para ver toda la información que posee el mismo, etc. Como fue mencionado con anterioridad dado a las diferencias entre Grafo Dirigido y No Dirigido fue importante realizar distintas implementaciones por lo tanto existen funciones que posee uno que no posee el otro, como por ejemplo en Grafo Dirigido existen funciones extras de sucesores, predecesores, existe vértice, obtener vértice ya que fue necesario para tener diversas formas de acceder a ésta información. Es por esto que la forma en la que es mostrada en pantalla la información de cada uno es distinta.

Casos de prueba:

Los casos de prueba utilizados para probar éste programa fueron en esencia cuatro (4), los cuales serían:

- Grafo Dirigido con información correcta: Fue construido un caso de prueba relativamente pequeño que tuviera toda la información del grafo correcta para probar la carga del grafo y verificar que en efecto toda la información está siendo almacenada de la forma correcta y no existan errores. De igual forma se usó de plantilla para probar las demás funciones y verificar su correctitud.

- Grafo Dirigido con información incorrecta: Fue necesario probar que si se proveen datos erróneos no se carguen de mala manera en el grafo y en efecto muestre al usuario las excepciones de los errores que están sucediendo. Al igual que el caso anterior el grafo es de un tamaño relativamente peq ueño.

- Grafo No Dirigido con información correcta: Al igual que el caso dirigido se creó un archivo de tamaño pequeño para probar la funcionalidad de los métodos implementados y verificar que no existan incoherencias a la hora de la carga de los datos.

- Grafo No Dirigido con información incorrecta: Para asegurarse que todo estaba en orden al igual que su contraparte se creó un archivo de prueba con datos incorrectos para corroborar las excepciones y carga de datos.

Es necesario mencionar que cuando se habla de datos incorrectos se refiere a que por ejemplo no se escriben líneas de información de ciertos vértices o el número del mismo no es igual al número de líneas, o en el id del vértice se pasa un string, etc.

Caso especial:

Pensamos que es necesario explicar un poco el caso del método “clonar”, el cual fue necesario realizar un implementación un poco más larga y pensada en general. Esto es debido a que lo que se quiere es retornar un grafo que posea la misma información que el grafo actual y de forma un poco ingenua se llegó a pensar en crear una nueva instancia de Grafo e igualarla al grafo actual, cosa que nos dimos cuenta es un error, ya que en Java todas las clases generadas por uno mismo, es decir las que no son primitivas del lenguajes, se copian por referencia por la tanto lo que se estaba logrando era generar una copia de la dirección de memoria a la que apunta el grafo actual, y de esta forma si se intenta cambiar uno de los dos se cambia la misma información.

Por esta razón, fue necesario crear poco a poco cada vértice y lado del nuevo grafo cuidando que la información que pasaramos fuera primitiva, para lograr pasarlo por valor y así tener un verdadero clon que trabaje por separado del grafo actual.