INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO
DEPARTAMENTO ACADÉMICO DE COMPUTACIÓN.
Estructura de Datos Avanzada.
Gerardo Andres Moguel Rovelo
Tarea No. 6

Metodos de ordenamiento.

Ordenar elementos es un aspecto fundamental en la programación y en la computación en general. Dos métodos comunes para llevar a cabo esta tarea son HeapSort y Quick Sort. Ambos métodos son algoritmos de ordenamiento eficientes que clasifican una lista de elementos en orden ascendente o descendente.

Heap Sort.

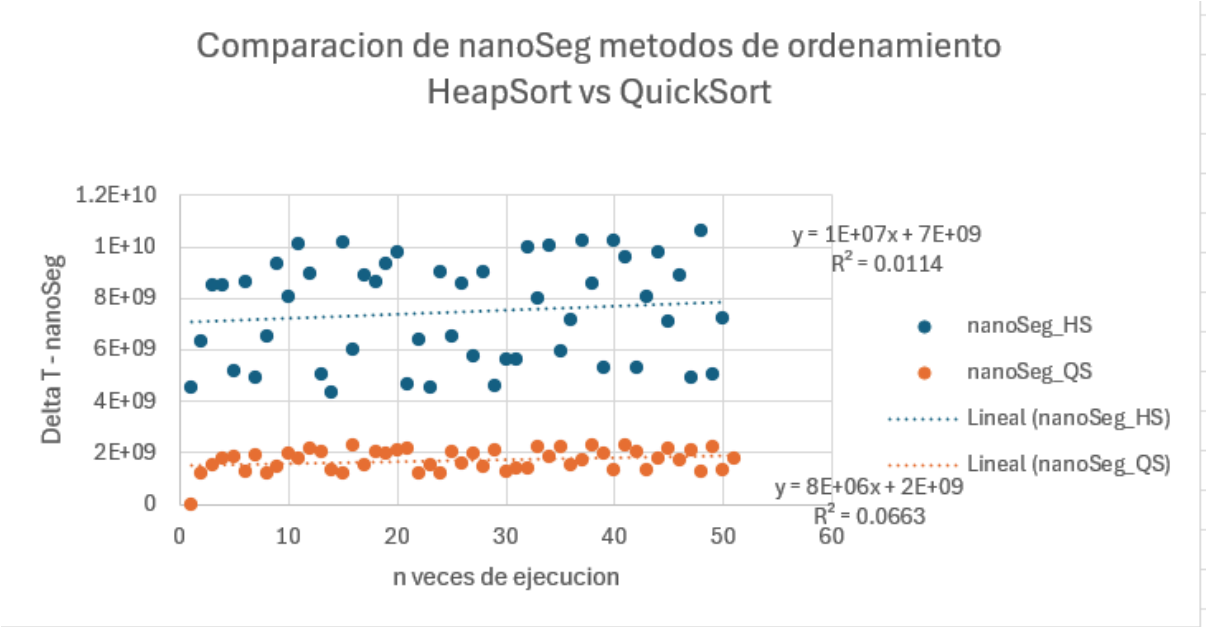
Heap Sort se basa en la estructura de datos conocida como "heap" o montículo. Un heap es un tipo especial de árbol binario en el que cada nodo padre es mayor (o menor) que sus nodos hijos, dependiendo de si es un heap máximo o mínimo, respectivamente. En Heap Sort, primero se construye un heap máximo a partir de los elementos desordenados y luego se extrae el elemento máximo (que es la raíz del heap) y se coloca al final de la lista ordenada. Este proceso se repite hasta que todos los elementos han sido extraídos del heap.

Quick Sort.

Quick Sort es un algoritmo de tipo "divide y vencerás" que elige un elemento pivote de la lista y reorganiza los elementos de modo que los elementos menores que el pivote estén a su izquierda y los elementos mayores estén a su derecha. Luego, el algoritmo se aplica recursivamente a las sub-listas formadas por los elementos menores y mayores que el pivote. El proceso continúa hasta que toda la lista esté ordenada.

Comparación de Métodos.

Para comprender mejor las diferencias entre HeapSort y QuickSort, llevamos a cabo un ejercicio práctico en el que ordenamos listas de diferentes tamaños utilizando ambos métodos. A través de este ejercicio, podemos observar cómo se comportan estos algoritmos en términos de tiempo de ejecución y eficiencia, lo que nos permite comprender mejor sus ventajas y desventajas en diferentes situaciones (aunque el HeapSort es claramente menos eficiente en cualquier situación).



En términos de rendimiento, QuickSort suele ser más rápido que HeapSort en la mayoría de los casos. Esto se debe a que QuickSort tiene un menor número de comparaciones en promedio y su implementación es más sencilla.