Abraham De Alba Franco A01634133

Act 1.3: Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

La complejidad computacional es altamente importante para analizar y buscar minimizar los recursos en una situación problema como la que se presenta en este caso. Debido a la cantidad de datos a interpretar, manejar y organizar, es de muy alta importancia mantener baja la complejidad de los algoritmos utilizados con la finalidad de que este sea lo más eficiente posible y consuma la mínima cantidad de recursos computacionales posibles. En nuestro caso se interpretaron alrededor de 16800 datos en la situación problema, y la máxima complejidad posible tiene un comportamiento asintótico de complejidad de O(nlogn) por lo cual se puede considerar que es bastante buena, sin embargo, si se utilizaran algoritmos con comportamientos superiores, como n^2 o 2^n, la complejidad se dispararía si se incrementara la cantidad de datos utilizados junto con el tiempo de solución del programa.

En nuestra solución del código utilizamos el algoritmo de búsqueda de Merge-Sort ya que este tiene una complejidad de tan solo O(nlogn), que comparándolo con otros algoritmos de búsqueda vistos en clase, que tienen una complejidad de al menos n^2, entonces Merge-Sort resulta altamente competente para el manejo de grandes cantidades de información.

El algoritmo de búsqueda utilizado en nuestra situación problema fue la búsqueda secuencial, ya que a nuestro criterio este presenta mayor facilidad para este caso que consiste en buscar la fecha más cercana a la ingresada por el usuario dentro del rango de datos que se tienen establecidos en la bitácora del archivo de texto. Este algoritmo presenta una complejidad de O(n).

Abraham De Alba Franco A01634133

Complejidad Computacional. (2018). *Numerentur*. https://numerentur.org/complejidad-computacional/