UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE COMPUTACIÓN
LABORATORIO DE ALGORITMOS III – CI2693

PROYECTO I

TAD GRAFO DIRIGIDO/NO DIRIGIDO

ELABORADO POR:

Pedro Rodríguez (15-11264)

Mariangela Rizzo (17-10538)

Decisiones de implementación

Implementación del grafo

Se hizo como lista de adyacencia; no obstante, en lugar de hacerse uso de una lista enlazada de listas enlazadas, se hizo uso de una lista enlazada de nodos, los cuales se presentan de la siguiente manera:

- *Estructura ALNode*

El nombre de ALNode hace referencia a Adjacency List Node. Esta estructura funciona como un nodo del grafo, almacenando datos importantes del mismo y siendo los elementos de la lista principal que refleja el grafo a implementar.

Guarda nodos adyacentes (predecesores y sucesores en caso del grafo dirigido), el identificador del nodo, su nombre, y otros elementos a los que se puede acceder con mayor facilidad haciendo uso de susodicha implementación.

Uso de Hash Set

Los HashSet son una estructura de datos basada en conjuntos de elementos (no ordenados y no duplicados) e implementada mediante tablas de Hash. En el proyecto se usaron para guardar tres cosas en específico: Ids de los vértices, nombres de los vértices e Ids de los lados.

La principal razón para el uso del HashSet es para aprovechar el hecho de que sus operaciones más básicas son de tiempo constante O(1) (add, remove, contains, size), así que operaciones de grafos que requerirían iterar, en el peor de los casos, sobre toda la estructura de lista de adyacencia para chequear si existe un vértice/lado con cierto identificador o nombre (equivalentemente con los lados) son reducidas a un tiempo constante.

La operación que más saca provecho a esto es verificar si existe un vértice/lado en el grafo con cierto identificador y cabe destacar que esta operación es usada en

muchas otras operaciones como agregar vértice/lado, eliminar vértice/lado, obtener vértice/lado.

La única desventaja puede ser que para grafos pequeños la diferencia no va a ser tanta y va a ocupar más memoria que iterar constantemente, pero para grafos grandes este consumo de memoria, a pesar de ser elevado, es justificado ya que evita tener que iterar, por ejemplo, un millón de elementos

Funcionamiento del cliente

El menú se realizó de manera que sea intuitivo para el usuario sin instrucciones previas. Se maneja por medio de números, los cuales seleccionan una de las opciones disponibles, permitiendo de esa manera cargar grafos de archivos, crearlos desde cero u observar los ya creados (como se aprecia en el menú principal).

A medida que se elige una de las alternativas disponibles, aparecen en el terminal distintos menús de opciones referidas a lo escogido. De esa manera el usuario puede solicitar lo que necesita. Hay una opción que permanece constante para regresarse al menú anterior y, una vez que se llega al menú principal, para dejar de ejecutar el programa; de esta manera se puede salir cuando se considere necesario.

En caso de cometer un error introduciendo un número o solicitando alguna de las opciones, el mismo Cliente se encarga de notificarlo y pedirle al usuario que lo corrija para poder seguir con la instrucción o que escoja otra de las que se ofrecen.

Métodos extras

En adición a los métodos y las funciones solicitadas en el enunciado del proyecto, por motivos de facilitar la realización de algunos de ellos, se agregaron funciones y métodos extra, de manera que fuesen auxiliares.

Por ejemplo, existeArista es una función que retorna un booleano e indica si hay una arista entre dos vértices particulares, cuyos identificadores numéricos son pasados como parámetros. Esta función ayudará a la eliminación de la arista, ya que permite ubicarla con mayor certeza y, en caso de un multigrafo, brinda la seguridad de que no se elimina otra arista existente entre esos dos vértices.