Tecnológico de Monterrey

Jose Manuel Martinez Morales

9 de Julio del 2023

Act 1.3 Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

Programación de estructuras de datos y algoritmos fundamentales

Iván Reyes Amezcua

Complejidades

Complejidad de convertDayToValue

Esta función convierte meses en un valor numérico para ordenarlos de manera mas sencilla, la complejidad puede ser constante si el mes es enero, pero en otros casos el algoritmo se repetirá n veces hasta que se encuentre el mes, por lo que la complejidad sera de O(n).

Complejidad de fileToVector

En este caso dentro de la función tenemos un loop while donde se ejecuta n veces para leer cada uno de los para meterlos dentro de un vector, dentro de la misma función se encuentra un loop que se repite 4 veces para almacenar cada tipo de dato dentro de un vector lo que nos da una complejidad constante, con esto podemos concluir que la complejidad del programa es de O(n).

Complejidad de Partition

La función Partition se utiliza dentro de la función sortV y se utiliza para dividir el vector en subvectores y se realiza n veces, por lo cual la complejidad de esta función es O(n).

Complejidad de sortV

La función sortV se basa en el algoritmo de ordenación quicksort, esta función se realiza iterativamente para ordenar los subvectores a la izquierda y derecha de nuestro pivote, cada vez que se manda a llamar a la función el tamaño se reduce a la mitad y se llamará n veces por lo cual la complejidad es de O(log n), al sumarle la complejidad de la función Partition obtendremos que la complejidad total es de la función es O(n log n).

Complejidad de printBitacora

La función printBitacora tiene un bucle for que itera n veces para imprimir el vector en un archivo linea por linea, por lo que la complejidad de la función es O(n).

Complejidad de printDates

Para la función printDates primero utilizamos un algoritmo de búsqueda binaria el cual al dividir los vectores en dos mas pequenos n veces podemos descubrir que tiene una complejidad de O(log n), pero al tener numeros repetidos también implementamos una

búsqueda secuencial como el peor caso que se va a repetir n veces, lo cual nos dará que en el peor caso la complejidad será de O(n)

Reflexión

El uso de algoritmos de ordenamiento y búsqueda en esta situación problema fueron cruciales para obtener un programa eficiente y efectivo.

Conocer los métodos de ordenamiento nos permite ver los datos de manera estructurada y secuencial, en este caso utilizamos el algoritmo QuickSort, el cual nos ayudó a ordenar de manera eficiente los datos de bitácora gracias a su complejidad de O(n log n) que le permite manejar grandes conjuntos de datos de manera eficiente.

Para poder localizar datos entre una rango de fechas, recurrimos a los algoritmos de búsqueda, en este caso utilizamos el algoritmo de binary search para encontrar un índice del rango de las fechas solicitado, pero al tener datos que se repetían también utilizamos unas búsqueda secuencial para definir nuestros límites en el índice.

Con la ayuda de estos algoritmos pudimos realizar de manera correcta y eficiente el problema presentado, sin embargo siempre hay que estar pendiente de las complejidades de nuestros programas para mejorar el rendimiento del programa.