ReflexAct1.3
Eugenio Peña García A01177348

Tras realizar la evidencia de competencia 1.3 (Algoritmos Fundamentales) y prestar atención en mis clases de programación de estructuras de datos y algoritmos fundamentales, aprendí mucho sobre los algoritmos de ordenamiento y búsqueda. Mi primer gran descubrimiento fue la cantidad de maneras que hay para ordenar una lista de datos. En las últimas clases vimos 6 distintos tipos de algoritmos para ordenar una lista, los 6 funcionando para cualquier tipo de dato (en casos especiales utilizando sobrecarga de operadores). No solo me di cuenta de que algunos tipos de algoritmos son más eficientes que otros, sino que también llegué a la conclusión de que es muy situacional el decidir cual algoritmo es “mejor”. Esto es debido a que en distintos casos la lista está originalmente ordenada de manera en que es simplemente más rápido utilizar algunos métodos que otros.

Estos métodos suelen tener grandes diferencias en sus tiempos de ejecución. Estas diferencias parecen insignificantes para nosotros, pues van de los milisegundos a incluso unidades más pequeños. No obstante, esto es debido a que las listas con las que lo probamos eran bastante pequeñas, pero en casos más grandes las diferencias podrían llegar a ser más significativas. El hecho de que un código tome más tiempo y requiera de más capacidad de procesamiento no es cualquier cosa. Puede llegar a complicar todo, e incluso en los casos de computadoras que no tengan suficiente poder de procesamiento, ser la diferencia entre si un programa corre o no de manera adecuada.

Los algoritmos pueden llegar a ser bastante sencillos, o bastante complejos. El más básico es el de ordenamiento por intercambio, que simplemente compara el primer dato con los siguientes, y luego avanza con el siguiente dato. Al tener dos ciclos for, este algoritmo tiene una complejidad de O(n^2). Por otro lado, tenemos el MergeSort, el cual es uno de los más complejos que vimos para realizar, pero es bastante rápido y eficiente, y su complejidad es de O(n * log n).

Esto todo influye para la velocidad de los programas y algoritmos, por lo que entre más eficiente más rápido, y menor capacidad computacional requerida. Por otro lado, también se encuentran los algoritmos de búsqueda, de los cuales vimos dos: Búsqueda binaria y búsqueda secuencial. La binaria tiene una complejidad de O(log n), y funciona de manera que toma el valor del centro de la lista, y utiliza un valor inferior y superior, comenzando con el 0 y el último número respectivamente. Estos límites los va moviendo, dependiendo de si el promedio es mayor o menor al dato buscado. Por otro lado, la secuencial tiene la complejidad de O(n), y es bastante simple, pues solo busca uno por uno de izquierda a derecha el dato. Esto puede parecer como algo simple, y por ende bueno, sin embargo, en casos de listas largas, si el dato que buscas está hasta el final, este método puede tardar demasiado.

En conclusión, el uso de los algoritmos más eficientes para cada caso siempre es importante. Esto es debido a que nos ahorra tiempo y recursos. Esto significa que nuestros códigos pueden ser más eficientes, pues los podrán correr las máquinas de manera más sencilla, y serán más disponibles a distintos aparatos en general.