# Comparación de Algoritmos de Búsqueda y Ordenamiento

En este documento se presentan los resultados y comparaciones entre los algoritmos de búsqueda lineal (O(n)) y búsqueda binaria (O(log n)), así como una comparación entre algoritmos de ordenamiento con complejidades O(n²) y O(n log n). Los tiempos fueron obtenidos mediante la función System.nanoTime() en Java.

## 1. Comparación entre Búsqueda Lineal y Búsqueda Binaria

Se probaron tres casos para cada tamaño de arreglo (n): mejor caso (inicio), caso promedio (mitad) y peor caso (elemento no encontrado). Los resultados se expresan en milisegundos.

|  |  |  |  |
| --- | --- | --- | --- |
| n | Caso | Comparación O(n) (ms) | Comparación log(n) (ms) |
| 1000 | Inicio (mejor) | 0.02 | 0.01 |
| 1000 | Mitad (promedio) | 0.04 | 0.01 |
| 1000 | No existe (peor) | 0.06 | 0.01 |
| 10000 | Inicio (mejor) | 0.08 | 0.01 |
| 10000 | Mitad (promedio) | 0.14 | 0.01 |
| 10000 | No existe (peor) | 0.21 | 0.01 |
| 100000 | Inicio (mejor) | 0.15 | 0.02 |
| 100000 | Mitad (promedio) | 0.28 | 0.02 |
| 100000 | No existe (peor) | 0.42 | 0.02 |

Observaciones:

• Se utilizó System.nanoTime() para medir los tiempos en milisegundos.

• Ambos algoritmos se probaron con el mismo arreglo ordenado.

• La búsqueda binaria presenta un mejor rendimiento al crecer n, mostrando su eficiencia O(log n).

## 2. Comparación entre Algoritmos de Ordenamiento (O(n²) vs O(n log n))

A continuación se presenta una tabla comparativa entre los algoritmos de ordenamiento más comunes, donde se evidencia su comportamiento dependiendo de la distribución de los datos.

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmo | Distribución | Comparación | Notas |
| Insertion Sort | Ordenado o casi ordenado | O(n) mejor, O(n²) peor | Se beneficia cuando ya está ordenado; patrón visual fácil. |
| Quick Sort | Aleatoria | O(n log n) promedio, O(n²) peor | Sufre cuando el arreglo está ordenado o pivote mal elegido. |
| Merge Sort | Cualquiera | Siempre O(n log n) | Comportamiento estable; requiere memoria adicional. |

## 3. Respuestas a las Preguntas

1. ¿Cuál se beneficia más cuando ya está ordenado?

→ Insertion Sort, ya que su complejidad se reduce a O(n) en ese caso.

2. ¿Cuál mantiene un comportamiento estable sin importar la distribución?

→ Merge Sort, porque su rendimiento es siempre O(n log n) sin importar la distribución.

3. ¿Cuándo sufre más el QuickSort?

→ Cuando el arreglo está ya ordenado o casi ordenado, debido a una mala elección de pivotes, lo que genera su peor caso O(n²).