Alexis Sánchez 181074 Estructuras de Datos Avanzadas

Orden Lexicográfico vs Merge Sort

Introducción:

El Trie es una estructura de datos en forma de árbol que sirve para la rápida recuperación de información mediante el almacenamiento de un conjunto de llaves que forman parte de un alfabeto. La palabra Trie viene de *Re**TRIE**val*, lo cual hace referencia a la recuperación de información. Cada una de las llaves es un símbolo que se guarda en las hojas del árbol que a su vez esta conectadas para guiar las búsquedas y navegaciones en el árbol.

Descripción del problema:

Mediante la implementación de la estructuras de datos Trie, se pretende hacer primer la inserción de una determinada cantidad de datos para luego ejecutar una operación que recorra el árbol e ir mostrando las llaves de manera ordenada. Para lograr este objetivo, primer es necesario hacer la implementación de una inserción sencilla y luego al árbol poderle aplicar una función que arroje las llaves ordenadas. Una vez que se tengan estos métodos, se debe comparar el tiempo de ejecución del ordenamiento lexografico contra el tiempo de ejecución de la misma tarea pero con el algoritmo de ordenamiento Merge Sort. En este experimento, usaremos un conjunto de 300,00 palabras que iremos usando de manera paulatina. Primero se usarán pequeñas partes del conjunto empezando por 100 datos y seguiremos hasta los 300,00 datos.

Desarrollo:

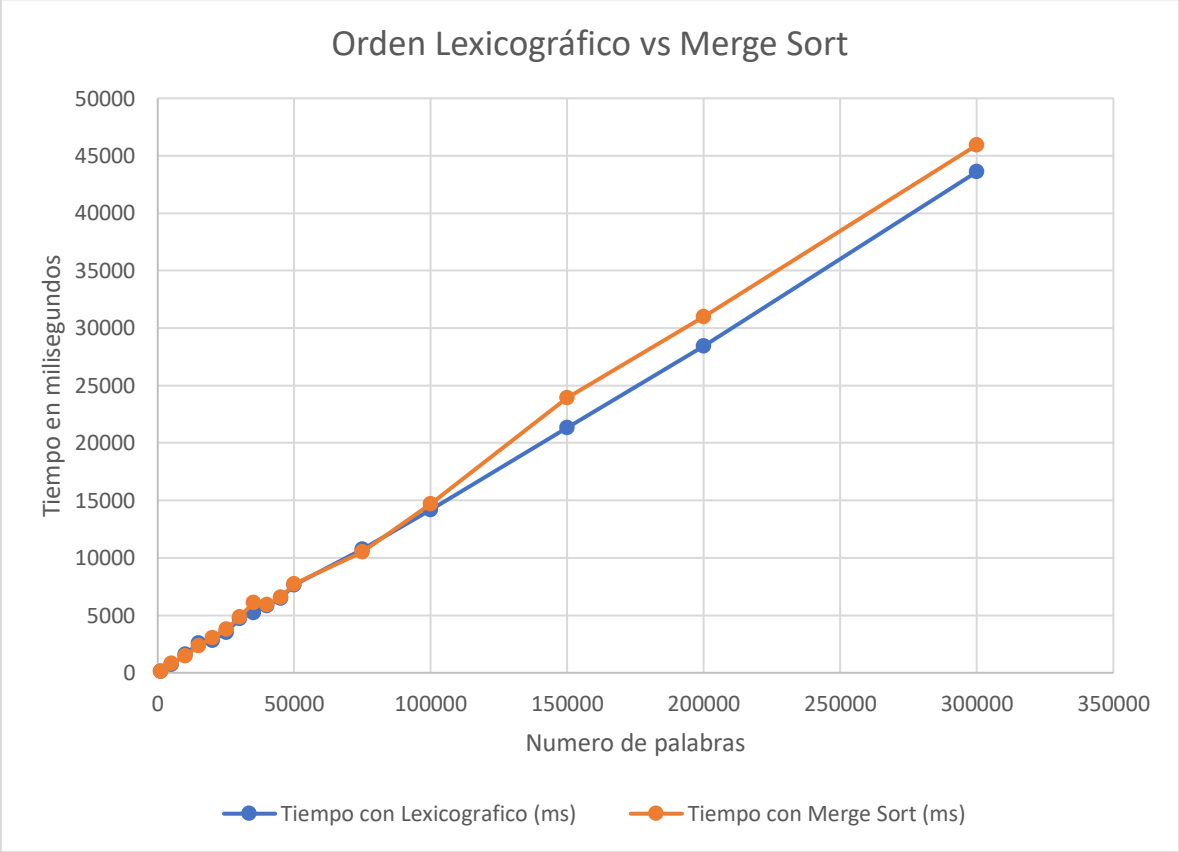
Para realizar la inserción, decidí optar por un método iterativo que reciba como único parámetro la llave a insertar. Se crea un nodo que inicialmente toma los atributos de la raíz y se toma la longitud de la llave. Se hace un ciclo de tipo for que irá recorriendo cada uno de los niveles del árbol y en cada nivel insertando el carácter correspondiente de acuerdo con su posición en el arreglo de símbolos. Una vez hecha la inserción de toda la llave, se pone verdadero en el fin de la palabra para indicar que ahí termina la palabra.

Para realizar el ordenamiento lexografico, hago uso de un método recursivo que regresa vacío pues solo se encarga de imprimir las llaves en orden. Si el nodo recibido es nulo, no regresa nada. En caso contrario, entra en un ciclo que recorre todo el arreglo de símbolos del alfabeto y va imprimiendo las llaves al mismo tiempo que se vuelve a llamar a si mismo para seguir realizando la operación.

Una vez implementadas estas dos operaciones, se pasa a leer el archivo con el número de datos deseados que no sobrepasen el total de datos en el archivo. Almanceno los datos en forma de cadenas en un arreglo. Como el archivo de palabras esta ordenado, realizado una operación que toma los elementos del nuevo arreglo y los mezcla para tener datos sin ordenar. Luego, paso a insertar cada elemento del arreglo dentro del Trie. Esta inserción puede resultar complicada para palabras que tienen caracteres especiales, es por eso que este experimento se reducirá a usar palabras sin espacios ni caracteres. Finalmente, se puede mandar llamar a la función que realizar el ordenamiento mientras que se miden los milisegundos que tarda en terminar el proceso. Este último paso también se realiza con el Merge sort para evaluar las diferencias.

Resultados:

Numero de palabras	Tiempo con Lexicográfico (ms)	Tiempo con Merge Sort (ms)
1000	105	132
5000	724	799
10000	1584	1437
15000	2585	2355
20000	2802	3017
25000	3495	3762
30000	4703	4859
35000	5205	6116
40000	5808	5930
45000	6475	6547
50000	7625	7730
75000	10744	10509
100000	14193	14694
150000	21341	23920
200000	28409	30958
300000	43601	45933



Análisis de Resultados:

En los datos arrojados como resultados, es posible observar que el método de ordenamiento resulta ser un tanto más rápido que el Merge sort para realizar la tarea. Sin embargo, se puede notar que entre los 40,000 y 75,000 datos la diferencia es muy poca. Es justo a los 75,000 datos que el Merge sort tiene un mejor tiempo que el ordenamiento lexicográfico. Hacia el final del experimento con los 200,000 y 300,000 datos se logra ver que el ordenamiento fue el más rápido en estas pruebas. En la grafica presentada, se puede ver como en pequeñas cantidades de datos el desempeño es similar, aunque a la mayor cantidad de datos se empieza a notar la diferencia.

Conclusiones:

Esta función de ordenamiento lexicográfico esta diseñada para funcionar con cadenas de caracteres e incluso valores numéricos que sea convertir a cadenas de texto. Tomando en cuenta esta consideración, no funcionaría para conjuntos de datos que representen objetos pues se perdería la eficiencia de este algoritmo. De tal forma que este ordenamiento trabaja con objeto primitivos mas no con objetos de complejidad mayor. Los algoritmos de ordenamiento visto anteriormente funcionan para todo tipo de datos y la función implementada en este experimento es util para cuestiones más sencillas como seria evaluar cadenas alfanuméricas.

Como se ha podido ver en los resultados, el ordenamiento implementado tiene un mayor rendimiento que el Merge Sort aunque la diferencias se empiezan a notar cuando la cantidad de datos es muy grande. El ordenamiento lexicográfico es muy util para ordenar cadenas alfanuméricas si se implementa de manera correcta aunque no para otro tipo de objetos. Por lo tanto, el uso de otros algoritmos de ordenamiento se encargarán de este de situaciones.