Valeria Salazar Pinto
A01736652

Reflexión sobre el uso de algoritmos de ordenamiento y búsqueda

Introducción

Actividad 2.3

En esta actividad el enfoque ya no fue solo ordenar o buscar información, sino pensar bien cómo guardar los datos desde el inicio. El problema seguía siendo el análisis de una bitácora relacionada con ataques cibernéticos, pero ahora el reto estaba en usar correctamente estructuras de datos lineales para que todo el proceso tuviera sentido y fuera eficiente.

Cuando se trabaja con una bitácora grande, no es práctico manejar los datos de cualquier forma. No basta con leer el archivo y mostrarlo, porque después se necesita ordenar por fecha y hora, buscar rangos específicos y guardar resultados. Todo eso obliga a pensar en qué estructura facilita realmente ese trabajo.

Uso de estructuras de datos lineales

Las estructuras de datos lineales permiten recorrer la información de manera ordenada y controlada. En este problema, cada registro tiene varios datos importantes, como fecha, hora, IP y descripción. Tenerlos organizados dentro de una estructura adecuada hace que el manejo de la información sea mucho más claro.

Si se hubiera usado una estructura que no se adapta bien a recorridos constantes o modificaciones, el código se volvería más complicado y menos eficiente. Por eso aquí no solo importaba que funcionara, sino que fuera lógico y práctico para el tipo de operaciones que se iban a hacer.

Por qué se usó una Doubly Linked List

En la solución se decidió usar una Doubly Linked List en lugar de una Linked List simple, y esta decisión tuvo mucho sentido durante el desarrollo. Al tener referencias tanto al nodo siguiente como al anterior, la lista se vuelve más flexible para recorrerla y manipularla.

Esto fue especialmente útil al momento de ordenar la bitácora y al recorrer los rangos de fechas. Una vez que se encuentran las fechas de inicio y fin, poder avanzar fácilmente entre nodos hace que el recorrido sea más natural. Además, durante el proceso de ordenamiento, dividir y unir sublistas resulta más sencillo cuando se tienen referencias dobles.

Aunque la Doubly Linked List usa un poco más de memoria, en este caso esa diferencia no fue un problema. La ventaja en claridad y facilidad de manejo fue mucho más importante que ese pequeño costo extra.

Complejidad de las operaciones básicas

Valeria Salazar Pinto
A01736652

Analizar la complejidad de las operaciones ayuda a entender por qué esta estructura funciona bien en este problema.

La inserción de los registros en la lista es eficiente, ya que se van agregando conforme se lee el archivo, lo cual tiene un costo constante. El borrado también es más sencillo si se tiene la referencia al nodo, gracias a que se puede ajustar tanto el nodo anterior como el siguiente.

Específicamente, la inserción al final de la lista tiene una complejidad de O(1), lo que permite cargar todos los registros de manera sumamente rapida.

La búsqueda sigue siendo lineal, ya que es necesario recorrer la lista, pero en este caso se complementa con el ordenamiento y la búsqueda binaria para localizar los rangos de fechas. Esto hace que, en la práctica, el proceso sea mucho más rápido que recorrer toda la bitácora sin ningún orden.

Mientras que el borrado es O(1) si ya se tiene la referencia al nodo , la búsqueda de una fecha específica sigue siendo O(n) esto es por la naturaleza secuencial de las listas. Sin embargo, al estar ordenada, la búsqueda binaria optimiza el acceso a los rangos pedidos.

Comparación de Merge Sort y Quick Sort

Para ordenar la Doubly Linked List se utilizó Merge Sort, y se comparó su desempeño con Quick Sort, que se había implementado en una actividad anterior. Merge Sort resultó ser una mejor opción para este caso, ya que funciona muy bien con listas ligadas y mantiene un desempeño estable sin importar el orden inicial de los datos.

Quick Sort es muy eficiente en promedio, pero depende más del acceso directo a posiciones, lo cual lo hace más adecuado para arreglos. Además, en el peor de los casos puede volverse mucho más lento. Por eso, para esta situación específica, Merge Sort fue la opción más conveniente tanto en teoría como en la práctica.

Si comparamos tiempos, ambos suelen ser O(n log n), pero la gran ventaja de Merge Sort es que siempre mantiene ese ritmo. Quick Sort, aunque es bueno, puede llegar a ser muy lento (O(n2)) si los datos no ayudan. Además, como en una lista doble nos movemos por punteros y no por índices, Merge Sort aprovecha mucho mejor la estructura para dividir y ordenar todo de forma eficiente.

Reflexión final

Esta actividad me ayudó a entender que elegir una estructura de datos no es algo que se haga al azar. La forma en la que se almacenan los datos influye directamente en qué tan fácil es ordenarlos, buscarlos y analizarlos después.

Usar una Doubly Linked List permitió manejar la bitácora de una forma más clara y flexible, y combinarla con Merge Sort y búsqueda binaria hizo que el programa fuera más eficiente y ordenado. Comparar distintos enfoques también ayudó a ver que no siempre el algoritmo más conocido o más simple es el mejor para todos los casos.

Valeria Salazar Pinto
A01736652

Tecnológico de Monterrey

Hacer que todo esto trabajara en equipo redujo muchísimo el tiempo de espera al buscar rangos de fechas. Me quedó claro que si eliges bien tu estructura de datos desde el inicio, los algoritmos de búsqueda funcionan mucho mejor.

En general, esta actividad refuerza la idea de que para resolver problemas reales no solo importa que el programa funcione, sino que esté bien pensado desde la estructura de datos hasta los algoritmos que se utilizan.

Referencias

Análisis del ordenamiento rápido (artículo). (n.d.). Khan Academy. https://es.khanacademy.org/computing/computer-science/algorithms/quick-sort/a/analysis-of-quicksort

R, V. (2021, August 6). What is Data Structure?| Important points explained|Great Learning. GreatLearning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/data-structure-tutorial-for-beginners/

Estructura de datos: ¿Para qué sirve y qué tipos existen? (2024, June 6). UCMA. https://www.universitatcarlemany.com/actualidad/blog/estructura-datos/

GeeksforGeeks. (2013, April 25). *QuickSort on Doubly Linked List*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/quicksort-for-linked-list/

GeeksforGeeks. (2015, May 2). Merge Sort for Doubly Linked List. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/merge-sort-for-doubly-linked-list/

HG, E. (2021, May). *Listas doblemente enlazadas o Double Linked List*. Medium. https://eddiejesus1197.medium.com/listas-doblemente-enlazadas-o-double-linked-list-f22f166e1bb8