Merge Sort: Explicación y Ejemplo

Kafrina Eraldi Serrano Perez

April 2, 2025

Introducción

¿Qué es Merge Sort?

La palabra Merge Sort se podría interpretar como "Ordenamiento por fusión", ya que "merge" significa fusionar y "sort" significa ordenar. Este término describe exactamente cómo funciona el algoritmo: divide la lista en partes más pequeñas, las ordena individualmente y luego las fusiona de forma ordenada.

- Algoritmo de ordenación eficiente, general y basado en la comparación.
- Se fundamenta en la estrategia de *divide y vencerás*, dividiendo un conjunto de datos en mitades, ordenándolas de manera recursiva y luego fusionándolas en una lista ordenada.

Explicación de la Notación Big-O

La notación Big-O, representada como $O(f(n))$, se utiliza para describir el crecimiento de la cantidad de operaciones necesarias en un algoritmo en función del tamaño de la entrada n.

Concepto de Big-O

- $O(1)$ - Tiempo constante: El tiempo de ejecución no cambia sin importar el tamaño de la entrada. Ejemplo: acceder a un elemento de un arreglo.
- $O(n)$ - Tiempo lineal: El tiempo de ejecución crece proporcionalmente a n. Ejemplo: recorrer una lista.
- $O(n^2)$ - Tiempo cuadrático: Se usa en algoritmos menos eficientes, como Bubble Sort.
- $O(n \log n)$ - Tiempo subcuadrático: Más rápido que $O(n^2)$, pero más lento que $O(n)$, típico en algoritmos eficientes de ordenamiento como Megesort y Merge Sort.

¿Por qué Merge Sort es $O(n \log n)$?

1. **División de la lista**: Se divide la lista en mitades hasta que cada sublista contiene un solo elemento. Como cada división reduce el tamaño de la lista a la mitad, el número total de divisiones es $O(\log n)$.
2. **Fusión de las sublistas**: Una vez divididas, las sublistas se combinan en orden. Cada nivel de la fusión requiere recorrer todos los elementos, lo que toma $O(n)$ operaciones.

Como hay $O(\log n)$ niveles de división y cada nivel requiere $O(n)$ operaciones, la complejidad total es:

$$O(n) \times O(\log n) = O(n \log n)$$

Características de Merge Sort

- **Eficiencia**: Complejidad $O(n \log n)$ en el peor caso. En comparación con algoritmos como Bubble Sort o Insertion Sort, que tienen una complejidad de $O(n^2)$ en el peor caso, Megesort es mucho más eficiente en listas grandes.
- **Ordenación estable**: Conserva el orden relativo de los elementos iguales.
- **Paralelización**: Es altamente paralelizable, ya que las diferentes partes del conjunto de datos pueden ordenarse simultáneamente antes de combinarse.

Historia de Merge Sort

El algoritmo Merge Sort fue inventado por John von Neumann en 1945. Un análisis detallado de su implementación fue publicado por Goldstine y von Neumann en 1948.

Funcionamiento del Algoritmo

El algoritmo sigue los siguientes pasos:

1. Dividir el conjunto de datos en mitades.
2. Ordenar recursivamente cada mitad.
3. Fusionar las mitades ordenadas en una sola lista ordenada.

Ejemplo Manual de Ejecución

Supongamos que tenemos el siguiente arreglo desordenado:

$$[12, 8, 9, 3, 11, 5, 4]$$

Dividimos el arreglo en mitades sucesivas:

$$[12, 8, 9][3, 11, 5, 4]$$

$$[12][8, 9][3, 11][5, 4]$$

$$[12][8][9][3][11][5][4]$$

Luego comenzamos la fusión comparando los elementos:

$$[8, 9][12][3, 11][4, 5]$$

$$[8, 9, 12][3, 4, 5, 11]$$

$$[3, 4, 5, 8, 9, 11, 12]$$

El proceso de división y combinación se realiza de forma recursiva, asegurando que cada subarreglo fusionado mantenga el orden correcto.

Resultado final:

$$[3, 4, 5, 8, 9, 11, 12]$$

Implementación en C++

Entrada: Una vector *arr* de tamaño n

if $n == 0$ **then**

    Retornar: el vector está vacía, no hay elementos para ordenar

else if $n == 1$ **then**

    Retornar: el vector ya está ordenada

else

    Inicializar $izquierda \leftarrow 0$

    Inicializar $derecha \leftarrow n - 1$

    Calcular $centro \leftarrow \frac{\text{izquierda+derecha}}{2}$

    Llamar a mergeSort(arr, izquierda, centro) ▷ Ordenar recursivamente la primera mitad

    Llamar a mergeSort(arr, centro + 1, derecha) ▷ Ordenar recursivamente la segunda mitad

    Llamar a merge(arr, izquierda, centro, derecha) ▷ Fusionar las mitades ordenadas

end if

Retornar: el vector está ahora ordenada

Implementación en Python

Conclusión

Merge Sort es un algoritmo eficiente y estable, ideal para ordenar grandes volúmenes de datos y ampliamente utilizado en informática debido a su capacidad de paralelización y consistencia en el rendimiento.