Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación de estructuras de datos y algoritmos fundamentales

Tarea individual: Act-Integradora-2 Estructuras de datos lineales

Profesor: Dr. Eduardo Arturo Rodríguez Tello

Marlon Yahir Martínez Chacón - A01424875

20/04/2023

Marlon Yahir Martínez Chacón – A01424875

Importancia y eficiencia del uso de las diferentes estructuras de datos lineales

Esta actividad fue parecida a la anterior actividad con una problemática que se nos encomendó resolver donde unos servidores fueron hackeados, por lo que se consiguió mucha información sobre los ataques desde el tiempo exacto en el cual fue hackeado, su dirección ip, el puerto y también el motivo de la falla.

Se crearon varios registros con estos datos por lo que se necesita una estructura de datos para poder manipular todos estos registros y que sea amigable para que los encargados de la tecnología que fue atacada puedan agregar otras funciones que les permitan explotar toda esta información para sacarle el mayor provecho y no vuelva a ocurrir este ataque; pero encontrar una estructura no es fácil debido a que se tienen más de 16 mil registros con valiosa información, lo principal que se busca es implementar búsquedas de las fechas que el usuario necesita para desplegar un intervalo pero para poder realizar primero se tiene que ordenar toda la información para que sea más fácil de hacer y práctico, además de permitir obtener archivos .txt con los datos solicitados y todos los registros ordenados, por lo que esto fue lo principal a implementar.

La estructura de datos que se usó para poder implementar todo lo que necesita el usuario fueron las listas doblemente encadenadas que tienen un componente más de las listas simples el cual nos permite que sean más dinámicas y prácticas para la manipulación de información que nos permite obtener el nodo anterior en cualquier nodo, esto hace que los procesos sean más rápidos en cuanto a los bucles y existen más formas de poder recorrer las listas de dos puntos (head y tail) al mismo tiempo que de uno solo, gracias a las listas doblemente encadenadas podemos manipular los más de 16 mil registros existentes para poder ordenar los datos, mediante los algoritmos de merge sort y quick sort que son los algoritmos más eficientes dentro de todos los algoritmos que vimos en clase y esto se pudo observar mediante todas las comparaciones realizadas en el programas ya que no hubo mucha diferencia entre ambos algoritmos mientras que el merge sort tuvo 231,599 comparaciones, el quick sort ordenó todos los registros después de 271,877 comparaciones, por lo que podemos llegar a la conclusión que ambos algoritmos son cercanos en la eficiencia que dan a la hora de ordenar datos.

Tabla: Complejidad de los algoritmos de ordenamiento.

Algoritmo	Mejor	Promedio	Peor	Estable	Espacio
Swap sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	No	$O(1)$
Buble sort	$O(n)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Sí	$O(n)$
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	Sí	$O(\log n)$

Y esto se puede complementar con la información donde se puede observar que la complejidad que tienen ambos algoritmos es igual por lo que realizan comparaciones similares, pero en cuanto al espacio que ocupan como se puede observar el claro ganador es el quick sort al ocupar un menor espacio que el merge sort.

Gracias al ordenamiento de los datos de manera ascendente se pudo emplear la binary search para poder encontrar el nodo de la fecha de inicio y el nodo de la fecha final, se uso la binary search debido a que es la búsqueda más eficiente que existe para trabajar con grandes volúmenes de datos como se puede observar tiene una función logarítmica por lo que fue la mejor opción para tener los métodos más eficientes

Tabla: Complejidad de los algoritmos de búsqueda.

Algoritmo	Mejor	Promedio	Peor
Búsqueda secuencial	$O(1)$	$O(n)$	$O(n)$
Búsqueda secuencial en arreglos ordenados	$O(1)$	$O(n)$	$O(n/2)$
Búsqueda binaria	$O(1)$	$O(\log_2 n)$	$O(\log n)$

En conclusión, al ser una estructura de datos que nosotros implementamos mediante nuestras propias clases nos permite un mayor grado de dinamismo y flexibilidad para trabajar con grandes cantidades de datos sin comprometer la eficiencia, ya que al ser punteros los componentes de cada una de las clases podemos liberar la memoria cuando ya no lo ocupemos y nos genere algún error, además de que al ser implementado por nosotros la librería de las listas podemos agregar más funciones que solamente puedan ser usados por los miembros o los objetos que instanciemos haciendo que el código que mostremos al usuario sea menor y podamos tener más cosas escondidas para que no sean modificadas al antojo de nadie. Al realizar esta actividad que es similar a la anterior, me di cuenta que cada una de las estructuras de datos tiene sus ventajas y desventajas, y que siempre se debe de buscar la más eficiente que en este caso fueron las listas doblemente encadenas para ocupar menos memoria y los procesos sean más rápidos.

Referencias

- El uso práctico de las listas enlazadas para el almacenamiento de los datos del usuario. (s. f.). Delfino.cr. https://delfino.cr/2022/02/el-uso-practico-de-las-listas-enlazadas-para-el-almacenamiento-de-los-datos-del-usuario
- Listas Doblemente Enlazadas. (s. f.). http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro9/listas_doblemente_enlazadas.html
- Estructura de Datos : Lista Enlazada Doble. (s. f.). https://www.fceia.unr.edu.ar/estruc/2005/listendo.htm