Tarea Corta #2

Andres Murillo Murillo – C15424

Tabla Comparativa de Algoritmos de Ordenamiento

Algoritmo	Mejor Caso	Peor Caso	Caso Promedio	Espacio
Insertion Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(1)$
Selection Sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(1)$
Merge Sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	$O(n)$
Heap Sort	$\Omega(n \log n)$	$O(n \log n)$	$\Theta(n \log n)$	$O(1)$
Counting Sort	$\Omega(n + k)$	$\Theta(n + k)$	$\Theta(n + k)$	$O(n + k)$
Radix Sort	$\Omega(\mathrm{d(n + k)})$	$\Theta(\mathrm{d(n + k)})$	$\Theta(nk)$	$O(n + k)$

1. Insertion Sort (Ordenamiento por Inserción)

- Mejor caso: Eficiente para conjuntos pequeños de datos o casi ordenados, $\Theta(n)$.
- Peor caso y caso promedio: Ineficiente para conjuntos grandes o desordenados, $\Theta(n^2)$.
- Espacio requerido: Mínimo, solo necesita espacio adicional constante, $O(1)$.

2. Selection Sort (Ordenamiento por Selección)

- Mejor, peor y caso promedio: Siempre tiene complejidad cuadrática, $\Theta(n^2)$ independientemente del orden de los datos.
- Espacio requerido: También es mínimo y constante, $O(1)$.

3. Merge Sort (Ordenamiento por Mezcla)

- Caso promedio y peor caso: $\Theta(n \log n)$, donde cada división del arreglo implica pasar por todos los elementos $O(n)$ y se hace una cantidad de divisiones proporcional al logaritmo del número de elementos $(\log n)$.
- Mejor caso: También $\Omega(n \log n)$, ya que incluso en el mejor de los casos se realizan las mismas divisiones y fusiones del arreglo.
- Espacio: $O(n)$, requiere memoria adicional para almacenar las divisiones temporales del arreglo durante la ordenación.

4. Heap Sort (Ordenamiento por montículo)

- Mejor caso: tiene un comportamiento que refleja la eficiencia $\Omega(n \log n)$.
- Peor caso: aun en la situación más desfavorable su tiempo de ejecución es $O(n \log n)$.
- Caso promedio: $\Theta(n \log n)$, que indica que, en el caso promedio, el tiempo de ejecución será proporcional a $n \log n$ también.

5. Counting Sort (Ordenamiento por Conteo)

- Mejor caso: $\Omega(n + k)$, incluso en el mejor de los casos, donde los datos están distribuidos de manera que permiten un conteo rápido.
- Peor caso: $\Theta(n + k)$, lo mismo sucede en el peor caso, siendo k el rango de los números a ordenar.
- Caso promedio: $\Theta(n + k)$, refleja que el tiempo de ejecución es usualmente proporcional a $n + k$.
- Espacio: $O(n + k)$, debido al espacio necesario para los conteos.

6. Radix Sort (Ordenamiento por Residuos)

- Mejor caso: $\Omega(\mathrm{d}(\mathrm{n} + \mathrm{k}))$, aquí dd representa el número de dígitos y k el rango de valores por dígito.
- Peor caso: $\Theta(\mathrm{d}(\mathrm{n} + \mathrm{k}))$, el peor caso tiene la misma complejidad que el mejor caso.
- Caso promedio: $\Theta(nk)$, esto indica que, en promedio, el algoritmo es lineal con respecto al número de dígitos y el rango de valores.
- Espacio: $O(n + k)$, necesita espacio adicional para la ordenación de los "buckets".