Fecha de Entrega: 03/06/2022

**Trabajo Práctico Especial**

Primera Entrega

**Programación III**

TUDAI, Facultad de Ciencias Exactas, UNICEN

Integrantes:

* Lautaro Sánchez   
  [lautasanchez11@gmail.com](mailto:lautasanchez11@gmail.com)
* Joaquín Iglesias Coqueti  
  [jiglesiascoqueti@alumnos.exa.unicen.edu.ar](mailto:jiglesiascoqueti@alumnos.exa.unicen.edu.ar)

Link GitHub: <https://github.com/exajoakinic/ProgIII_TPE>

**Introducción:**

El presente trabajo práctico tiene como objetivo el desarrollo de una herramienta de indexado utilizando una estructura de almacenamiento de datos a elección.

Para ello se cuenta un listado de archivos .csv con tamaños considerablemente diferentes. Éstos contienen un listado de libros, a partir de los cuales se deben generar los índices por género.

Así mismo se recibe como ayuda funciones aproximadas tanto para poder leer los datos de entrada almacenados en los archivos .csv, como para generar la escritura de los datos de salida.

Se solicita que a partir de un género cualquiera dado se genere un nuevo archivo csv con el listado de los títulos de los libros que pertenezcan a ese género.

**Desarrollo del trabajo:**

Para generar el listado de los libros en memoria, consideramos que la mejor estructura es el LinkedList. Si bien tanto para esta estructura como el ArrayList se acostumbra a decir que el costo de agregar elementos es O(1), no siempre es así para el ArrayList, dado que internamente tiene un arreglo fijo, generando el inconveniente de que cuando se llena debe crear uno nuevo más grande copiando todos los elementos en él.

En cuanto a los índices, la estructura elegida es el Árbol Binario de Búsqueda. Esta elección se sustenta en el hecho de que las operaciones a desarrollar sobre éste son la inserción de libros por géneros y la búsqueda por género. Ambas funciones tienen la misma complejidad *O(h)* dado que en la inserción primero debe encontrar el nodo o el elemento en el cual está situado ese género o bien dónde se debe agregar, para luego insertar el libro (*O(1)* porque para la lista de libros utilizamos LinkedList). Si bien es cierto que en el peor de los casos el árbol va a quedar con la misma estructura que un LinkedList (árbol degenerado o enredadera), la complejidad de ambas operaciones tiende a ser log2cantidadGeneros en el caso promedio.

Por medio de la implementación de un Árbol de Búsqueda Binario AVL, es decir un árbol que se balancea automáticamente cada vez que se produce una inserción o se elimina un elemento, se puede garantizar que las búsquedas sean siempre O(log2cantidadGeneros).[[1]](#footnote-1)

Eligiendo un ArrayList para el índice se puede lograr una búsqueda eficiente por medio de búsqueda binaria que va a ser O(log2n), sin embargo, esto requiere mantener siempre ordenada la lista, con lo cual cuando se deba agregar un género nuevo, este proceso será O(n) y por lo tanto de menor eficiencia que el ABB. Dicho problema no puede ser solucionado por un LinkedList, dado que no posee la eficiencia en el acceso aleatorio necesaria para la búsqueda binaria.

Se adjunta a continuación tablas con diferentes mediciones de tiempos de ejecución.

|  |  |  |  |
| --- | --- | --- | --- |
| Leer e indexar .csv | | | |
| Origen | Tiempo (ms) | Cant Libros | Tiempo por libro |
| dataset1 | 7.5 | 20 | 0.375 |
| dataset2 | 68.1 | 1000 | 0.0681 |
| dataset3 | 589.3 | 100000 | 0.005893 |
| dataset4 | 4081.3 | 1000000 | 0.00040813 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Buscar por género: | | | | | | | | |
| Leyendas | | | Filosofía | | | Inexistente | | |
| Origen | Tiempo (ms) | Cant Libros | Tiempo por libro | Tiempo (ms) | Cant Libros | Tiempo por libro | Tiempo (ms) | Cant Libros | Tiempo por libro |
| dataset1 | 3 | 4 | 0.75 | 1.5 | 2 | 0.75 | 1.5 | 0 | - |
| dataset2 | 12.7 | 155 | 0.0819 | 2.3 | 148 | 0.0155 | 0.8 | 0 | - |
| dataset3 | 28.9 | 13618 | 0.00212 | 15.9 | 13991 | 0.00113 | 0.7 | 0 | - |
| dataset4 | 82.9 | 138235 | 0.000599 | 62.5 | 137935 | 0.000453 | 1.1 | 0 | - |

1. Dicho árbol no ha sido implementado en el presente trabajo, sólo se lo menciona como una posible mejora de eficiencia. [↑](#footnote-ref-1)