Tecnológico de Monterrey

Materia

Programación de estructuras de datos y algoritmos fundamentales (Gpo 4)

Act 1.3

Profesor:

Dr. Jesús Guillermo Falcón Cardona

Franco Sotomayor Casale -A00831450

Fecha de entrega
10 de septiembre del 2021

Importancia y eficiencia de diferentes algoritmos de ordenamiento y búsqueda.

Al comenzar la actividad 1.3 era difícil poder encontrar una idea o manera de poder almacenar los tipos de datos y poder ordenarlo todo al mismo tiempo. Al completar la actividad es mucho mas claro lo importante y eficiente que son estos algoritmos de ordenamiento y búsqueda. Reconocer esto ya que al iniciar la actividad originalmente había usado el algoritmo de inserción. Al correr el programa se tardaba medio minuto para poder completar el ordenamiento completo de la bitácora.txt. Al cambiarlo para el algoritmo de Merge la diferencia era muy aparente. La diferencia entre ambos algoritmos era muy evidente ya que este medio minuto cambió hacia pocos segundos. Esto me ayudó a pensar y entender lo importante que es que los algoritmos sean eficientes, ya que probablemente existen bases de datos mucho más grandes de los que tenemos en este archivo .txt el usar un algoritmo inapropiado podría llevar a que tomen segundos o incluso minutos en poder acabar de hacer su propósito. Es importante reconocer que en el mundo real, es necesario escoger el algoritmo correcto ya que esos pocos minutos o segundos de diferencia podría crear problemas grandes.

Explicación de diferentes algoritmos de ordenamiento y búsqueda para esta situación problema, así como la complejidad computacional de cada uno de ellos.

Los algoritmos de ordenamiento y búsqueda que se usaron fueron Merge y Binary Recursive Search. El mergeSort como a su nombre divide es un algoritmo de división y conquista, la manera que sirve es que rompe el algoritmo entre dos algoritmos más pequeños hasta el punto en el cual se solucionen más fáciles. Al hacer la comparación de cada subarreglo se ordenan y se juntan después para hacer un arreglo ordenado. La complejidad computacional termina siendo O(nlog(n))

La manera en que sirve la búsqueda binaria recursiva es al igual un tipo algoritmo de división, lo que hace es que divide el algoritmo entre dos subarreglos y luego se forma recursiva. De esta manera puede reducir el espacio cada tiempo, luego se comparan los valores de en medio para buscar un tipo de dato deseado. Si no se encuentra el dato de un lado del subarreglo se busca en el otro subarreglo, así se puede bajar el espacio para buscar. La complejidad computacional de este algoritmo termina siendo un O(log(n))