Taller 6

Jose David Ramirez Beltran - 506222723

24 de septiembre de 2023

1. Introducción

En este caso se tratara un metodo de ordenamiento merge sort, el cual en el ejemplo mas sencillo toma parejas de numeros y dependiendo de cual sea mayor los va ordenando en orden hasta que todos terminen ordenados. para ello, se usa este metodo si:

Si la lista es de tamaño pequeño (vacía o de tamaño 1), ya está ordenada y no hay opción para hacer nada; si no se deberia hacer lo siguiente

1. Dividir la lista en dos sublistas de aproximadamente el mismo tamaño.

2. Ordenar cada una de esas dos sublistas.

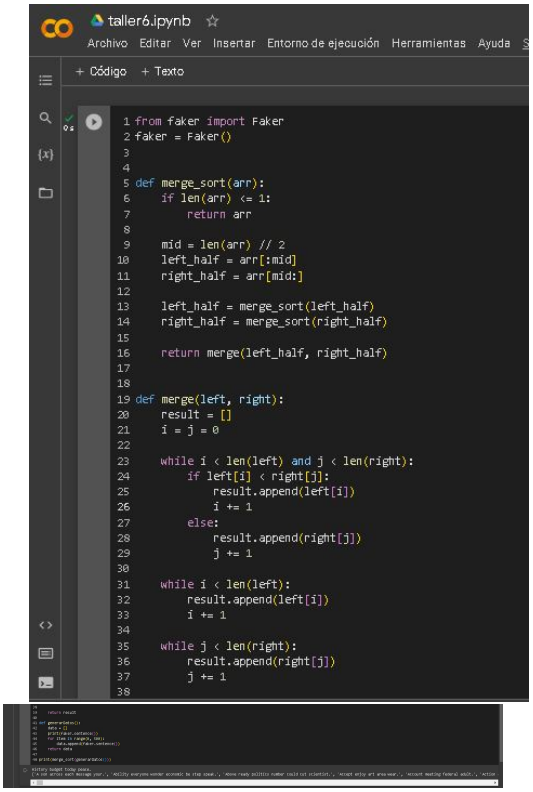
3. Después de ordenar las dos sublistas, intercalarlas (mergerlas) de manera cuidadosa.

Por ejemplo, si la lista inicial es $[6, 7, -1, 0, 5, 2, 3, 8]$, debemos ordenar nuevamente $[6, 7, -1, 0]$ y $[5, 2, 3, 8]$ para obtener $[-1, 0, 6, 7]$ y $[2, 3, 5, 8]$ respectivamente.

La solución buscada se puede obtener intercalando ordenadamente las dos listas ordenadas: $[-1, 0, 1]$.

2. Desarrollo

Para llevar el desarrollo del algoritmo, se puede plantear de la siguiente manera:



3. Funcionamiento

Este código utiliza el algoritmo de ordenamiento Merge Sort y la biblioteca Faker para crear una lista de 500 oraciones inventadas. El algoritmo divide la lista en sublistas más pequeñas en la función $\text{merge}_sorthastaquecadasublistatengaunooningúnelemento.$

Luego, se usa la función merge para combinar ordenadamente las sublistas ya ordenadas y la función de mezcla toma dos sublistas ordenadas y las fusiona en una sola lista ordenada, comparando los elementos de ambas sublistas y seleccionando el más pequeño en cada paso. Y ya por ultimo, el re-

sultado es una lista de oraciones ficticias ordenadas alfabéticamente.

4. Coclusiones

En conclusión, a medida del desarrollo del ejercicio se nota la eficiencia del Merge Sort que es un algoritmo de ordenamiento efectivo que divide una lista en sublistas más pequeñas, las ordena de forma recursiva y luego las combina en una lista ordenada. Tambien esta técnica de "dividir y conquistar"que se evidencio en varios articulos investigados, tiene un tiempo de ejecución promedio y un caso peor de O(n log n), lo que la hace adecuada para manejar conjuntos de datos grandes.

Tambien se usa la biblioteca Faker para generar datos ficticios, es útil para pruebas y simulaciones con datos de prueba, y es una forma conveniente de generar una lista de elementos para probar el algoritmo de ordenamiento.

5. Bibliografia

https://github.com

http://www.cs.uns.edu.ar/ mlg/ipoo/downloads/Presentaciones

https://colab.research.google.com/drive/$1_aDpkHl1E7tKrR_ouYhFgTn1Qc0dwt1sscrollTo = 0BD68Z\_kWbS$