Luis Cano Irigoyen - A00827178

Act1.3 Reflexión

A lo largo del bloque estuvimos trabajando con algoritmos de búsqueda y ordenamiento, en los cuales yo siempre use ejemplos de vectores con entre 4 a 20 números/elementos para ver de manera sencilla como funcionaban. Estas son listas las cuales sencillamente podría haber ordenado a mano, o con la vista encontrado el elemento buscado en cuestión de segundos. Pero conforme las listas se vuelven más grandes, es mayor la necesidad de contar con un algoritmo de ordenamiento/búsqueda; y no solo se vuelve necesario contar con uno de estos algoritmos, sino se requiere de conocer cómo funciona cada uno y su complejidad para determinar cuál funcionaría mejor con los datos que se tienen. En el caso de esta situación problema se contaba con más de 16mil líneas de datos, y en una primera instancia en donde como ejemplo utilizaba sólo 50 líneas de datos de la bitácora, mientras que probaba mis códigos de obtención de datos, ordenamiento, búsqueda e impresión de datos, contaba con un algoritmo que ordenaba a aquellos de manera iterativa y uno que realizaba una búsqueda binaria de los datos. Como era de esperarse, con los 50 datos funcionó bien y completé mi código; pero al cambiar de mi archivo de prueba de 50 datos al de 16807, mi código comenzó a requerir de varios minutos para lograr ordenar la bitácora, mientras que la búsqueda seguía completándose de manera casi instantánea. Fue entonces que cambié mi algoritmo de ordenamiento al de Merge, siendo uno que utilizaba recursividad. Utilizar algoritmos de ordenamiento y búsqueda con complejidades de O(n log n) y O(log n) volvió posible el realizar sus procesos en mi código de manera notablemente rápida.

En este caso, en el cual se cuenta con 16807 líneas de datos, el utilizar algoritmos de ordenamiento iterativos, cómo el que yo utilicé de ordenamiento al inicio de complejidad $O(n^2)$, termina siendo bastante impráctico. Si realizamos el cálculo de la complejidad obtenemos: $16807^2 = 282475249$. Por esto, para el ordenamiento fue necesario utilizar algoritmos recursivos. Ordena Merge es uno que va dividiendo la lista en dos sublistas (mitades), y luego recursivamente las ordena aplicando el ordenamiento por merge, realizando dicho merge hasta que las sublistas queden ordenadas en una sola. En la complejidad de este algoritmo de búsqueda obtenemos O(n log n), y si realizamos el cálculo obtendríamos $16807 \cdot \log(16807) = 71017.8$, siendo marginalmente menor que el de ordenamiento iterativo.

De igual manera, en cuanto a los algoritmos de búsqueda, el realizar una búsqueda secuencial me habría llevado una complejidad de O(n), la cual claramente no es tan alta como la complejidad del ordenamiento iterativo, pero al realizar la búsqueda de manera binaria, dividiendo la lista ordenada en mitades hasta encontrar el elemento deseado, obtuve una complejidad de O(log n), y log 16807 = 4.2255 es bastante menor a 16807.

En situaciones problemas de esta naturaleza, es importante conocer el comportamiento y la eficiencia de estos diferentes algoritmos para los diferentes tipos de datos; habrá unos que funcionen mejor con datos semiordenados, otros que ordenen mejor listas completamente aleatorias, así como es importante conocer qué algoritmos funcionan mejor con pequeñas, medianas y cantidades masivas de datos. Esto con el objetivo de realizar soluciones eficaces y óptimas para los problemas.