

**Universidad Tecnológica de Santiago, (UTESA)**

**Tema:**

Carrera de Algoritmos

Realizado por:

Henri A. Valmes (1-16-0530)

Ramon Minaya (

Docente:

IVAN MENDOZA

Materia:

INF-025-001 ALGORITMOS PARALELOS

**1. Introducción**

La comparación de algoritmos de ordenamiento y búsqueda es una tarea fundamental en el campo de la informática y la ciencia de la computación. Estos algoritmos son esenciales en numerosas aplicaciones, desde bases de datos hasta sistemas de recuperación de información. Entender su eficiencia en términos de tiempo de ejecución y consumo de memoria es crucial para seleccionar el algoritmo más adecuado para cada escenario específico.

**2. Descripción de la aplicación**

La aplicación desarrollada implementa cinco algoritmos clásicos: dos de búsqueda (secuencial y binaria) y tres de ordenamiento (burbuja, quick sort e inserción). La aplicación genera un arreglo aleatorio de números enteros, ejecuta todos los algoritmos de manera concurrente usando hilos, mide tanto el tiempo de ejecución como el consumo de memoria de cada algoritmo, y finalmente presenta los resultados en formato textual y gráfico para facilitar la comparación.

**3. Objetivos**

**a. Objetivo General**

Desarrollar una aplicación que permita comparar el rendimiento de diferentes algoritmos de búsqueda y ordenamiento, ejecutándolos concurrentemente sobre un mismo conjunto de datos.

**b. Objetivos Específicos**

* Implementar correctamente los algoritmos de búsqueda secuencial y binaria.
* Implementar correctamente los algoritmos de ordenamiento: burbuja, quick sort e inserción.
* Medir con precisión el tiempo de ejecución de cada algoritmo.
* Medir el consumo de memoria de cada algoritmo durante su ejecución.
* Ejecutar todos los algoritmos de manera concurrente mediante hilos.
* Determinar cuál algoritmo completa su tarea en el menor tiempo.
* Visualizar gráficamente los resultados para facilitar la comparación.

**4. Algoritmos de Búsquedas y Ordenamiento**

**a. Búsqueda Secuencial**

La búsqueda secuencial es el algoritmo de búsqueda más simple, que examina cada elemento de la lista secuencialmente hasta encontrar el valor deseado o recorrer toda la lista. Su complejidad temporal es O(n).

**b. Búsqueda Binaria**

Este algoritmo requiere que la lista esté ordenada previamente. Funciona dividiendo repetidamente a la mitad el espacio de búsqueda, lo que le da una complejidad temporal de O(log n). Es mucho más eficiente que la búsqueda secuencial para listas grandes.

**c. Algoritmo de Ordenamiento de la Burbuja**

El método de la burbuja es un algoritmo simple que recorre repetidamente la lista, compara elementos adyacentes y los intercambia si están en el orden incorrecto. Su complejidad temporal es O(n²), lo que lo hace ineficiente para listas grandes.

**d. Quick Sort**

Este algoritmo utiliza la estrategia "divide y vencerás", seleccionando un elemento pivote y particionando el arreglo en subarreglos que contienen elementos menores y mayores que el pivote. Tiene una complejidad promedio de O(n log n), pero puede degradarse a O(n²) en el peor caso.

**e. Método de Inserción**

El algoritmo de inserción construye el arreglo ordenado final un elemento a la vez, tomando cada elemento y colocándolo en su posición correcta. Su complejidad es O(n²), pero es eficiente para conjuntos pequeños o casi ordenados.

**5. Programa desarrollado**

**a. Explicación de su funcionamiento**

El programa está estructurado en una clase principal llamada Algoritmos Comparación que implementa todos los algoritmos requeridos. A continuación, se describen los componentes clave:

1. **Inicialización**: La clase genera un arreglo aleatorio con el tamaño especificado y selecciona un valor aleatorio para las búsquedas.
2. **Implementación de algoritmos**: Cada algoritmo está implementado en su propio método, que:
   * Copia el arreglo original para no modificarlo
   * Inicia el cronómetro y el rastreo de memoria
   * Ejecuta el algoritmo
   * Detiene el cronómetro y el rastreo de memoria
   * Almacena los resultados
3. **Ejecución concurrente**: El método ejecutar\_algoritmos\_concurrentes() utiliza la clase ThreadPoolExecutor para ejecutar todos los algoritmos en paralelo.
4. **Resultados y visualización**: El programa ordena los resultados por tiempo de ejecución, muestra los resultados en la consola y genera gráficas para una comparación visual.

**Conclusiones**

1. La búsqueda binaria resultó ser el algoritmo más rápido, lo cual era esperado dado que tiene una complejidad de O(log n), aunque requiere que el arreglo esté ordenado previamente.
2. Quick Sort fue significativamente más rápido que los otros algoritmos de ordenamiento, confirmando su eficiencia teórica de O(n log n).
3. El algoritmo de ordenamiento burbuja, como era de esperar, fue el más lento debido a su complejidad de O(n²).
4. En términos de memoria, la búsqueda secuencial consumió menos recursos, lo que es lógico dado que no requiere estructuras de datos adicionales ni realiza copias del arreglo.
5. La implementación concurrente permitió ejecutar todos los algoritmos simultáneamente, facilitando una comparación directa en las mismas condiciones de sistema.

Esta aplicación proporciona una herramienta útil para entender de manera práctica las diferencias de rendimiento entre estos algoritmos fundamentales, confirmando las predicciones teóricas sobre su eficiencia.

**Bibliografías:**

<https://es.linkedin.com/advice/1/what-comparison-based-sorting-algorithm-skills-algorithms-paldc?lang=es#:~:text=Un%20algoritmo%20de%20clasificación%20basado%20en%20comparaciones%20es%20un%20tipo,e%20intercambiándolos%20si%20están%20desordenados>.

<https://medium-com.translate.goog/marktech/comparison-sort-algorithms-for-dummies-e621cbff2c3f?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sge#:~:text=Algoritmo%20de%20comparación%20de%20clasificación,Selección%20rápida%20(extra)>

<https://www.alejandro-leyva.com/algoritmos/09_operadores_comparacion/>