2018

.

TRABAJO PRÁCTICO PROGRAMACIÓN 3 - TUDAI

ETAPA 1/CORRECCION

CARTON NELSON - VÁZQUEZ RODRIGO

NELSONCARTON@GMAIL.COM
RODRIGOVAZQUEZ420@GMAIL.COM

FACULTAD DE CIENCIAS EXACTAS
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Introducción

1 PROBLEMA A RESOLVER

En este trabajo se busca implementar un algoritmo que permita listar, a partir de un archivo .CSV provisto por la cátedra, una cantidad de libros con sus respectivas características.Luego se debe poder realizar un filtrado por géneros de los mismos, y devolver un archivo .CSV con los libros que cumplan con la condición dada.

Desarrollo

2 DECISIONES DE DISEÑO

Entre las estructuras que se ofrecieron:

1. Una lista simplemente vinculada.
2. Alguna de las implementaciones conocidas de la interface List de Java.
3. Un árbol binario de búsqueda.

Decidimos optar por ArrayList en el índice de géneros como en el listado libros, debido a que:

- En el caso de la colección de libros solo se requería tener el listado de los mismos por lo que no se necesitaba una estructura optimizada para búsqueda como lo es **ABB**.

- En el caso del índice de géneros, como era necesario realizar búsquedas en una lista de gran tamaño debíamos optar por utilizar un **ABB** o mantener ordenada una lista y realizar búsqueda binaria, ya que la complejidad computacional de estos es por lo general de **O(log_2(n))**, donde n es es la cantidad de elementos que posee la estructura, comparado con **O(n)** de la busqueda lineal. El arbol lo descartamos debido a que, para su correcto funcionamiento, este debe encontrarse balanceado y la complejidad de desarrollar dicha tarea nos pareció mucho mayor a implementar una busqueda binaria en una lista, la cual posee varios metodos ya predefinidos que nos permiten una mejor utilizacion y ahorrarnos varias lineas de codigo. Aquí entonces se descarta tambien la utilización de la lista simplemente vinculada porque para el desarrollo de esta busqueda necesitamos acceder a la posición del medio directamente (Cosa que en la lista simplemente vinculada no se puede hacer).

3 IMPLEMENTACIÓN

Como principio implementamos la clase **Libro** el cual posee como atributos todos los campos necesarios para poder crear un libro a partir del dataset (titulo, autor, cantidad de páginas, array de géneros) y la clase **genero** con su nombre y una lista de libros que cumple con el requisito de poseerlo en su lista de géneros.

Luego pasamos a incorporar la clase **indiceGenero**, la cual va a cumplir con la mayoría de los requerimientos del practico. Como inicio, al instanciar la clase, recibe como parámetro una lista de libros provenientes de los datasets, por cada uno de estos va a preguntar si cada genero perteneciente a dicho libro se encuentra en la lista de géneros principal. En el caso de que el género ya este en la lista solo introduce el libro en el array de ese género, de lo contrario creara un nuevo género, lo inserta ordenado en el arreglo de géneros principal (lo mantenemos ordenado para poder implementar la búsqueda binaria) y luego le asignara ese libro a su lista.

Por último, añadimos los métodos **buscarLibros** y **busquedaBinariaLibro** los cuales retornan un arrayList de libros que cumplen con un género dado. Con estos dos métodos podemos observar las diferencias entre la utilización de una búsqueda lineal y una binaria.

En la clase **simulador**(main) leemos los datasets y cargamos los libros en un array, luego con esa lista generamos la lista de géneros como explicamos anteriormente e implementamos las búsquedas para analizar los resultados obtenidos.

También generamos un archivo .csv donde volcamos los resultados de los libros filtrados.

4 RESULTADOS OBTENIDOS EN LAS PRUEBAS

Al principio desarrollamos el problema con una lista de géneros desordenada buscando linealmente una cantidad de generos dados y luego realizmos el promedio de los resultados obtenidos.Para saber cuanto tiempo nos tomo la busqueda utilizamos la funcion que nos provee java **System.nanoTime** y para las comparaciones implementamos un contador por cada vez que itera hasta encontrar el genero pedido, tanto en la funcion de busqueda lineal como binaria

Búsqueda lineal		
	Tiempo de ejecución(ns)	Comparaciones
Dataset 1	34282	21.75
Dataset 2	40555	22.8
Dataset 3	25800	22.5
Dataset4	30100	15.25

Luego recolectamos los datos de una busqueda binaria implementada en el arreglo de generos, esta vez ordenado desde un principio.

Búsqueda binaria		
	Tiempo de ejecución(ns)	Comparaciones
Dataset 1	44111	4.6
Dataset 2	55200	4.6
Dataset 3	51200	4.6
Dateset4	46900	4.6

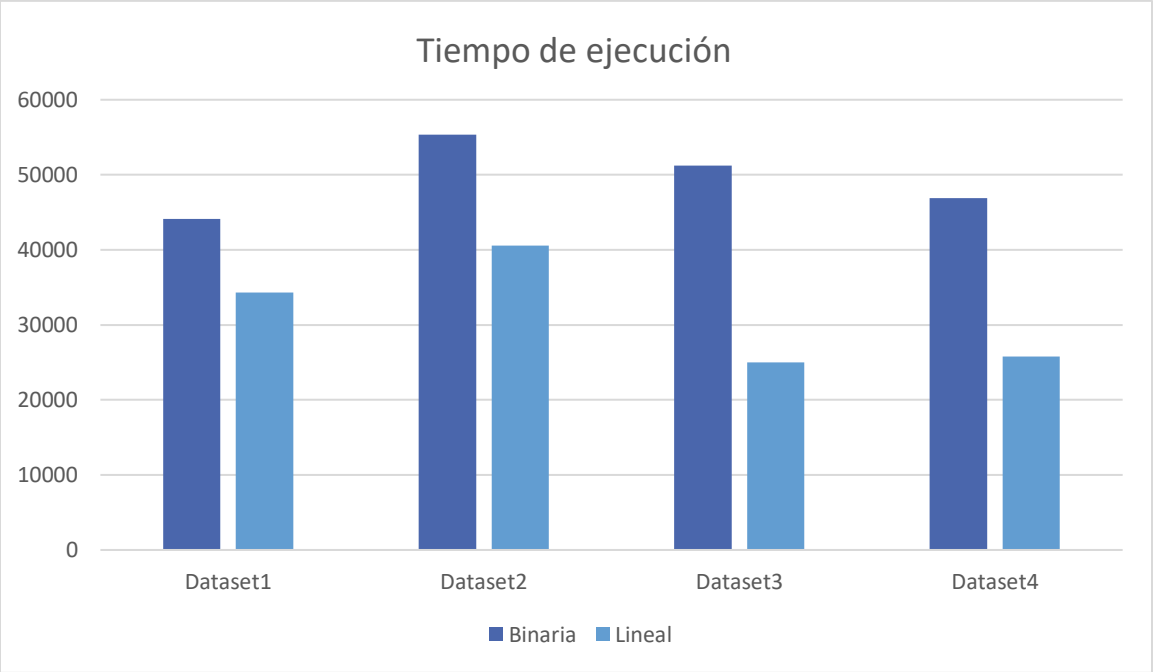
Para recolectar estos datos implementamos busquedas de generos que se encuentran en distintas partes de la estructura para luego generar un promedio y asi nos de como resultado datos un poco mas concretos del comportamiento de la busqueda.

Como podemos observar la cantidad de iteraciones entre ambos tipos de busquedas dan un claro resultado de cual es mejor en cuanto a complejidad. Vemos que los tiempos de búsqueda son muy similares en todos los casos, suponemos que esto es debido a que, en primer lugar es una medida poco confiable ya que entran en juego ciertos factores como la cache, la mamoria y el tiempo de respuesta de la maquina en donde se ejecuta el programa, y ademas el arreglo de géneros no cambia significativamente su tamaño al cambiar de dataset, creemos que si se incrementara el tamaño del array de géneros veriamos resultados aun mas concretos que muestren el beneficio de la busueda binaria sobre la lineal para estos casos, ya que en teoria empieza a arrojar mejores resultados cuanto mayor sea la estructura donde se implementa.

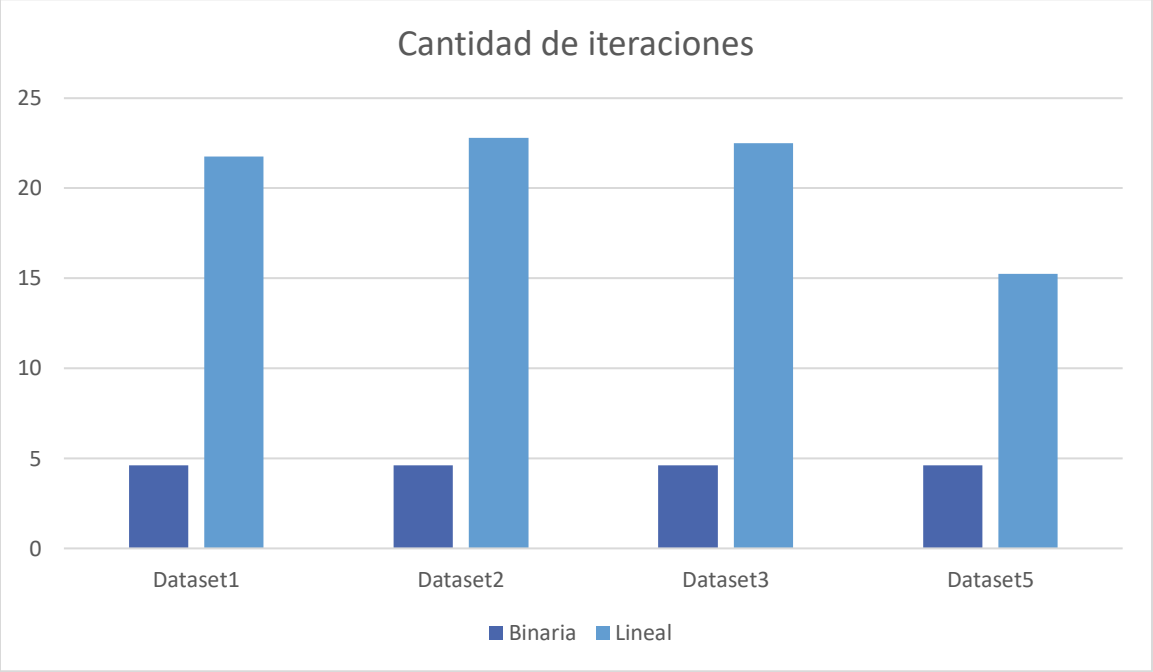
5 GRÁFICOS

A continuacion dejamos los graficos comparativos de ambas busquedas.

Tiempo de busqueda(nanosegundos)



Cantidad de iteraciones en el array de géneros



En el último grafico referido a las iteraciones vemos lo mencionado anteriormente, una gran diferencia comparando los dos tipos de búsquedas en la cual podemos observar el beneficio de la búsqueda binaria.

6 CONCLUSIÓN

Al final de esta etapa descubrimos el impacto de costo que puede tener la elección de una estructura equivocada. Si bien los gráficos de **tiempo de ejecución** no muestran una gran diferencia (debido quizás a factores del estado de la memoria en ese mismo instante, cache, etc.) si puede resaltarse que la búsqueda binaria gana con gran ventaja, con respecto a las iteraciones, comparada con la búsqueda lineal.

Si expandiéramos más la estructura de donde están contenidos los géneros podríamos observar una mayor diferencia entre ambas búsquedas, así como también una mayor eficiencia, ya que mientras el algoritmo de la búsqueda binaria va dividiendo el arreglo a la mitad (**$\log(O_2(n))$**), la lineal en el peor de los casos nos llevaría a recorrer toda la estructura lo cual tendría un costo enorme relacionado a el tamaño del arreglo (**O(n)**)