Carlos Eduardo Ruiz Lira – A01735706

Reflexión

Durante el desarrollo de la actividad de ordenamiento y búsqueda se observó la necesidad de implementar este tipo de algoritmos para poder realizar la búsqueda en una base de datos de la formas más rápida y eficiente posible. Actualmente existen varios algoritmos que nos permiten tanto como ordenar y buscar una serie de elementos, por lo tanto, es responsabilidad de los programadores determinar cuál es el algoritmo más apropiado dependiendo de la situación en la que se encuentra. Para conseguir esto, se pueden utilizar varios parámetros de comparación.

En primer lugar, se debe considerar el tamaño de la muestra total de datos que desean ordenar. Se puede observar que entre menos elementos existen sobre los cuales buscar un elemento, u ordenarlos, el tiempo de complejidad de algoritmo no es tan relevante ya que todos tienen un tiempo de ejecución similar, por lo que probablemente un método de ordenamiento por intercambio, o una búsqueda linear, ambas con complejidad O(n) podrían ser opciones simples pero eficaces para trabajar con pocos elementos.

Sin embargo, como se observó en la realización de esta actividad, cuando se tratan con muchos elementos, o con bases de datos que irán aumentando de tamaño con el tiempo, es mejor recurrir a algoritmos más complejos, pero más eficaces para reducir el tiempo de ejecución del programa. En este caso, al trabajarse con alrededor de 16 mil datos, se decidió utilizar el algoritmo Merge Sort ya que el tiempo de ejecución se reduce a sólo O(nlogn), bastante significativo para el número de elementos que se tiene.

Por otro lado, en métodos de búsqueda también depende si la base de datos ya se encuentra ordenada o no, ya que métodos como la búsqueda binaria no podrían ser utilizados si los datos no se encuentran ordenados. En este caso como los datos ya habían sido ordenados anteriormente, la búsqueda binaria fue una opción óptima, pues el método tiene un orden de ejecución de O(logn). Sin embargo, esto sólo es correcto si sólo se busca un solo elemento, por lo que este método es más efectivo si no hay valores repetidos. En este contexto, el tiempo de ejecución es realmente de O(logn) + O(k), donde k es el número de valores repetidos encontrados aledaños al valor que localizó el algoritmo, lo cual sigue siendo un tiempo mucho menor que utilizar un algoritmo linear O(n)