Tarea05 - Michael Estrada

Insertion Sort

Descripción: Insertion Sort ordena una lista iterando de izquierda a derecha, tomando un elemento a la vez y colocándolo en su posición correcta respecto a los elementos ya ordenados a su izquierda.

Tiempo de ejecución:

- Peor caso: (O(n^2))
- Mejor caso: (O(n)) (cuando la lista ya está ordenada)
- Caso promedio: (O(n^2))

Ventajas:

- Simple de implementar.
- Eficiente para listas pequeñas o casi ordenadas.
- Estable (mantiene el orden relativo de los elementos iguales).

Desventajas:

- Ineficiente para listas grandes.
- El tiempo de ejecución puede ser lento en listas desordenadas.

MergeSort

Descripción: MergeSort es un algoritmo de tipo divide y vencerás. Divide la lista en mitades hasta que cada sublista contiene un solo elemento, y luego las combina de forma ordenada.

Tiempo de ejecución:

- Peor caso: (O(n \log n))
- Mejor caso: (O(n \log n))
- Caso promedio: (O(n \log n))

Ventajas:

- Eficiente para listas grandes.
- Estable.
- El tiempo de ejecución es predecible y no depende del orden de los elementos.

Desventajas:

- Requiere espacio adicional para las listas temporales.
- Más complicado de implementar que otros algoritmos simples.

HeapSort

Descripción: HeapSort convierte la lista en un montículo (heap) y luego extrae el elemento máximo (o mínimo) repetidamente para construir la lista ordenada.

Tiempo de ejecución:

- Peor caso: (O(n \log n))
- Mejor caso: (O(n \log n))
- Caso promedio: (O(n \log n))

Ventajas:

- Eficiente para listas grandes.
- No requiere espacio adicional significativo.
- El tiempo de ejecución no depende del orden de los elementos.

Desventajas:

- No es estable.
- Puede ser más lento en la práctica que otros algoritmos (O(n \log n)) debido a las operaciones de montículo.

QuickSort

Descripción: QuickSort es un algoritmo de tipo divide y vencerás que selecciona un pivote y reorganiza la lista de forma que todos los elementos menores que el pivote queden a su izquierda y los mayores a su derecha. Luego se aplica recursivamente a las sublistas.

Tiempo de ejecución:

- Peor caso: (O(n^2)) (ocurre cuando el pivote es el elemento más grande o más pequeño repetidamente)
- Mejor caso: (O(n \log n))
- Caso promedio: (O(n \log n))

Ventajas:

- Eficiente en la práctica para listas grandes.
- Requiere espacio adicional mínimo.
- Puede ser muy rápido con una buena elección de pivote.

Desventajas:

- El peor caso puede ser lento.
- No es estable.
- Sensible a la elección del pivote.

BubbleSort

Descripción: BubbleSort ordena una lista iterando repetidamente a través de ella y permutando los elementos adyacentes si están en el orden incorrecto. Este proceso se repite hasta que no se necesiten más permutaciones.

Tiempo de ejecución:

- Peor caso: (O(n^2))
- Mejor caso: (O(n)) (cuando la lista ya está ordenada)
- Caso promedio: (O(n^2))

Ventajas:

- Simple de entender e implementar.
- Detecta rápidamente si la lista está ya ordenada.

Desventajas:

- Ineficiente para listas grandes.
- Generalmente más lento que otros algoritmos (O(n^2)).
- No es adecuado para listas grandes o no ordenadas.