Act 1.3 - Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

Alumno: Ricardo González Leal

Matrícula: A01639036

Materia: Programación de estructuras de datos y algoritmos fundamentales (TC1031)

Grupo: 13

Profesor: Luis Ricardo Peña Llamas

Fecha: 10/09/21

Importancia y Eficacia de los algoritmos de ordenamiento y búsqueda

La ordenación de datos es una aplicación fundamental en la computación. La mayoría de los datos producidos por un programa están ordenados de alguna manera, y muchos de los procesos computacionales que realizan los programas son mucho más eficientes si los datos manejados se encuentran ordenados.

Normalmente, cuando se requiere efectuar una ordenación o una búsqueda manejando un gran volumen de datos es importante hacer uso de algoritmos que no perjudiquen el rendimiento y se logre la mayor eficacia (rapidez de ejecución) posible con el tamaño del conjunto de datos. Para esto se busca hacer uso de algoritmos que tengan la menor complejidad computacional.

A continuación, se explican algunos de los métodos de ordenamiento más comunes, incluyendo su complejidad computacional. Así mismo se explica cuáles son los que utilizamos para está situación problema y el porqué.

Método de la Burbuja (Bubble Sort)

Consiste en recorrer un arreglo un cierto número de veces, realizando comparaciones entre pares de valores que se encuentran juntos. Si ambos datos no están ordenados, se intercambian. Esta operación se repite n-1 veces, donde n corresponde al tamaño del arreglo. Al final de la última iteración, el elemento mayor habrá llegado a la última posición; mientras que al final de la segunda iteración, el segundo elemento mayor habrá llegado a la penúltima posición, y así sucesivamente. Su tiempo de complejidad promedio es O(n^2).

Método de Inserción:

Consiste en comparar cada elemento con todos sus elementos anteriores, con esto, el elemento es colocado o insertado en la posición adecuada. Su tiempo de complejidad promedio es O(n^2).

Merge Sort:

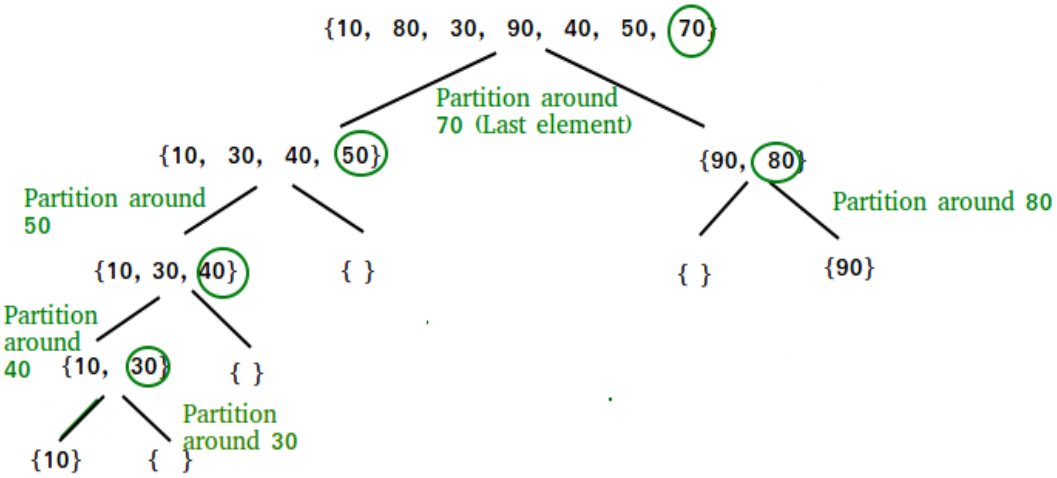
Merge Sort es un algoritmo de ordenamiento basado en la técnica de "divide y vencerás". Lo que hace es dividir el arreglo en dos mitades, se llama a sí mismo para las dos mitades y luego fusiona las dos mitades ordenadas. La función merge () se usa para fusionar dos mitades. La combinación (arr, l,

m, r) es un proceso clave que asume que arr [l..m] y arr [m + 1...r] están ordenados y fusiona los dos subarreglos ordenados en uno.

El tiempo de complejidad promedio de este algoritmo es O (n log(n)).

Quick Sort (Utilizado para esta situación problema)

QuickSort es un algoritmo basado en la técnica "divide y vencerás". Lo que hace es seleccionar un elemento como pivote y dividir el arreglo dado alrededor del pivote elegido. Hay muchas maneras de escoger el pivote en el QuickSort, se puede escoger el primer elemento, el último, uno aleatorio y el central, . Para este programa utilizamos el elemento central.



La parte central del algoritmo es la partición, que parte el algoritmo alrededor del pivote y acomoda a los elementos menores del pivote de un arreglo y a los elementos mayores en otro arreglo, después particiona las particiones y repite el proceso hasta que todo el arreglo esta ordenado.

El tiempo de complejidad promedio de este algoritmo es O(n log(n)).

Métodos de Búsqueda

- **Búsqueda Secuencial (Utilizado para esta situación problema)**
  Este método consiste en recorrer el arreglo elemento por elemento e ir comparándolos con el valor buscado. El resultado de la búsqueda es la posición del elemento buscado, si es que se encontró, de lo contrario es -1. Su complejidad es de O(n).

- **Búsqueda Binaria**

La búsqueda binaria es el método, donde si el arreglo o vector esta bien ordenado, se reduce sucesivamente la operación eliminando repetidas veces la mitad de la lista restante.
Funciona dividiendo repetidamente por la mitad la parte de la lista que podría contener el elemento, hasta que haya reducido las posibles ubicaciones a solo una.
El tiempo de complejidad promedio de este algoritmo es O(log(n)).

Conclusión

Con esta situación problema pude reflexionar sobre la utilidad de los métodos de ordenamiento y búsqueda, y más que nada, el saber sobre cuáles de ellos utilizar en ocasiones específicas. Dado que la bitácora que se nos proporcionó tenía miles de registros (más de 16000 registros), nos dimos cuenta que definitivamente para ordenar los registros lo mejor era hacer uso de un algoritmo que fuera lo más rápido y eficiente posible; por lo cual no pensamos más de dos veces y elegimos el QuickSort, ya que además de que este algoritmo cuenta con el mejor tiempo de complejidad, también cuenta con el mejor espacio de complejidad (O(log(n)), siendo este el más viable para ordenar una gran cantidad de elementos como el del archivo proporcionado.

También es importante mencionar que nosotros utilizamos la búsqueda secuencial para poder obtener una fecha cercana a la buscada en caso de que la buscada no existiera; y además, la búsqueda binaria nos regresaba el índice de cualquier registro encontrado en el día buscado, lo cual no nos funcionaba a nosotros, y fue otro motivo más por el cual utilizar búsqueda secuencial, para poder obtener el primer registro correspondiente de la fecha ingresada.