Estudiantes:

Jesus De Aguiar 15-10360

Wilfredo Graterol 15-10639

Curso: CI-2693

**Informe - Proyecto I: TAD Grafo**

**Diseño de la estructura:**

Para este proyecto era necesario implementar el TAD como una lista. En este caso decidimos utilizar las siguientes interfaces y clases abstractas:

* *Graph*
* *Node*
* *Transformer*

Al implementar la interfaz *Graph* creamos las clases *DirectedGraph* y *UndirectedGraph.* Al implementar la clase abstracta *Node,* creamos las clases *NodeD* y *NodeU,* las cuales seran los tipos de nodos que se utilizaran en lod digrafos y grafos no dirigidos respectivamente*.* Ademas para representar los lados se crearon las clases *Arc* y *Edge* para representar arcos y aristas respectivamente.La manera en la que se decidio guardar los nodos y los lados en los grafos fue la siguiente: En cada una de las clases *DirectedGraph* y *UndirectedGraph* se guarda un A*rrayList* en el que se guardara todos los nodos del grafo, un par de enteros que representan el numero de lados y nodos que hay en el grafo, un par de *HashMaps* para guardar los nombres de los nodos y los lados y un par de *Transformers* que permitiran parsear los datos especiales al momento de agregar nodos y lados al cargar el grafo de un archivo. Todos los nodos tienen como atributo un String que representa su id y un *Double* que representa su peso. Cada *DNode* guarda como atributo su grado interno y externo, un par de *HashSets* para sus sucesores y predecesores y un par de *ArraList* para los arcos que salen y los arcos que entran en el. Para el caso de los *UNode* es similar, con la diferencia de que solo se guarda un entero para el grado, un *ArraList* para los lados incidentes y un *HashSet* para los nodos adyacentes. Los lados solo guardan su nodo inicial y final, su dato y su peso.

En nuestra implementacion le damos prioridad a las operaciones de insercion, busqueda y query, pero pagando el precio de hacer la operacion de eliminacion mas costosa. Esta decision fue tomada dado que en nuestro caso, esperamos utilizar la operacion de eliminacion muy poco en comparacion con las demas. Al mantener todas estas estructuras alternativas podemos saber en O(1) si un nodo o lado esta o no en el grafo o si esta adyacente a otro.

**El cliente:**

Para correr el cliente es necesario descomprimir el archivo Proyecto1.zip en una carpeta y correr *javac \*.java* en ese directorio, una vez hecho eso debe correr en el terminal java *ClienteGrafo*. Al correr el comando se iniciara el menu en el cual tendra acceso a todas las operaciones basicas de grafos, solo debe escribir el numero de la opcion que desee y marcar *enter.* El menu se vuelve a mostrar al final de cada seleccion, a menos que el usuario seleccione la opcion de salir.Al inicio del main del cliente se inicializan dos grafos, uno dirigido y uno no dirigido, esto se hizo pues se generaban errores de compilacion inicializar el grafo como *Graph*, inicializarlo luego como digrafo y utilizar operaciones unicas de los digrafos. Sin embargo, en el cliente solo se permite mantener un grafo a la vez, al crearse uno, el anterior se deja de utilizar y solo se utiliza el nuevo para todas las operaciones. Si intenta usar operaciones unicas de digrafos en un grafo no dirigido (como calcular el grado interno de un nodo) , el programa lanzara una excepcion y finalizara su ejecucion. Cualquier entrada de un tipo de dato no valido al agregar un nodo o un lado tambien provocara que el programa lance una excepcion. Todas las operaciones del cliente se basan en las operaciones de las clases de los grafos. En particular las opciones para cargar un grafo desde un archivo, agregar un nodo y agregar un lado se apoyan de unos atributos que implementan la interfaz *Transformer*. Estos atributos se inicializan al leer el archivo o al crear el grafo y se encargan de parsear los datos especiales de los nodos y lados una vez definido este tipo.

**Casos de prueba usados:**

Para probar cargar un grafo desde un archivo probamos con dos archivos adjunto en el proyecto, uno es un archivo llamado “dirigido” el cual carga en el programa un digrafo y contiene lo siguiente:

D

B

D

7

10

cero 0.5 0

uno 1.5 1

dos 2.5 2

tres 3.5 3

cuatro 4.5 4

cinco 5.5 5

seis 6.5 6

ceroconuno false 0 cero uno

unocondos true 1 uno dos

cerocondos false 2 cero dos

dosconcero false 2 dos cero

tresconcuatro true 3 tres cuatro

cuatroconcinco true 3 cuatro cinco

seisconseis true 6 seis seis

cincocondos true 5 cinco dos

trescondos true 2 tres dos

ceroconcuatro false 0 cero cuatro

El otro archivo, llamado “nodirigido” es similar pero carga en el programa un grafo no dirigido. En las pruebas no se detecto ningun error con ninguno de los dos archivos, tanto al cargarlos como al realizar operaciones sobre ellos. Tambien se realizaron pruebas iniciando el grafo mediante las operaciones del menu y tampoco se detecto ningun error.