Facultad de Ciencias Exactas

Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas

Trabajo Práctico Especial

Segunda Etapa

"Buscador por Géneros Implementación con Java"

Meliendrez Agustín
Santos Luciano

Tandil, Junio 2018

1. Introducción

A partir de la cátedra "Programación 3", perteneciente la carrera "Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas" (TUDAI), de la Facultad de Ciencias Exactas, perteneciente a la Universidad Nacional del Centro de la Provincia de Buenos Aires, se planteó la problemática de implementar, en Java, nuevas funcionalidades a la herramienta desarrollada para una primer entrega. Dichas funcionalidades se enfocan en la realización de un "Buscador" de libros en base a una serie de "Géneros" buscador por un usuario.

Para ello se asume, que las funcionalidades anteriores ya poseen un correcto funcionamiento, por lo que no se implemento a lo largo del próximo informe. Por su parte, se presentarán las discusiones realizadas para la búsqueda de las mejores implementaciones posibles.

Una vez planteada la discusión de las diferentes opciones disponibles, se presentará cuál fue la opción seleccionada para llevar a cabo el desarrollo del programa, como así también el por qué de su selección.

En tercer lugar, se presentarán los ensayos de pruebas realizadas, a través de los resultados obtenidos, para demostrar si la implementación seleccionada era la correcta, o si se encontraron errores en su desarrollo.

Por último, se presentarán las conclusiones alcanzadas a través del desarrollo del Trabajo Práctico Especial, como así también dificultades o limitaciones encontradas en el desarrollo del mismo.

2. Desarrollo

2.1 Consideraciones e Implementación

Antes de comenzar con la implementación del código, para resolver el problema planteado, se esquematizó cómo debía funcionar el programa, teniendo en cuenta a las consignas planteadas por la cátedra. A partir de ello, se determinaron, en principio, tres clases que debían existir, en base al tipo de grafo que se utilizaría.

Es por ello, que en primer lugar se determino la elección del grafo en base a una serie de criterios:

- **Dirigido:** Corresponde a la dirección de las aristas desde un vértice a otro. En este caso solo posee una dirección de origen y otra de destino.
- **No Dirigido:** Corresponde a la dirección de las aristas desde un vértice a otro. En este caso no hay dirección definida, sino que la arista se comunica con los vértices de igual manera.

- **Ponderado:** Constituye el peso que va a poseer la arista entre dos vértices.
- **No ponderado:** Corresponde a un grafo en el cual las aristas no tienen un peso o valor asociado.
- **Lista de Adyacencias:** Corresponde a un grafo implementado a través de listas.
- **Matriz de Adyacencias:** Corresponde a un grafo implementado a través de una matriz.

En base a las consideraciones planteadas, se opto por la construcción de un "Grafo" el cual tiene las características de ser "Dirigido" y "Ponderado". En el primer caso, por el hecho de que en una búsqueda, un género podía provenir de otro género, pero no implicaba que también exista la relación en viceversa. En el segundo caso, y en relación al primer punto, se opto por éste por el hecho de que se requería el servicio de saber cuántas veces se paso de un genero a otro. De esta manera el peso de la arista se va modificando dinámicamente.

Por otra parte, se selecciono la "Lista de Adyacencias", por la incógnita de saber cuántos géneros se podían ingresar en una búsqueda, o en múltiples búsquedas, situación más complicada en una "Matriz de adyacencias", ya que tienen un tamaño definido previamente. A su vez, al ser "Ponderado" y "Dirigido", en una "Matriz de Adyacencias" se hubiera utilizado espacio sobrante, el cual no sería utilizado, ya que existirían menos de la mitad de las relaciones posibles.

Una vez seleccionado un Grafo el cual debía ser "Dirigido" y "Ponderado" a partir de una "Lista de Adyacencias", se planteó que "Clases" debían existir, como así también como se iban a estructurar cada una de ellas.

Para ello, y teniendo en cuenta a las funcionalidades realizadas para la primera entrega, se determinó las siguientes "Clases" con sus principales características:

- **Género:** Es la encargada de poseer el género, como así también una lista de los géneros "afines". De esta manera, esta clase corresponde a los "Vértices" o "Nodos" dentro de un grafo.
- **Arco Género:** Esta clase corresponde a la "Arista" que conecta dos "Vértices".
- **Grafo Géneros:** Constituye a la estructura del "Grafo". Para ello se compone lista que contiene todos los "Vértices" existentes
- **Índice:** En relación a la primera entrega, es la encargada de recibir las consultas de búsqueda por género.
- **Biblioteca:** En relación a la primera entrega, constituye el "main" del programa desarrollado.

Una vez esquematizada la relación entre las clases que se debían utilizar, se selecciono que estructuras debían poseer cada una para su correcto funcionamiento.

Por ello, para el "Género", se utilizo un "String" para contener el nombre del género asociado, y una "LinkedList" compuesta por "Arco Género". Dicha selección se realizo teniendo en cuenta de que es el "Género" el que conoce a los próximos géneros asociados.

Por su parte, el "Arco Género" posee un "int", el cual corresponde a un contador de las veces que se paso del primer género al segundo, y "Genero", el cual corresponde al próximo género desde un "Género" de origen. Es por ello que existirán tantos arcos (aristas) como caminos posibles.

En tercer lugar, la clase "Grafo Género" debía poseer los métodos necesarios para cumplir con los servicios requeridos, los cuales son:

- Obtener los N géneros más buscados luego de buscar por el género A.
- Obtener todos los géneros que fueron buscados luego de buscar por el género A.
- Obtener el grafo únicamente con los géneros afines, es decir, que se vinculan entre sí (pasando o no por otros géneros).

Para el primer servicio, se determino que se debían obtener los "adyacentes" de un género determinado, y obtener los que posean los "Arcos Género" con mayor valor. Para ello se utiliza el método "getMasBuscados" dentro del "Índice" el cual recibe como parametros a un "String" con el "genero A" y un "int N" el cual indica la cantidad de géneros que filtrar u obtener. Desde el método mencionado anteriormente, se utiliza el método "getProximos", implementado en el grafo, el cual obtiene todos los arcos del genero A.

En relación a lo anterior, a la clase "Arco" se le implementó el método "compareTo" para poder utilizar "collection.sort", el cual se encarga de ordenar una lista determinada, de mayor a menor, con el objetivo de obtener "N Géneros" que se encontraran al comienzo de la lista.

Para el segundo servicio, se utiliza el método "getBuscadosPostGenero", el cual recibe como parámetro a un "género A". Desde ahí, se hace una llamada al método del grafo denominado "getPostGenero", el cual devuelve una lista con los generos recorrido a partir de A. A su vez, desde este ultimo se invoca al método "recorridoGenero" que es el encargado de recorrer, en profundidad (dfs), al grafo, considerando a los generos que se vayan vinculando, teniendo como origen al "género A".

Para dicha implementación se determino utilizar un "HashMap" compuesto de "String, String", el cual es denominado "estado". De esta manera, se puede acceder a dicha estructura a través de una "clave", la cual se corresponde a un género determinado. Con la utilización de dicha estructura, a cada "género" se le asignó un estado el cual puede ir modificando a medida que se recorra el grafo. Los "estados" existentes o utilizados son:

- **BLANCO**: Significa que un "vértice" no fue visitado.
- **AMARILLO**: Significa que está siendo recorrido pero no finalizo.
- **NEGRO**: Significa que la fue visitado completamente.

Para el tercer servicio, se accede con el método "getGénerosAfines" dentro de la clase "Índice", la cual retornara una "Lista" con los géneros vinculaos entre sí (es decir que conforman un ciclo). Para ello, dicho método hace uso de otro, denominado "getGénerosAfines", el cual se ubica dentro del grafo. Este será el encargado de verificar si se ha encontrado un "ciclo", a partir de un género determinado.

Por su parte, para dicho servicio, también se hizo uso de un "HashMap <String, String>", denominado "padres", el cual va almacenando, para cada género, un género "padre" que es por el cual se accedió.

Por último, una vez encontrado un ciclo, se guardan los géneros correspondientes en variables denominadas "padreCiclo" e "hijoCiclo", los cuales corresponden al inicio y fin del ciclo. A partir de ellos se recorrerá el ciclo y se lo almacenara, a través de los nombres de los géneros pertenecientes al ciclo, devolviendo un "Grafo". Por cuestiones prácticas, desde el "Índice" se obtiene dicho "Grafo", traspasa la información obtenida a una "LinkedList" y lo imprime.

2.1 Aplicaciones Prácticas

Para evaluar la eficacia de la implementación, se realizaron se realizaron una serie pruebas para observar el comportamiento en la carga de datos, a partir de los archivos provistos por la cátedra ("dataset1.csv", "dataset2.csv", "dataset3.csv" y "dataset4.csv"), como a su vez para la búsqueda de un géneros en base a cada archivo cargado previamente.

Para ello se tomaron dos parámetros. En primer lugar se utilizo la clase "Timer" para observar el tiempo que tardaba cada operación.

En segundo lugar se busco generar la cantidad de iteraciones realizadas en las diferentes operaciones.

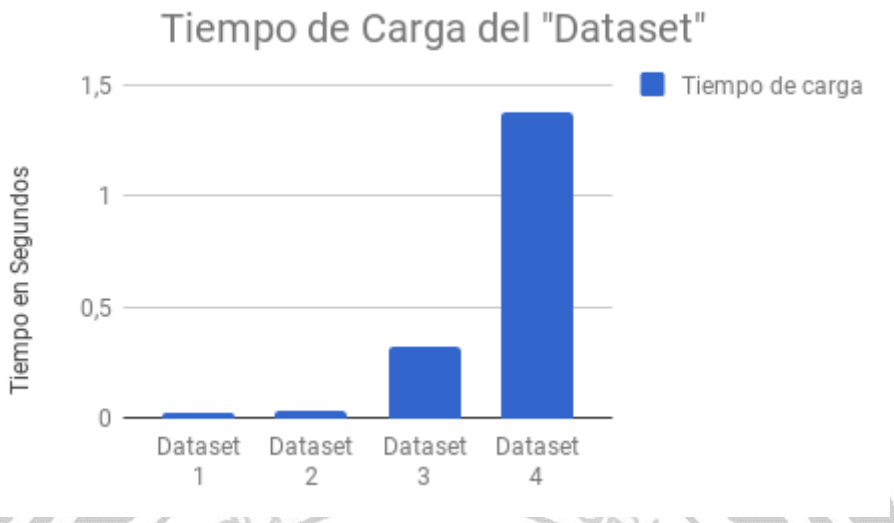

Imagen 1: "Tiempo de carga según Dataset". **Fuente**: Elaboración Propia

Como se puede observar en la "Imagen 1", el tiempo de carga de datos varia progresivamente, en relación a la cantidad de datos que contiene cada "Dataset".

Con respecto al tiempo de búsqueda, tanto del ciclo como del DFS se tomaron como parámetros, en cada uno de los dataset, a los géneros "Viajes", "Romance" y "Terror".

Como se puede observar en las siguientes imágenes (Imagen 1 y 2), el tiempo de búsqueda va a variar en relación a los géneros que posee, como así también a la relación preexistente entre ellos, en cada "Dataset", en base al género seleccionado.

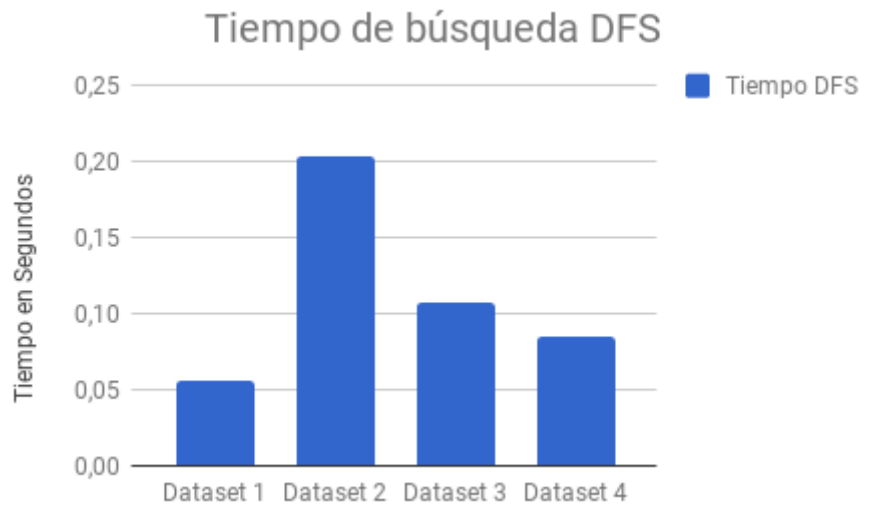

Imagen 2: "Tiempo de carga de búsqueda DFS según Dataset". **Fuente**: Elaboración Propia

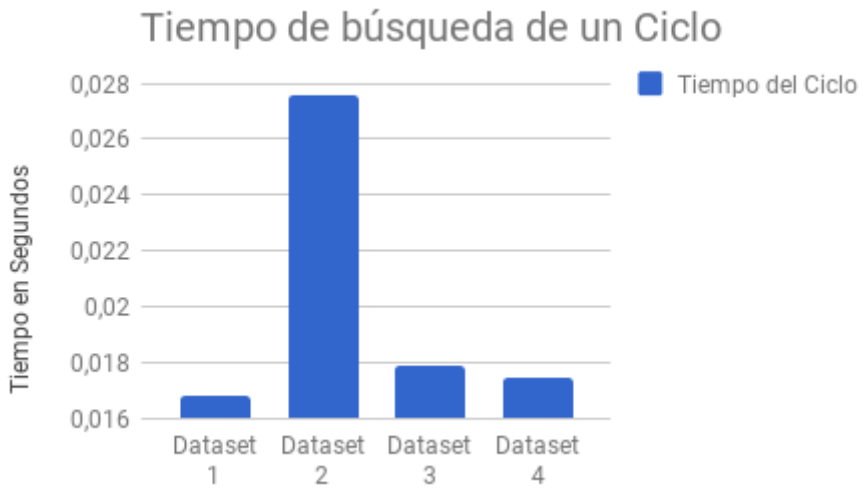

Imagen 3: "Tiempo de carga de búsqueda de Ciclo según Dataset". **Fuente**: Elaboración Propia

Con respecto a la cantidad de "Iteraciones", como se puede obserar en las siguientes imágenes (Imagen 4 y 5), las iteraciones representan la cantidad de géneros en cada "Dataset". En la imagen 4, se puede observar que, salvo el "Dataset 1", presentan un total de 40 géneros diferentes.

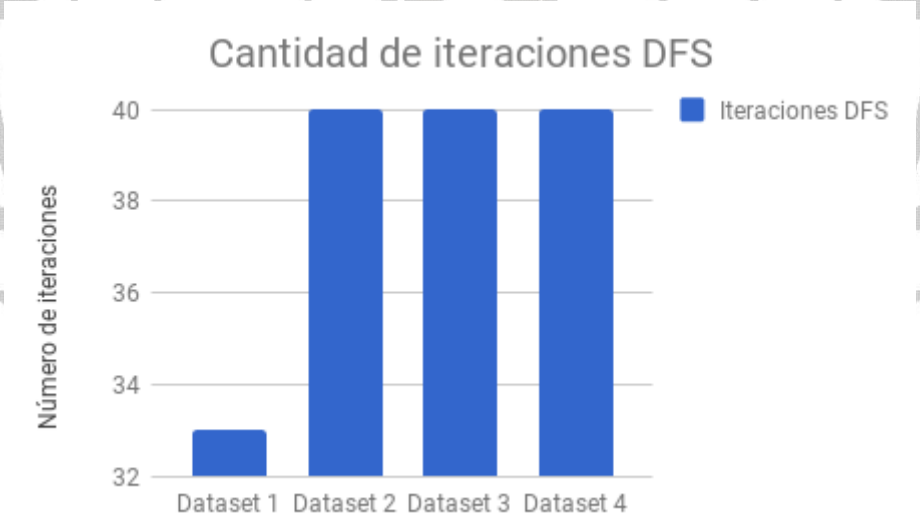

Imagen 4: "Cantidad de iteraciones en la búsqueda DFS según Dataset". **Fuente**: Elaboración Propia

Por su parte, cuando se busca un ciclo, este presenta valores similares, ya que al encontrar el primer ciclo lo devuelve. Al igual que el anterior, variara según la relación existente entre los géneros, según el "Dataset", como así también al género inicial.

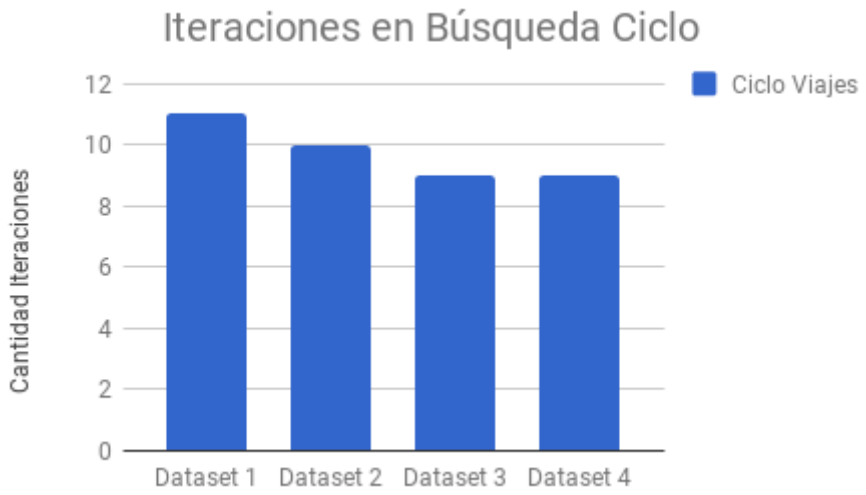

Imagen 5: "Cantidad de iteraciones en la búsqueda de Ciclo según Dataset". **Fuente**: Elaboración Propia

3. Conclusiones

Para finalizar, cabe destacar que se utilizaron estructuras no vistas en la cátedra (HashMap), porque resultaba más apropiado para recorrer una estructura, a partir de la utilización de una "clave → valor", simplificando la implementación, ya que de otra manera, habría que utilizar un arreglo (u otra estructura a seleccionar), y recorriéndola todas las veces necesarias, para comprobar el "estado" o el "padre" de un género determinado.

De esta forma, se selecciona directamente al género buscado, y se le asigna el valor deseado.