Luis Angel Zavala Robles
A01706481

Reflexión Actividad.

El problema para resolver requirió de la implementación de algoritmos tanto de búsqueda como de ordenamiento para su solución.
Primeramente, para ordenar un conjunto de números, disponemos teórica y prácticamente de varios algoritmos con diferentes complejidades que son mayor o menormente convenientes según los requerimientos y las características del programa al que no estemos enfrentando. Para este problema en particular, he decidido utilizar el algoritmo de ordenamiento Quicksort ya que es uno de los más rápidos con una complejidad promedio de **O(n log n)**, además, disponíamos de una base de datos relativamente grande y este algoritmo es de los que mejor se comporta ante estas circunstancias. Este algoritmo es un algoritmo recursivo que a grandes rasgos va dividiendo nuestro vector eligiendo pivotes y formando nuevos vectores en función de si los números son mayores o menores al pivote elegido.
Otros algoritmos como Bubble Sort $O(n^2)$, InsertionSort $O(n^2)$ ó SelectionSort $O(n^2)$, hubieran tenido tiempos de ejecución significativamente más lentos debido a que su complejidad algorítmica era ordenes mayores. Otro algoritmo eficaz en este problema hubiera sido MergeSort quedando a juicio y gusto del programador su implementación ante la situación.

Por su parte las opciones para realizar la búsqueda dentro de un vector ordenado eran mediante búsqueda secuencial o búsqueda binaria. Sin embargo, la búsqueda binaria es más rápida que la búsqueda secuencial al tener una complejidad de **O(log n)** mientras que la búsqueda secuencial presenta $O(n^2)$.
Es por ello por lo que se opto por usar el algoritmo de búsqueda binaria para abordar la solución de nuestra situación problema.
Al algoritmo original de la búsqueda binaria se le realizaron algunas modificaciones debido a que en nuestro problema no solo requeríamos el índice del numero buscado o un -1 en caso de no encontrarse, sino que requeríamos saber, además, si el numero no encontrado era menor al primer elemento del vector o era mayor al ultimo elemento del vector. Es por ello por lo que se agregaron algunas condicionales en el algoritmo para obtener el return deseado según la situación, esto sin afectar su complejidad algorítmica.

Sin duda se trato de una actividad que puso a prueba conocimientos y habilidades al momento de solucionar problemas complejos que se asemejan a situaciones reales, retándonos a tener un pensamiento critico y buscar la que desde mi perspectiva era una solución óptima y fácil de implementar. Puso a prueba también el entendimiento que teníamos sobre los algoritmos tanto de búsqueda como de ordenamiento, ya que era necesario modificarlos en ciertos detalles y solo se podría lograr conociendo en su totalidad la forma de trabajar del algoritmo en cuestión.