Tecnológico de Monterrey, Campus Monterrey
Programación de estructuras de datos y algoritmos fundamentales
Rodrigo Llaguno Cárdenas, A01198067
Grupo 206

Reflexión Act 1.3

Durante el desarrollo de esta actividad, nos encontramos con distintos desafíos tanto al principio de la creación del programa como al final. Empezamos a idear la manera en la que podíamos crear este programa de forma que se ordenen los datos de la bitácora de forma eficiente. Para esto teníamos claro el orden de los algoritmos que debíamos de seguir para hacerlo. Primero teníamos que pedir al usuario un rango de fechas. Este rango de fechas debemos pasarlo a algún tipo de dato con el cual podamos comparar las demás fechas de tipo entero. Después teníamos que leer el archivo de la bitácora y ordenarla. Finalmente debíamos buscar las fechas dentro de los rangos e imprimirlas en el archivo de salida.

Decidimos crear un struct para organizar la información de cada renglón de la bitácora. Basamos la mayoría de nuestro programa en este concepto, pero no contábamos con un problema con el cual nos íbamos a topar más adelante. Para tener un dato con el cual basar la comparación, creamos un mapa el cual dependiendo del mes y el día, regresa los días totales que han transcurrido. Las demás funciones simplemente leen la bitácora, escriben en un archivo nuevo y separan los strings e ints de una línea de texto.

Los siguientes dos algoritmos son un poco más complicados. Empezamos con el algoritmo de búsqueda, sequentialSearch el cual también es conocido como linearSearch ya que empieza del inicio y va valor por valor hasta llegar al final para encontrar el valor. Creamos este algoritmo de manera iterativa debido a su simplicidad. Este algoritmo tiene una complejidad de O(n) (GeeksforGeeks, 2022). Después implementamos el algoritmo de ordenamiento quickSort. Este algoritmo funciona de manera recursiva. Se asigna un elemento como pivote y parte nuestro vector en base a dicho pivote. Después se llama otra función donde se le asigna la posición correcta al valor. Este algoritmo tiene una complejidad de O(n^2) aunque normalmente es más rápido debido a que se puede modificar el valor del pivote (Jain, 2022). Cabe mencionar que en nuestro programa terminamos utilizando la función sort() de la librería <algorithm> ya que debido al tipo de dato que utilizamos para nuestra información

(struct dato), no pudimos implementar de manera directa este algoritmo porque no podíamos acceder a esta información. Tuvimos que utilizar esta función externa la cual utiliza internamente el mismo algoritmo, quickSort (Great Learning Team, 2021).

Esta actividad me permitió trabajar de manera aplicada lo que habíamos estado aprendiendo en clase. Comprendí la utilidad de estos algoritmos ya que pude trabajar con información real y crear un programa que pudiera manipular una cantidad de datos inmensa, el cual tardaría una cantidad muy grande de tiempo de hacer manualmente. Trabajé por primera vez con la lectura y escritura de archivos y amplié mis conocimientos en C++ al tener que trabajar con conceptos como mapas, vectores, structs, apuntadores, entre otras cosas para poder completar el objetivo. Disfruté del trabajo en equipo para la resolución de problemas y lluvia de ideas de cómo íbamos a resolver la actividad.

Referencias:

GeeksforGeeks. (2022). *Linear Search Algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/linear-search/

Great Learning Team. (2021, February 26). *Sort Function in C++ | C++ Algorithm Sort*. Great Learning. https://www.mygreatlearning.com/blog/sort-function-in-cpp/

Jain, P. (2022). *QuickSort*. GeeksforGeeks. https://www.geeksforgeeks.org/quick-sort/