Alejandro Lizárraga Vizcarra
A01284610

Reflexión Evidencia1

Con este proyecto pude aprender sobre la aplicación de algoritmos de ordenamiento y de búsqueda. Es importante contar con algoritmos eficientes para poder trabajar con los datos. La eficiencia de estos algoritmos se vuelve aún más importante al trabajar con una cantidad sustancial de datos. Un proceso que se tarda un cuarto de segundo puede que no sea notorio cuando se trabaja con menos de 10 datos, pero al utilizar estos procesos en casos reales con millones de datos, conseguir la información con algoritmos ineficientes puede resultar demasiado tardada.

Sin embargo, también es importante considerar las limitaciones computacionales que tienen los sistemas instalados. Fuera del tiempo computacional que se tarda, se debe de considerar que ciertos sistemas tienen a su disponibilidad memoria más rápida y a mayores cantidades mientras que otros sistemas pueden tener mejor procesamiento de CPU. Dependiendo de si se trabajan con comparaciones y procesos aritméticos o con instanciaciones de copias de datos, diferentes algoritmos pueden ser mejor aplicados en diferentes situaciones.

Para este proyecto en la parte de ordenamiento se utilizó el Quicksort. Esto es debido a que este es algoritmo más rápido que corre en mi sistema y no tenemos limitaciones de memoria o uso de CPU que nos pudiera llevar a buscar otra alternativa. En promedio y en el mejor caso el Quicksort tiene una complejidad computacional de **O(N log (N))** cual es bastante bueno y en peor caso la complejidad es de **$O(N^2)$**.

Para la búsqueda se utilizó tanto una búsqueda binaria como una secuencial. Para encontrar entre la lista de datos ordenadas los UBIs correctos se utilizó una búsqueda binaria. Sin embargo, por la naturaleza de la búsqueda, no se puede asegurar que encontramos el primer dato de los múltiples que comparten ese identificador, cosa que necesitamos para imprimir los datos en orden. Sabiendo que encontramos un área de datos con esos 3 caracteres iniciales, procedemos a utilizar una búsqueda secuencial que recorre el arreglo en reversa para encontrar cuando es que el dato cambia y así tendremos la primera instancia de ese identificador. Se decidió utilizar una búsqueda secuencial debido a los datos que nos proporcionamos, donde la cantidad de iniciadores de UBI que se repiten son pocos, por lo que resulta más rápido que las repetidas instanciaciones de una búsqueda binaria. Sin embargo, si es que el sistema llegara a tomar más datos en un futuro, esta pudiera llegar a ser una futura mejora para el algoritmo. Considerando que la complejidad computacional de la búsqueda binaria es **O(log N)** mientras que la de la búsqueda secuencial es **O(N)**, esta diferencia se debe a la manera que procesamos el algoritmo con código y el procesamiento que se lleva a cabo por la computadora. Debido a que la secuencial se hace después de la binaria, la complejidad se suma y se utiliza el más grande por lo que el toral seria de **O(N)**.

De este modo es que se pudo cumplir con esta evidencia, con tiempos de computación suficientemente rápidos que una persona no percibe la espera de procesamiento en los datos. Se pudo cumplir con el ordenamiento de los datos al igual que la búsqueda e impresión de datos que tienen una misma serie de UBI identificado por las primeras tres letras.