Método Merge Sort

Andrés Salcedo Vera[1], Eder David Barrero Castañeda[2], Sharay Gabriela Farias Quemba[3], Yorinder Lucia Restrepo Giraldo
Fundacion Universitaria Konrad Lorenz
[1]jaimea.salcedov@konradlorenz.com, [2]ederd.barreroc@konradlorenz.edu.co [3]sharayg.fariasq@konradlorenz.edu.co
[4]yorinderl.restrepog@konradlorenz.edu.co

***Resumen*—El método de ordenamiento Merge Sort es una técnica eficaz y ampliamente utilizada para organizar datos de manera eficiente. En este artículo, exploraremos en detalle este algoritmo de ordenamiento y su implementación en Python. Comenzaremos analizando su funcionamiento, luego profundizaremos en el código Python que lo implementa y discutiremos su complejidad temporal (Big O).**

I. Introducción

Ordenar elementos en una lista es una tarea común en programación y resuelve una variedad de problemas. Entre los muchos algoritmos de ordenamiento disponibles, Merge Sort se destaca por su eficiencia y simplicidad conceptual. Desarrollado por John von Neumann en 1945, este algoritmo divide una lista en dos mitades, ordena cada mitad de manera recursiva y luego combina las mitades ordenadas para obtener una lista completamente ordenada.

II. Análisis de Python

Antes de sumergirnos en la explicación detallada de Merge Sort, es importante destacar que Python proporciona una amplia gama de bibliotecas y funciones integradas para ordenar listas y secuencias. Sin embargo, comprender algoritmos de ordenamiento como Merge Sort es esencial para programadores, ya que proporciona una base sólida para abordar problemas de ordenamiento más complejos.

II-A. Explicación de Merge Sort

En el script de Python, existen varias formas de medir el tiempo de ejecución. Una de ellas consiste en ejecutar todo el script en un cuaderno de Jupyter, aunque esta no es la opción más eficiente, proporciona una estimación del tiempo que lleva su ejecución. Otra alternativa es utilizar la biblioteca time en Python y agregarla al script para obtener un contador que registre el tiempo desde el inicio hasta el final de la ejecución.

El algoritmo Merge Sort se basa en el enfoque "divide y conquista". Su funcionamiento se puede dividir en tres pasos principales:

```python
def merge_sort(arr):
    if len(arr) > 1:
        # Split the array into two halves
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        # Recursive calls to sort both halves
        merge_sort(left_half)
        merge_sort(right_half)

        # Initialize indices for traversing the two halves and the main array
        i = j = k = 0

        # Merge the two halves back into the original array
        while i < len(left_half) and j < len(right_half):
            if left_half[i] <= right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        # Check if any elements were left in either of the halves
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

    return arr
```

Figura 1. Algoritmo

II-B. Dividir

Divide la lista no ordenada en dos mitades de tamaño aproximadamente igual. Esto se logra mediante la búsqueda del punto medio de la lista.

II-C. Conquistar

Ordena de manera recursiva ambas mitades. Esto implica dividir cada mitad en sub-mitades, ordenarlas y fusionarlas.

II-D. Combinar

Fusiona las mitades ordenadas para formar una lista completamente ordenada. Esto se hace comparando elementos de ambas mitades y colocándolos en el orden correcto en una nueva lista.

El proceso de división y combinación continúa hasta que se haya ordenado toda la lista.

III. BIG O

Una de las ventajas clave del Merge Sort es su eficiencia en términos de tiempo. El tiempo de ejecución de Merge Sort es O(n log n), lo que lo hace muy eficiente incluso para listas muy grandes. Esto se debe a su enfoque "divide y conquista", que minimiza la cantidad de comparaciones necesarias.

En conclusión, el método de ordenamiento Merge Sort es un algoritmo poderoso y eficiente que puede manejar grandes conjuntos de datos de manera efectiva. Su implementación en Python es relativamente sencilla y proporciona una comprensión valiosa de los algoritmos de ordenamiento. Al comprender cómo funciona Merge Sort y su complejidad temporal, los programadores pueden tomar decisiones informadas sobre cuándo utilizar este algoritmo en sus aplicaciones.

IV. BIBLIOGRAFIA

OpenAI. (2021). ChatGPT: A large language model.

Andrés Salcedo Vera. (2023). Método Merge Sort. Eder David Barrero Castañeda. (2023). Método Merge Sort. Sharay Gabriela Farias Quemba. (2023). Método Merge Sort. Yorinder Lucia Restrepo Giraldo. (2023). Método Merge Sort.