A01713396 - Santiago Martin del Campo

Reflexión Sobre el Uso de Doubly Linked List

En esta actividad se presenta una situación en la que es necesario manejar grandes cantidades de información provenientes de una bitácora de accesos, similar a una base de datos que almacena registros generados continuamente por un sistema.

El problema es que este tipo de situaciones no se limita únicamente a guardar los datos, sino a estructurarlos de forma que puedan ser ordenados, consultados y analizados de manera eficiente. Cuando se trabaja con este tipo de información, el volumen de registros y la secuencia en la que ocurren los eventos juegan un papel fundamental, ya que de ello depende la rapidez con la que se pueden detectar patrones o encontrar eventos específicos dentro de la bitácora.

Importancia del Uso de double linked lists:

El uso de doubly linked lists en esta actividad fue fundamental, debido a las capacidades de recorrido bidireccional que ofrece esta estructura.

A diferencia de una lista simplemente enlazada, donde solo es posible avanzar nodo por nodo, una doubly linked list permite moverse tanto hacia adelante como hacia atrás, lo cual facilita mucho la manipulación de los registros dentro de la bitácora.

Esto resulta especialmente útil cuando se requiere eliminar, reorganizar o analizar elementos específicos, ya que se tiene acceso directo al nodo anterior y al siguiente sin necesidad de recorrer toda la estructura. Aunque este tipo de lista utiliza más memoria al almacenar referencias adicionales, ofrece un mayor número de operaciones eficientes en tiempo constante, lo que la hace más adecuada para un problema donde los datos se consultan y modifican con frecuencia.

Ventaja al Utilizar doubly linked list sobre linked list simple:

Al trabajar con información ordenada por fecha y hora, poder desplazarse tanto hacia adelante como hacia atrás permite un manejo más flexible de los datos y simplifica operaciones como el ordenamiento y la búsqueda de rangos específicos.

Aunque esta estructura requiere un mayor uso de memoria en comparación con una linkedlist simple, en la práctica ofrece un mayor control sobre los registros

Comparación entre algoritmos de ordenamiento QuickSort y MergeSort.

Para poder comparar estos algoritmos es necesario comprender de qué manera funciona cada uno y cómo se comportan en la práctica al trabajar con grandes volúmenes de datos.

Estas son algunas de las características más relevantes de los algoritmos de ordenamiento analizados:

QuickSort:

- Presenta una complejidad promedio de **O(n log n)**.
- Divide la información en dos partes: valores menores y mayores al pivote.
- Trabaja de forma recursiva sobre los subconjuntos generados.
- El número de comparaciones puede variar dependiendo del orden inicial de los datos.

Merge Sort:

- Mantiene una complejidad de **O(n log n)** en todos los casos
- Divide la información en mitades más pequeñas de manera recursiva.
- Ordena cada mitad por separado y luego las combina.
- Es muy adecuado para manejar grandes volúmenes de datos.
- El rendimiento se mantiene estable sin importar el orden inicial de los datos.

En cuanto a rendimiento en este caso el algoritmo más adecuado es Merge Sort, porque se adapta mejor a su forma de recorrer y reorganizar los datos.

Al dividir la lista en partes más pequeñas y unirlas ajustando los enlaces entre nodos, el algoritmo aprovecha la estructura secuencial de la lista sin depender de accesos directos. Además, su rendimiento se mantiene constante incluso cuando el volumen de datos aumenta, lo que lo convierte en una opción más estable y confiable para ordenar registros en este tipo de estructura.

Algoritmo de Búsqueda Binaria

En esta actividad se utilizó el algoritmo de búsqueda binaría para encontrar la fecha inicial del rango y después la fecha final del rango originado de una consulta hecha por el usuario, facilitando la localización de rangos específicos dentro de la bitácora.

Complejidad temporal de operaciones solicitadas.

Inserción al inicio (addFirst) – O(1)

Se actualizan los apuntadores del nodo inicial

Recorrido de la lista – O(n)

Para mostrar o procesar todos los registros de la bitácora es necesario recorrer la lista completa nodo por nodo.

Búsqueda binaria en lista ligada – O(n log n)

El nodo medio se obtiene recorriendo la lista en cada iteración.

Borrado – O(n)

Requiere recorrer la lista para localizar el elemento antes de eliminarlo.

El mayor impacto se presenta en el ordenamiento, ya que al trabajar con listas enlazadas la forma de recorrer y reorganizar los datos es distinta, lo que provoca que algunos algoritmos no mantengan el mismo comportamiento que tendrían en otras estructuras.

Reflexión Personal:

En mi caso es la primera vez utilizando listas doblemente ligadas; inicialmente fue difícil entender el funcionamiento de esta estructura debido a estar acostumbrado al uso de vectores, donde el acceso y la manipulación de los datos resultan más directos. Sin embargo, conforme fuimos trabajando con los nodos y sus apuntadores, fue más claro cómo esta estructura termina ofreciendo una mayor flexibilidad para recorrer y reorganizar la información, especialmente al manejar grandes volúmenes de datos.

Referencias Bibliográficas:

GeeksforGeeks. (2025, 27 diciembre). Binary search. GeeksforGeeks.
https://www.geeksforgeeks.org/dsa/binary-search/

www.naukri.com. (s. f.). Code 360 by Coding Ninjas. 2024 Naukri.com.
https://www.naukri.com/code360/library/why-is-quick-sort-preferred-for-arrays-and-merge-sort-for-linked-lists

GeeksforGeeks. (2025a, julio 23). Why Quick Sort preferred for Arrays and Merge Sort for Linked Lists? GeeksforGeeks.
https://www.geeksforgeeks.org/dsa/why-quick-sort-preferred-for-arrays-and-merge-sort-for-linked-lists/

GeeksforGeeks. (2025a, julio 11). Difference between Singly linked list and Doubly linked list. GeeksforGeeks.
https://www.geeksforgeeks.org/dsa/difference-between-singly-linked-list-and-doubly-linked-list/

GeeksforGeeks. (2025b, julio 23). Merge Sort for Doubly Linked List. GeeksforGeeks.
https://www.geeksforgeeks.org/dsa/merge-sort-for-doubly-linked-list/

GeeksforGeeks. (2025c, julio 23). QuickSort on Doubly Linked List. GeeksforGeeks.
https://www.geeksforgeeks.org/dsa/quicksort-for-linked-list/

Wikipedia contributors. (2026, 23 enero). Sorting algorithm. Wikipedia.
https://en.wikipedia.org/wiki/Sorting_algorithm#Comparison_of_algorithms