Paulina Lizet Gutiérrez Amezcua
A01639948
10 de septiembre del 2021
Programación de estructuras de datos y algoritmos fundamentales

Act 1.3
Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

Los algoritmos de ordenamiento y búsqueda son de suma importancia, no solo para resolver esta situación problema si no que para hacer cualquier programa mas eficiente. Respecto a la ordenación, la mayoría de datos de un programa están desordenados, una vez ordenándolos se vuelven mas eficientes y simplifican la búsqueda de información al aplicar un algoritmo de búsqueda. La complejidad computacional es un dato clave al usar los algoritmos ya que depende del número de datos a procesar, con esto se puede saber que algoritmo aplicar ya que la mayoría de programas pueden llegar a tener miles de datos, por lo tanto, se debe buscar el algoritmo que no degrade el rendimiento del conjunto de datos.

Para la actividad necesitábamos un algoritmo de ordenamiento y uno de búsqueda, decidimos usar el método burbuja y búsqueda secuencial, porque nos parecieron los mas sencillos de aplicar de acuerdo al código que teníamos, aquí una breve explicación de lo que hace cada uno:

Método burbuja: compara cada elemento del vector, intercambiando la posición si el orden está equivocado, revisa el vector varias veces hasta que no se necesiten mas cambios. Su complejidad es $O(n^2)$.

Búsqueda Secuencial: este método va recorriendo el vector elemento por elemento y va comparando con el valor que se busca. Su complejidad es mejor O (1) cuando el elemento a buscar este en la primera posición, promedio O(n/2), peor O(n) cuando el elemento este en la última posición.

Al final nos dimos cuenta que no eran los métodos mas eficientes porque tardaba mas tiempo en compilar, pero aun así hacían lo que buscamos y eran los mas simples de aplicar. Descubrimos que lo ideal para compilar mas rápido hubiera sido aplicar el método Merge Sort el cual consiste en partir el vector en mitades, ordenar cada mitad, intercalarlas ordenadamente y repetirlo para cada mitad, con una complejidad de O (n log n) y la búsqueda binaria la cual compara el elemento central con el que se está buscando, si coinciden finaliza, si no busca si el elemento es mayor o menor y se hace la búsqueda en el arreglo correspondiente, su complejidad es O (log n). Son más efectivos porque en el caso de la situación problema ordenamos los elementos del vector por lo tanto hubiera sido mas rápido que encontrara pero mas difícil de implementar.