Laboratorio 3: Árboles y ordenamiento

Fecha de entrega: 14 de junio a las 23:59

Stopify 2: El Ataque de los Clones

Luego de su exitosa aplicación anterior, inversores de todas las partes del mundo quieren copiarle su gran idea y crear un reproductor para competirle a usted. Para poder diferenciarse, nuevamente acude al poderoso conocimiento de su ramo favorito "Estructura de datos y algoritmos" y pensó en desarrollar un buscador de canciones para su aplicación.

Este laboratorio consiste entonces en la implementación de un buscador para su aplicación implementado sobre un árbol y diferentes métodos de ordenamiento.

Se debe usar, al igual que el laboratorio anterior, una clase Canción:

```
class Cancion {
    public int id;
    public String nombre;
    public String artista;
    public double duracion; // en milisegundos
    public Cancion(String nombre, String artista, double duracion){
        this.nombre = nombre;
        this.artista = artista;
        this.duracion = duracion;
    }
}
```

Para poder funcionar debe implementar los siguientes métodos:

- **agregarCanción(song)**: Recibe una canción y la agrega a una base de datos con todas las canciones. La base de datos es un arreglo o lista.
- **crearArbol(songs)**: Recibe un arreglo o lista de canciones y genera un árbol binario. Cree un árbol desde un arreglo/lista ordenado usando búsqueda binaria (lo que resulta en un árbol balanceado) y cree otro árbol desde un arreglo sin ordenar.
- **buscarCanción(String nombre)**: Recibe el nombre de una canción y devuelve el ID de la canción, sino existe retorna -1. Busque en los dos árboles creados anteriormente.
- **buscarCanción(double duración)**: Recibe una duración en milisegundos y devuelve el ID de las canciones que tienen esa duración como un arreglo o lista. En caso de no haber ninguna canción con esa duración, devuelve un arreglo o lista vacío. Busque en los dos árboles creados anteriormente.
- **ordenarPorNombre()**: Toma una base de datos de canciones (un arreglo o lista) y devuelve la misma estructura en orden lexicográfico.
- **ordenarPorDuracion()**: Toma una base de datos de canciones (un arreglo o lista) y devuelve la misma estructura en orden lexicográfico.

Los métodos de ordenamiento deben ser implementados usando 4 formas: MergeSort, QuickSort, InsertionSort y BubbleSort. Tanto para el ordenpor nombre como por duración, genere una tabla midiendo el tiempo de ejecución de cada método y comente los resultados.

```
long start, finish;

start = System.nanoTime();
dataMergeSort = SortingMethods.MergeSort(dataMergeSort); //Ordena un arreglo
finish = System.nanoTime();
System.out.println("MergeSort time: " + (finish - start) + " ns.");
```

Un ejemplo de cómo medir el tiempo en nanosegundos de cuánto demoró la ejecución de MergeSort.

Ejecute un número Q de queries sobre el árbol ordenado y el desordenado. Comente los resultados.