Tecnológico de Monterrey

Programación de estructuras de datos y algoritmos fundamentales

Act 2.3 - Estructuras de datos lineales

Maestro: Eduardo Arturo Rodríguez Tello

Ana Regina Rodríguez Múzquiz | A01286913

Enero 2026

Reflexión

En esta actividad pude entender mucho mejor la importancia de elegir correctamente una estructura de datos dependiendo del problema que se quiere resolver. En este caso, la actividad consistía en leer una bitácora con miles de registros, ordenarlos por fecha y hora, y después realizar búsquedas dentro de un rango de fechas específico. Para este tipo de problema, usar estructuras de datos lineales es lo ideal, ya que los datos se procesan de manera secuencial y ordenada.

Aunque yo pensaba que un vector puede parecer una buena opción, al trabajar con grandes cantidades de datos y con operaciones como inserciones, ordenamientos y búsquedas, su eficiencia se ve más limitada. Por esta razón, en esta actividad fue mejor utilizar una Doubly Linked List (lista doblemente ligada), ya que permite recorrer los datos tanto hacia adelante como hacia atrás y facilita el trabajo con algoritmos que se basan en apuntadores y no en índices.

Una de las principales ventajas de usar una Doubly Linked List en lugar de una Linked List simple es que se puede recorrer la lista en ambos sentidos gracias a sus apuntadores next y prev. Esto resulta útil al implementar algoritmos de ordenamiento como Merge Sort y Quick Sort adaptados a listas ligadas, donde no se puede acceder directamente por índices como en un vector. A parte, al momento de realizar búsquedas dentro de un rango, poder avanzar nodo por nodo sin necesidad de cálculos extra mejora la claridad del algoritmo.

Dentro de nuestro programa sí se nota cómo la complejidad de la lista doblemente ligada afecta el desempeño. Por ejemplo, al momento de insertar los registros, yo los iba agregando uno por uno al final de la lista conforme se leía el archivo, y eso fue bastante eficiente porque siempre tenía un apuntador al último nodo, entonces cada inserción se hacía rápido, sin tener que recorrer toda la lista.

El borrado no lo usé directamente en esta actividad, pero al trabajar con la lista doblemente ligada me quedó claro que, si en algún punto tuviera que eliminar un registro específico, sería más sencillo hacerlo porque cada nodo conoce al de adelante y al de atrás. Eso evita tener que buscar otra vez el nodo anterior, como pasaría en una lista simple.

En cuanto a la búsqueda, ahí sí se nota más, porque para encontrar registros o recorrer un rango de fechas es necesario ir nodo por nodo. Aun así, como primero ordené la lista, pude apoyarme en una búsqueda tipo binaria para ubicar el inicio y el fin del rango, y después solo recorrer los registros que sí me interesaban. Eso hizo que el proceso fuera más ordenado.

Para el ordenamiento, se implementaron y compararon dos algoritmos: Merge Sort y Quick Sort, ambos adaptados para trabajar con listas doblemente ligadas usando apuntadores. Teóricamente, ambos algoritmos tienen una complejidad promedio de O(n log n), pero en la práctica los resultados mostraron diferencias. En la ejecución del programa, Merge Sort tuvo un desempeño más estable y predecible, mientras que Quick Sort realizó más comparaciones y swaps, lo que confirma que Merge Sort suele ser más eficiente y consistente cuando se trabaja con listas ligadas.

También, al aplicar la búsqueda por rango de fechas, se utilizó una adaptación de búsqueda binaria sobre la lista ordenada, lo cual permitió localizar de forma más eficiente los límites del rango solicitado por el usuario. Esto, combinado con el uso de la lista doblemente ligada, permitió mostrar los resultados en pantalla y almacenarlos correctamente en un archivo.

En conclusión, esta actividad me ayudó a comprender que no siempre la estructura de datos más común es la mejor opción. En problemas donde se manejan grandes cantidades de información y se requiere flexibilidad para ordenar y recorrer datos, una Doubly Linked List resulta más adecuada que una lista simple tanto por su eficiencia como por su facilidad de adaptación a algoritmos avanzados.

Referencias

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

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

GeeksforGeeks. (2025c, septiembre 19). *Doubly linked list tutorial*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/doubly-linked-list/

Replit. (s. f.-b). *Doubly linked list*. Replit. https://replit.com/@ertello/Doubly-Linked-List

KC Ang. (2014, 16 octubre). *Quicksort: Partitioning an array* [Vídeo]. YouTube. https://www.youtube.com/watch?v=MZaf_9IZCrc