BORRADOR- Pendiente para pasar al formato

**Título:** Estructura de datos para listar el contenido de un directorio.

**Autores:**

Kevin Arley Parra Henao

Daniel Alejandro Mesa Arango

**Introducción:**

Lo que dice en el pdf:

Los avances en la tecnología han hecho que hoy en día sea necesario almacenar y procesar grandes volúmenes de información. Para poder manejar la información en el disco duro se utiliza un sistema de archivos. Un sistema de archivos es un componente de un sistema operativo encargado de asignar espacio a los archivos, administrar el espacio libre y permitir el acceso a los datos guardados. Hoy en día existe la necesidad de almacenar y acceder a un número muy grande de archivos en un sistema de archivos.

Desafortunadamente, sistemas de archivos como ext2 soportan máximo 32.000 archivos dentro de un directorio y esto no es suficiente para las aplicaciones que se utilizan hoy en día. Un problema que tiene ext2 es que el contenido de cada directorio se representa como una lista enlazada y esto no es eficiente para un número grande de archivos. Las empresas que manejan altos volúmenes de datos no estaban conformes con las limitaciones de ext2. Como un ejemplo, Google actualizó, en el 2010, toda su infraestructura del sistema archivos ext2 al sistema de archivos ext4.

**Problema:**

De acuerdo a la motivación, el problema de listar el contenido de un directorio consiste en encontrar eficientemente, los archivos y subdirectorios que se encuentran en un directorio. El objetivo de esta práctica es desarrollar una estructura de datos para representar los directorios y archivos en un sistema de archivos, y permitir consultar eficientemente los archivos y subdirectorios que se encuentren en un directorio.

**Soluciones para este problema:**

Realizando el informe, investigamos cuatro árboles que pueden servir para afrontar la realización del proyecto, además de la descripción que se dará a continuación, tenemos la Figura 3, en la cual se pueden encontrar varias características sobre la complejidad de estos árboles y otros más.

**Árbol AVL:**

Un árbol AVL es un árbol binario de búsqueda que cumple con la condición de que la diferencia entre las alturas de los subárboles de cada uno de sus nodos es, como mucho 1. La denominación de árbol AVL viene dada por los creadores de tal estructura (Adelson-Velskii y Landis). Recordamos que un árbol binario de búsqueda es un árbol binario en el cual cada nodo cumple con que todos los nodos de su subárbol izquierdo son menores que la raíz y todos los nodos del subárbol derecho son mayores que la raíz. Recordamos también que el tiempo de las operaciones sobre un árbol binario de búsqueda son O(log n) promedio, pero el peor caso es O(n), donde n es el número de elementos.

A continuación de muestra un ejemplo sencillo de un árbol AVL:

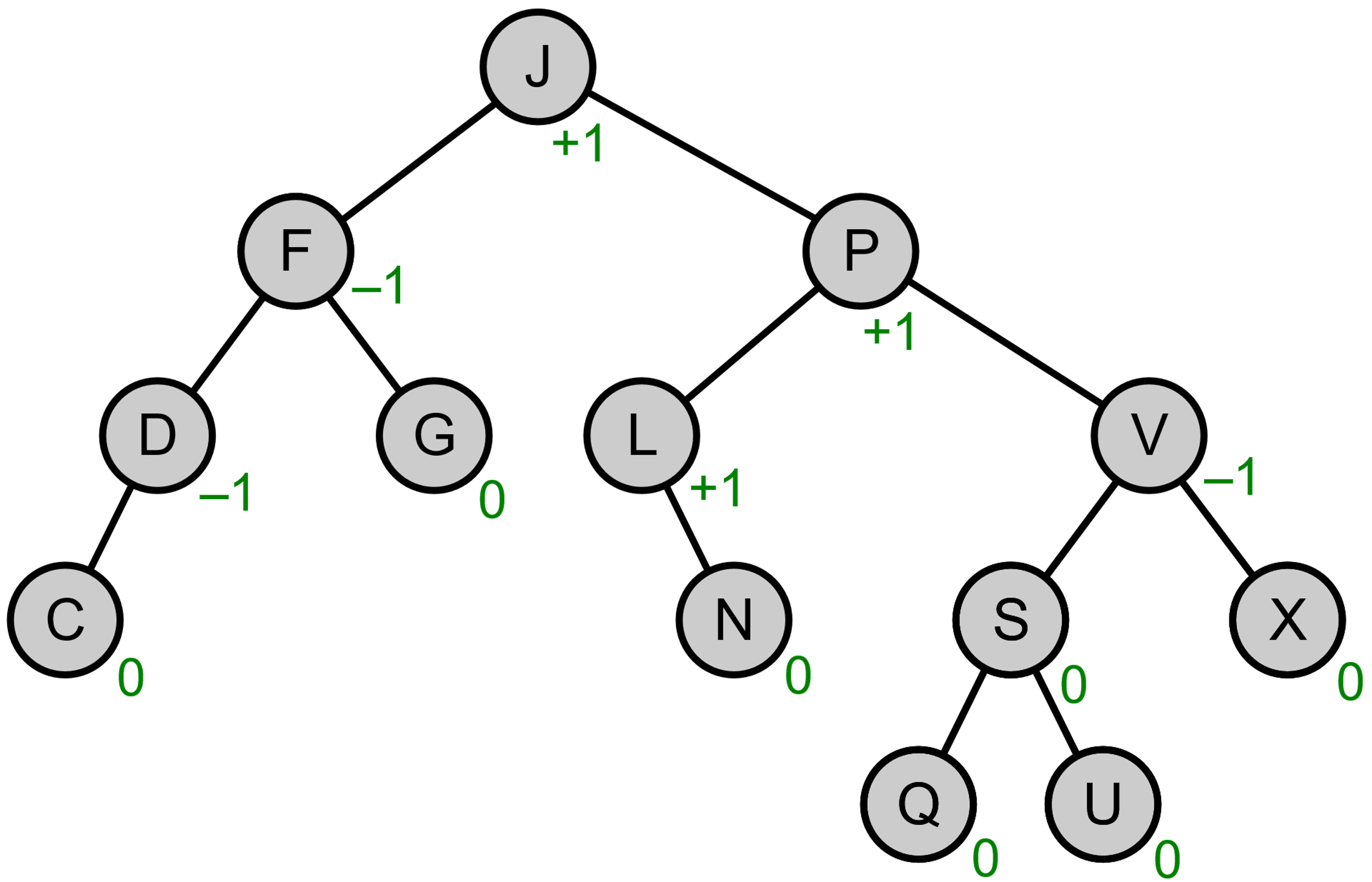


Figura 1

**Árbol biselado:**

Un Árbol biselado o Árbol Splay es un Árbol binario de búsqueda auto-balanceable, con la propiedad adicional de que a los elementos accedidos recientemente se accederá más rápidamente en accesos posteriores. Realiza operaciones básicas como pueden ser la inserción, la búsqueda y el borrado en un tiempo del orden de O(log n). Para muchas secuencias no uniformes de operaciones, el árbol biselado se comporta mejor que otros árboles de búsqueda, incluso cuando el patrón específico de la secuencia es desconocido. Esta estructura de datos fue inventada por Robert Tarjan y Daniel Sleator.

Todas las operaciones normales de un árbol binario de búsqueda son combinadas con una operación básica, llamada biselación. Esta operación consiste en reorganizar el árbol para un cierto elemento, colocando éste en la raíz. Una manera de hacerlo es realizando primero una búsqueda binaria en el árbol para encontrar el elemento en cuestión y, a continuación, usar rotaciones de árboles de una manera específica para traer el elemento a la cima. Alternativamente, un algoritmo "de arriba abajo" puede combinar la búsqueda y la reorganización del árbol en una sola fase.

Un ejemplo grafico de un árbol biselado seria:

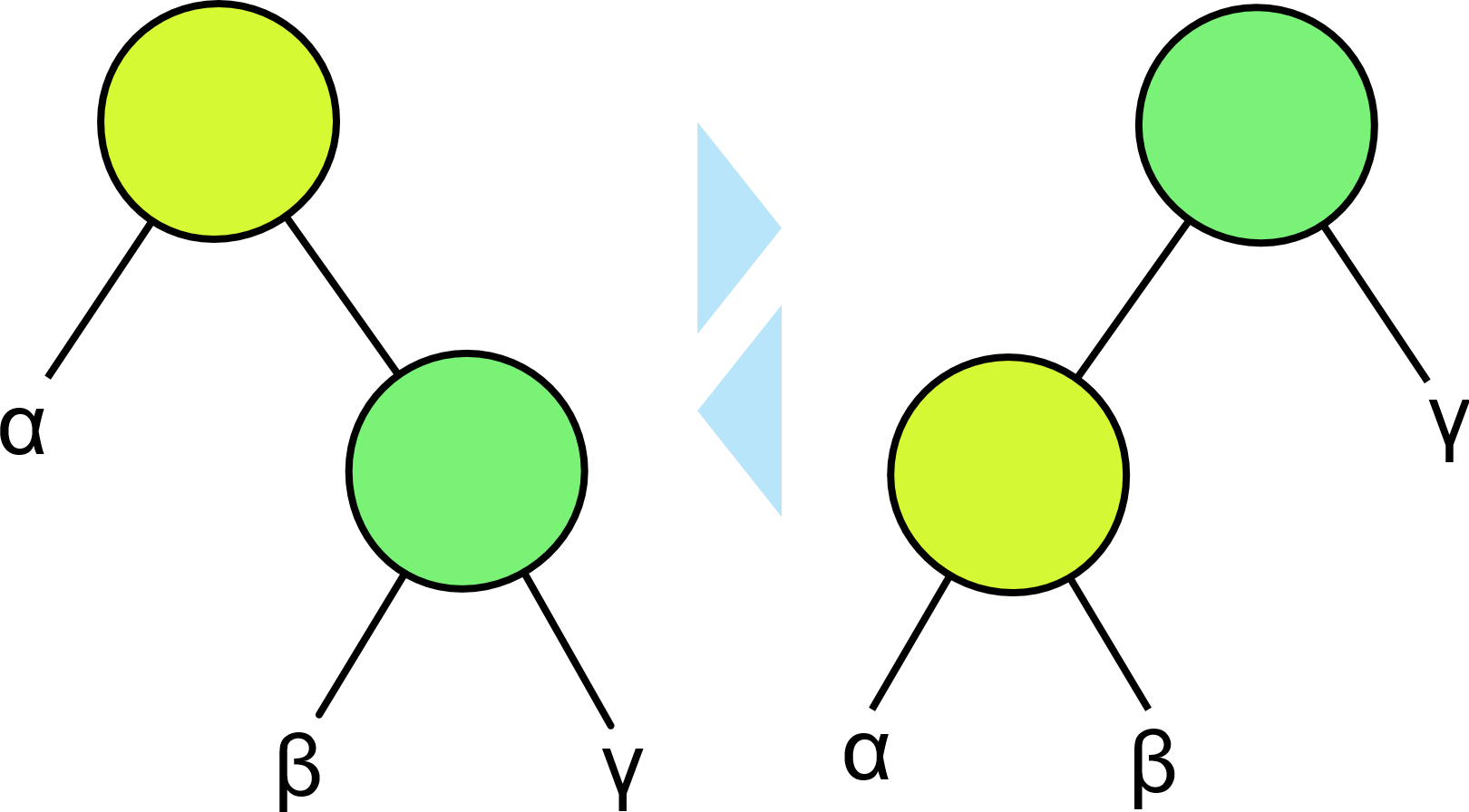


Figura 2

**Elección de la estructura de datos:**

Elegimos la estructura de datos B-tree para empezar, ya que nos parece que puede servir, y no decidimos intentar crear una nueva ya que como dice un principio del libro 201 principles of software development [[[1]](#endnote-1)], “no invente la rueda”. Sin embargo, dejamos abiertas las posibilidades de cambiar más delante de estructura o incluso tratar de implementar alguna si vemos que la elegida no funciona adecuadamente. A continuación, exponemos las razones por las cuales elegimos la estructura B-tree.

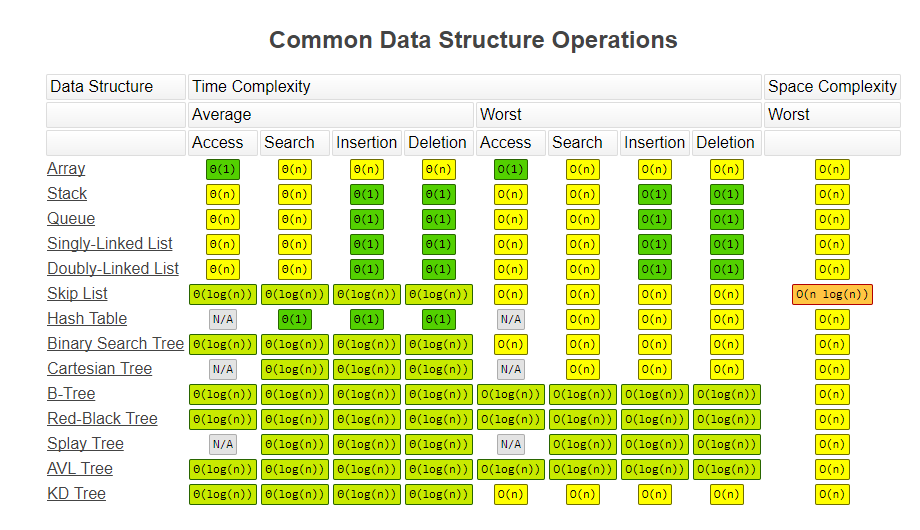


Figura 3

Basados en la figura 3, podemos ver que hay varias estructuras cuya complejidad es bastante notable, entre ellas está la B-tree, por eso decidimos elegirla, por su nivel de complejidad y porque tiene muchas características, ya que otras estructuras en la tabla incluso tenían mejor complejidad en algunos aspectos que la B-tree pero en otros su complejidad no era tan buena. Esa es otra razón por la cual escogimos el B-tree, ya que está bastante nivelado en todos los aspectos.

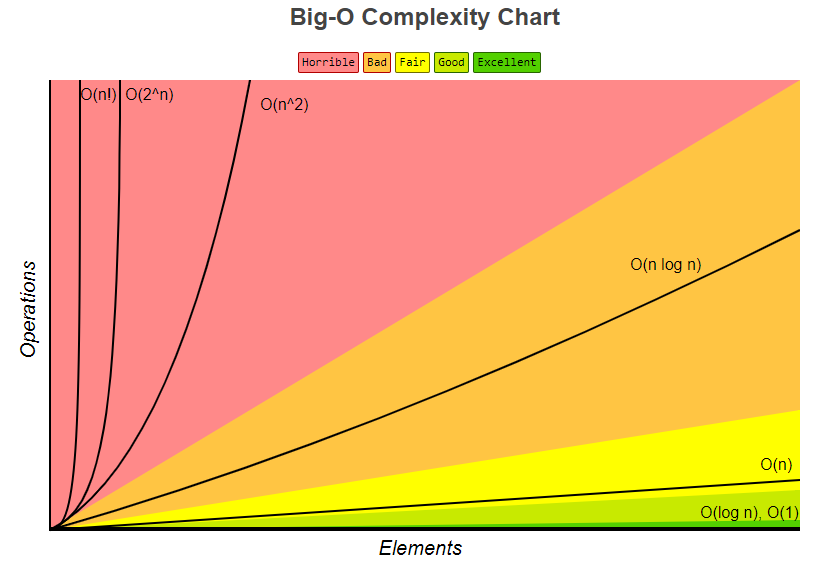


Figura 4

La figura 4 nos da una idea bastante aceptable para entender la complejidad y nos apoyamos en esta también para elegir la estructura de árbol B-tree.

Profesor en el apartado e) de la entrega 1, a parece que solo es una solucion, pero en la plantilla dice que es una solucion por cada problema, ¿cual es la correcta?

R\= Hola, para ser precisos, la idea es que como aquí está muy claro el problema y es un problema

muy conocido, simplemente busquen 4 soluciones para ese problema, es decir, 4 estructuras de datos

para ese problema y nos cuentan brevemente de qué tratan... y van pensando cuál les puede servir,

o una combinación de varias o si quisieran hacer algo nuevo porque no les sirve lo que ya existe.

Profesor, en la guia para la entrega del proecto usted dejo alguno problemas, como el arbol B o el H tree, pero ¿como se puede buscar una solucion a este tipo de problemas?

R\= Hola de nuevo. Sí, el problema es hacer una estructura de datos para representar eficientemente los directorios y archivos de un computador y permitir hacer búsquedas. Algunas soluciones, es decir, estructuras de datos, para este problema con el árbol B, el árbol H, el árbol B+, el árbol rojo negro, el árbol avl, tablas de hash... entonces esas son

posibles soluciones que conviene ir mirando antes de lanzarse a escoger una o a hacer la propia de uno.

Referencias:

1. Alan M. Davis. 1995. 201 Principles of Software Development. McGraw-Hill, Inc., New York, NY, USA. << formato ACM ESTA LISTO EN ESA REFERENCIA

   Figura 1(Arbol Avl): <https://en.wikipedia.org/wiki/AVL_tree>

   Árbol avl: <http://es.tldp.org/Tutoriales/doc-programacion-arboles-avl/avl-trees.pdf>

   Figura 2() [↑](#endnote-ref-1)