Adrian Gonzalez Saldivar A00827845

Reflexión

Cada algoritmo de ordenamiento y búsqueda cuenta con una complejidad y eficiencia que depende de un factor diferente al otro. En cada algoritmo de ordenamiento y búsqueda se presentan diferentes niveles de complejidad: el mejor caso, el peor caso y en promedio. Los algoritmos comparten el resultado del nivel de complejidad del mejor caso, en este nivel solo se hace una comparación, esto quiere decir que en todos los algoritmos de búsqueda y ordenamiento en el mejor caso solo ocurre una comparación. La velocidad de ejecución, que es la eficiencia, depende del tamaño del arreglo, sin embargo este tamaño puede cambiar en relación del algoritmo, puede ser lineal, cuadrático, logarítmico. En esta actividad se utilizaron varios algoritmos de ordenamiento y búsqueda, algunos de estos fueron vistos en clase como el quicksort con su función conjunta y la búsqueda binaria. El quicksort tiene complejidad en el mejor caso logarítmica de n (O(nlogn)) y en su peor caso, el cual es el que se usa para representar la función, O(n^2) , al utilizar esta función se requiere una función adicional, su complejidad es lineal O(n), por otro lado la función de búsqueda binaria tiene una complejidad logarítmica O(log n). Al usar estos algoritmos de ordenamiento y búsqueda nos ahorramos tiempo al momento de correr el programa, esto se debe a la eficiencia (velocidad) que se está aplicando al arreglo.

El algoritmo de quicksort fue utilizado para ordenar los datos que se encontraban en el archivo “bitacora.txt.” y la función búsqueda binaria se utilizó para encontrar los datos que pedimos, como el mes inicial, día inicial, mes final y dia final, que son los rangos de búsqueda, un punto importante que me di cuenta en esta actividad y en la pasada es que para que la búsqueda binaria funcione correctamente ocupa trabajar con un arreglo previamente ordenado.Otras dos funciones fueron usadas, ambas tienen una complejidad (O(1)) y cumplen un trabajo similar, ambos trabajan afectando los valores de la fecha.

En esta situación problema nos encontramos con varios pequeños problemas, uno de ellos se centra en el uso del “struct” pero esto fue solamente al inicio cuando empezamos a investigar el uso, y las diferentes maneras de establecer sus variables, ya sea después de la llave “}” y antes del punto y coma “;” o declararlo en el main. Al principio no entendía cómo usarlo hasta que empecé a verlo como una clase o un grupo de elementos bajo el mismo nombre. Otro problema que en particular me enfrente yo fue más la implementación de leer el archivo que nada, esto se debe porque aunque me acordaba/sabía de varias líneas de código que son clave para el algoritmo, algunas las terminaba cambiando ya sea por declaración o el orden en donde los colocaba. Un problema que teníamos en esta parte fue al desplegar en la consola los datos buscados ya que en algunos casos salía un dato que tenía otra fecha fuera del rango del que se pedía, esto se arreglo cambiando el límite final del ciclo for. Otro problema que tuve su entender en su totalidad como funciona la complejidad del Quicksort, no fue hasta que mi compañero de equipo me explico la razón por el cual se establece como cuadrática, este es porque dentro de la función se llama a sí misma dos veces, aunque con una diferencia de parámetros, cuenta como una repetición de llamada, lo cual lo hace cuadrática.

Referencias:

Teddy Alfaro Olave. (S.F). Algoritmos de Búsqueda y Ordenamiento. 12/09/2020, de Departamento de Informática Universidad Técnica Federico Santa María Sitio web: https://www.inf.utfsm.cl/~noell/IWI-131-p1/Tema8b.pdf