|  |
| --- |
|  |
| **Recuperatorio: Informe** |
| **Ing. Informática**  **Algoritmos y Estructuras de Datos** |
| Bruno Olivera - CI: 4.867.466-7 |
| **Fecha de informe 11/07/2024** |
|  |

# Propuesta

En la Universidad Católica se desea desarrollar un sistema de bibliotecas para los alumnos, que les permitirá solicitar y devolver libros de manera eficiente, el sistema utilizara un árbol AVL para manejar el gran volumen de alumnos que solicitarán un libro, guardando en el cada alumno que tenga un libro prestado, dichos alumnos serán guardados por su número de cedula. AVL es una buena estructura para implementar en este caso ya que proporciona operaciones eficientes de búsqueda, inserción y eliminación, que es exactamente para lo que la usaremos.

Luego tendremos una estructura de Trie junto con un diccionario, para la búsqueda y gestión de libros, en la estructura de trie se almacenarán nombres de los libros, para tener una eficiente indización de estos, el usuario tendrá la capacidad de colocar en un buscador el nombre del libro, categoría o autor y el sistema le sugerirá los posibles resultados encontrados, este nombre se usará en un hashmap como clave, para poder obtener el valor, que será el libro.

Ya que la biblioteca debe cumplir con estrictas reglas de préstamos dictadas por la universidad cada alumno solo podrá llevarse consigo un solo libro. Un trie en este caso se utilizaría más que nada para la búsqueda rápida de nombres de libro, y un hashmap aseguraría un acceso rápido a los libros una vez encontrado el nombre.

Cada libro tendrá una ID, Nombre, Categoría, Autor y su disponibilidad, y cada Alumno contara con una CI, su nombre y el libro que actualmente tiene prestado.

La consigna consiste en crear un SistemaBiblioteca el cual contenga funcionalidades como:

**public void registrarAlumno(String cedula, String nombre)**

**public void registrarLibro(String nombreLibro, String categoria)**

**public boolean prestarLibro(String cedula, String nombreLibro)**

**public boolean devolverLibro(String cedula)**

**public Lista<Libro> buscarLibro(String libro)**

# Algoritmos Utilizados

## **Árbol BB**

Un árbol de búsqueda binaria es una estructura de datos en la que cada nodo tiene como máximo dos hijos. Los nodos se organizan de tal manera que, para cualquier nodo, todos los valores en su subárbol izquierdo son menores y todos los valores en su subárbol derecho son mayores.

## **Árbol AVL**

Un árbol AVL es un tipo de árbol de búsqueda binaria que se mantiene balanceado por medio de rotaciones. Después de cada operación de inserción o eliminación, el árbol se reequilibra automáticamente si sus nodos no difieren de una altura de más de una unidad, garantizando operaciones eficientes. Tiene un orden de ejecución de O (log n) lo que lo hace muy eficaz en lo que hace.

## **Trie**

Un Trie es una estructura de datos en forma de árbol que se utiliza para almacenar un conjunto de cadenas. Cada nodo en un Trie representa un carácter de una cadena. Los Tries son eficientes para búsquedas de prefijos y sugerencias automáticas. La forma en la que los caracteres se almacenan permite hacer búsquedas eficientes cuando comparten prefijos. Su orden de tiempo de ejecución es de O(k) con k siendo la longitud de la clave a buscar.

# Justificación

Se decidió usar un árbol AVL como estructura principal para manejar a los alumnos que desean inscribirse en la librería para poder guardar los datos eficientemente, ya que estos datos podrían ser un gran volumen de datos, este algoritmo tiene un tiempo de O (log n) por cada alumno que se inserta.

Se decidió también usar una estructura de trie para procesar la búsqueda de los libros y guardarlos ya que esta estructura funciona eficientemente con una búsqueda. En esta estructura se recorre el trie por la clave, lo cual tiene un tiempo de ejecución de O(k) siendo k el largo de la clave.  
En este caso seria 3.O(k), ya que el libro tiene que guardarse tres veces, para guardar así el nombre del libro, la categoría y el autor en el trie y poder así llevar a cabo la búsqueda de todos ellos en un solo buscador. Incluso con múltiples inserciones la eficiencia de una complejidad de tiempo lineal sigue permitiendo al programa ser rápido y la flexibilidad de las múltiples inserciones nos permite buscar datos de distintas maneras, incluso si la clave propuesta está incompleta.

# Conclusión

Dado el nuevo requerimiento de búsqueda por autor, categoría y nombre combinado, se decidió descartar la implementación de un hashmap para guardar el dato del libro, en vez de eso se decidió guardar el dato del libro en el nodo (es palabra).

En la función original el hashmap solo contenía el nombre del libro como clave y el trie devolvía dicho nombre para la búsqueda, con la nueva implementación esto ya no es necesario y tampoco eficiente, ya que hashmap solo puede guardar una sola clave, lo que se hacía difícil para guardar claves iguales como lo es una categoría de un libro, ya que de estas habría siempre más de una.

El programa tiene un tiempo de ejecución que como máximo será O (log n) + O(k)).

## Análisis de tiempo por función:

* registrarAlumno()

Inserción en un árbol AVL: O (log n)

* registrarLibro()

Inserción en un Trie: O (k)

Inserción por Nombre, Categoría y Autor: 3.O(k)

* prestarLibro()

Búsqueda del alumno en el árbol AVL: O (log n)

Búsqueda del libro en el trie: O(k)

Complejidad Total: O (log n) + O(k)).

* devolverLibro()

Búsqueda del alumno en el árbol AVL: O (log n)

Búsqueda del libro en el trie: O(k)

Complejidad Total: O (log n) + O(k)).

Actualización del estado del libro y del alumno: O (1)

* buscarLibro()

Filtrado de texto: O(k)

Busqueda en el Trie: O(k)

Complejidad Total: O(k).

Una consideración importante es el almacenamiento, ya que guardar tres veces la información de cada libro puede consumir más espacio. Sin embargo, esta decisión permite una mayor flexibilidad y eficiencia en las búsquedas por múltiples criterios.