2018

.

TRABAJO PRÁCTICO PROGRAMACIÓN 3 - TUDAI

ETAPA 1

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(n))** 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. 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 RESULTADOS OBTENIDOS EN LAS PRUEBAS

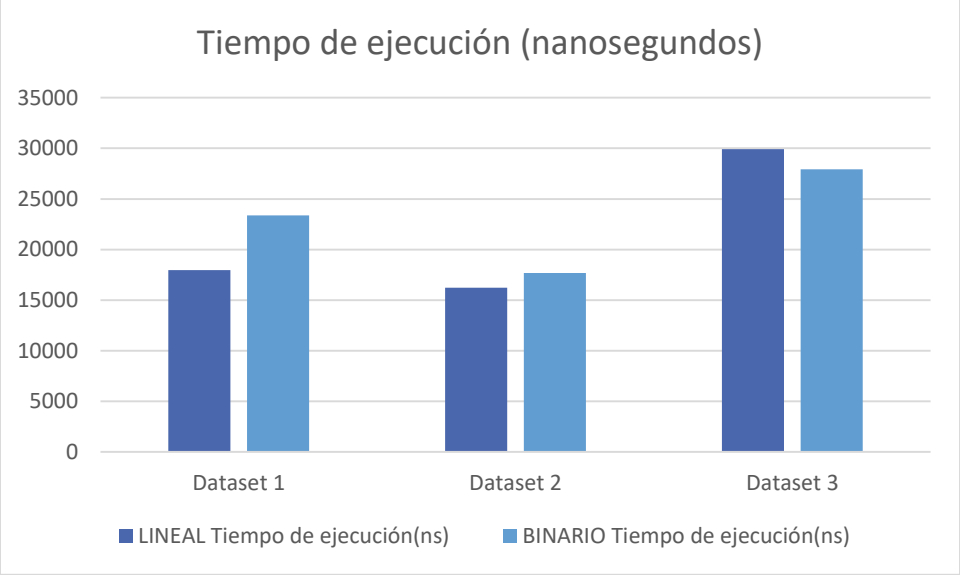
Al principio desarrollamos el problema con una lista de géneros desordenada buscando linealmente y estos fueron los datos obtenidos al buscar el genero “infantil”:

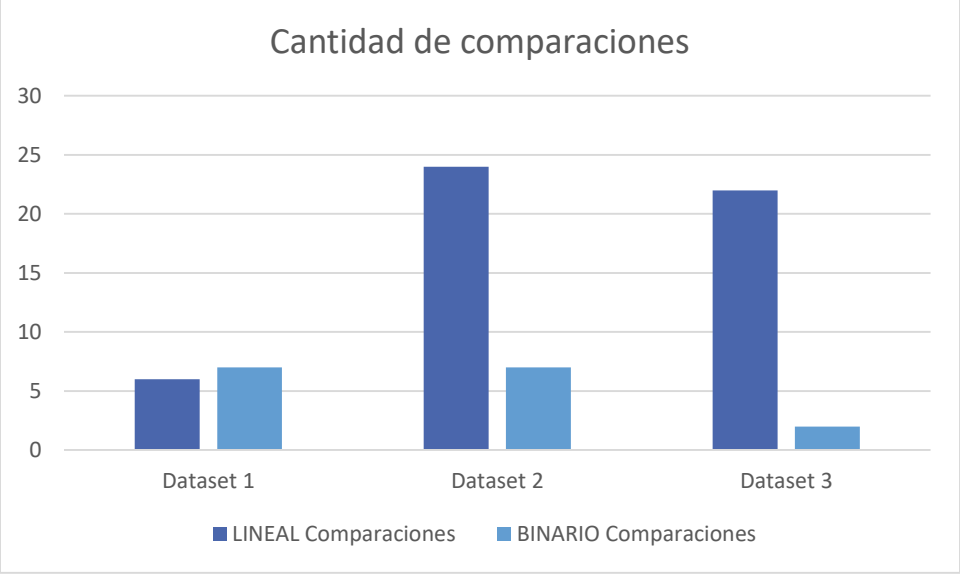
Búsqueda lineal		
	Tiempo de ejecución(ns)	Comparaciones
Dataset 1	17953	6
Dataset 2	16242	24
Dataset 3	29921	22

Estos fueron los resultados luego de mantener ordenada la lista al insertar y hacer una busqueda binaria

Búsqueda binaria		
	Tiempo de ejecución(ns)	Comparaciones
Dataset 1	23367	7
Dataset 2	17668	7
Dataset 3	27927	2

GRÁFICOS


Tiempo de ejecución (nanosegundos)
35000
30000
25000
20000
15000
10000
5000
0
Dataset 1
Dataset 2
Dataset 3
LINEAL Tiempo de ejecución(ns)
BINARIO Tiempo de ejecución(ns)


Cantidad de comparaciones
30
25
20
15
10
5
0
Dataset 1
Dataset 2
Dataset 3
LINEAL Comparaciones
BINARIO Comparaciones

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 en velocidad en el Dataset más grande.

La diferencia más grande se encuentra si observamos la **cantidad de comparaciones** realizadas por cada algoritmo, las cuales son mucho menores a medida que aumenta el tamaño de los datos para la búsqueda binaria. Mientras que en la búsqueda lineal en el peor de los casos nos llevaría a recorrer toda la lista.