ALGORITMOS DE BUSQUEDA Y ORDENAMIENTO

Documentación de Clases y Métodos

Elaborado por: Diego Chevez

Repositorio: Búsqueda y Ordenamientos en Java

Índice

1. Introducción

Este documento presenta la documentación técnica del proyecto de algoritmos de búsqueda y ordenamiento en Java, desarrollado como parte del Bootcamp Java de Kodigo. El objetivo es describir la funcionalidad de cada clase, sus responsabilidades y métodos clave, facilitando el mantenimiento, escalabilidad y comprensión del proyecto.

2. Main.java

La clase **Main** es el punto de entrada de la aplicación. Entre sus responsabilidades se encuentran:

- Inicializar las clases Searcher, Sorter y OperationRecord.
- Mostrar el menú interactivo que permite:
- Realizar búsquedas secuenciales y binarias.
- Ejecutar algoritmos de ordenamiento (Bubble Sort, Insertion Sort, Selection Sort).
- Visualizar el historial de operaciones realizadas.
- Salir de la aplicación.

La clase maneja el flujo del programa de forma modular, permitiendo que el usuario elija de forma clara las operaciones a realizar.

3. Searcher.java

La clase **Searcher** se encarga de gestionar los algoritmos de búsqueda secuencial y binaria. Sus características incluyen:

- Mostrar los valores disponibles a buscar antes de solicitar el número al usuario.
- Solicitar el valor a buscar directamente dentro de la función.
- Medir el tiempo de ejecución utilizando System.nanoTime().
- Mostrar la complejidad de cada operación en una sola línea (mejor, peor y promedio).
- Registrar en el historial cada operación realizada, indicando operación, valor buscado, tiempo y complejidad.

Métodos principales:

- **sequentialSearch():** Ejecuta la búsqueda secuencial recorriendo el arreglo hasta encontrar el valor solicitado.
- **binarySearch():** Ejecuta la búsqueda binaria sobre una copia ordenada del arreglo.
- **postSearchMenu():** Muestra un submenú tras cada búsqueda para permitir al usuario realizar otra búsqueda, regresar al menú principal o salir.

4. Sorter.java

La clase **Sorter** gestiona los algoritmos de ordenamiento Bubble Sort, Insertion Sort y Selection Sort. Sus funcionalidades incluyen:

- Mostrar el estado del dataset antes y después del ordenamiento.
- Medir el tiempo de ejecución del algoritmo.
- Mostrar la complejidad de cada algoritmo en una sola línea.
- Registrar cada operación realizada en el historial con información detallada.

Métodos principales:

- **bubbleSort():** Ordena utilizando el algoritmo Bubble Sort.
- **insertionSort():** Ordena utilizando el algoritmo Insertion Sort.
- **selectionSort():** Ordena utilizando el algoritmo Selection Sort.

5. OperationRecord.java

La clase **OperationRecord** gestiona el registro del historial de operaciones realizadas durante la ejecución de la aplicación. Para cada operación, se almacena:

- Nombre de la operación.
- Elemento buscado (si aplica).
- Tiempo de ejecución en nanosegundos.
- Resultado de la operación (índice encontrado o "Sorted").
- Complejidad en sus tres casos: mejor, peor y promedio.

Este historial permite al usuario comparar el rendimiento de cada algoritmo de forma práctica.

6. Util.java

La clase **Util** contiene métodos utilitarios que mejoran la experiencia del usuario, como:

- **clearScreen():** Intenta limpiar la pantalla de la consola de forma multiplataforma.
- **printHeader(String titulo):** Muestra un encabezado con el nombre de la operación en ejecución.

Esto permite mantener el programa ordenado y facilitar su uso.

7. Conclusiones

Este proyecto permitió reforzar habilidades de:

- Programación orientada a objetos (POO).
- Aplicación de principios SOLID para mantener un código modular y limpio.
- Comprensión de algoritmos de búsqueda y ordenamiento.
- Análisis de tiempos de ejecución utilizando System.nanoTime().
- Manejo de menús interactivos y submenús para flujos de usuario claros.
- Generación de datasets aleatorios para prácticas realistas.