
# Act-Integradora-2

## Conceptos básicos y algoritmos fundamentales

Programación de estructuras de datos y algoritmos fundamentales
(Gpo 850)

**Profesor**
Eduardo Arturo Rodríguez Tello

**Nombres:**

Federico Melo B. A00833536

**07/04/2025**

A. **Analiza la importancia y eficiencia del uso de las diferentes estructuras de datos lineales en una situación problema de esta naturaleza.**

Para procesar  bitácoras, como  es el caso en este reto, las estructuras de datos lineales como arreglos (std::vector), listas simplemente ligadas (Linked List) y listas doblemente ligadas (Doubly Linked List) son primordiales. Estas nos dejan representar secuencias ordenadas de elementos, que es de gran ayuda cuando buscas preservar el orden cronológico de los registros, hacer búsquedas, o usar algoritmos de ordenamiento.

B. **Reflexiona por qué en la solución de este reto es preferible emplear una Doubly Linked List y no una Linked List.**

Una Linked List nos deja  recorrer los elementos solo hacia adelante. Puede ser mas que suficiente para operaciones básicas de inserción y recorrido secuencial, pero tiene limitaciones en tareas más complejas como mergeSort, búsquedas binarizadas o delimitación de rangos. Por estas razones, es preferible emplear una Doubly Linked List y no una Linked List.  La Doubly Linked List va a permitir que podamos acceder tanto al nodo siguiente como al anterior, implementar de una manera más intuitiva algoritmos de ordenamiento y búsqueda y tener mejor control al recorrer y comparar elementos desde la mitad de la lista, lo cual es muy importante en una búsqueda binaria modificada.

C. **Analiza la complejidad computacional de las operaciones básicas de la estructura de datos empleada en tu implementación (inserción, borrado, búsqueda) y cómo esto impacta en el desempeño de tu solución.**

En nuestro caso, al tener una lista ordenada de forma descendente, adaptar la búsqueda binaria fue todo un reto. Como no se puede acceder a un nodo de manera directa por su índice como en un vector, tuvimos que recorrer la lista desde el inicio cada vez que necesitábamos llegar al nodo medio. Esto obviamente afecta el rendimiento porque

aunque la búsqueda binaria reduce la cantidad de comparaciones, moverse por la lista sigue tomando muhco tiempo.

Por otro lado, implementar el mergeSort en una lista doblemente ligada fue más complicado de lo que esperábamos, principalmente por el manejo de  next y prev. Aun así, elegimos este algoritmo porque es mucho más eficiente que otros como el insertion sort en este tipo de estructuras. Al final el desempeño fue bueno.

D. **Explica con tus palabras los retos a los que te enfrentaste para implementar los algoritmos de ordenamiento y búsqueda haciendo énfasis en aquellos relacionados con a) su implementación en una Doubly Linked List y b) su implementación para una estructura de datos ordenada descendentemente.**

**a) Implementación en una Doubly Linked List**

Uno de los retos más grandes fue implementar el algoritmo mergeSort en una lista doblemente ligada, porque no se puede acceder de manera directa a un nodo por su posición como en un vector. Para dividir y fusionar las sublistas, tuvimos que trabajar con punteros que hace más complejo el código y hace que haya más riesgo de errores. Además, teníamos que tener mucho cuidado de actualizar correctamente los prev y next de cada nodo, para  que la lista resultante siguiera siendo válida y navegable.

**b) Implementación para una estructura ordenada descendentemente**

Otro reto importante fue adaptar la búsqueda binaria para que funcionara con una lista ordenada de forma descendente. Normalmente, el algoritmo está pensado para listas ordenadas de menor a mayor entonces tuvimos que ajustar las condiciones de comparación para que buscara correctamente de mayor a menor. También, tuvimos que tener mucho cuidado al definir el rango de búsqueda entre la fecha de inicio y la fecha de fin. Como los datos están en orden descendente, era fácil equivocarse y terminar

recorriendo el rango al revés. Por eso, validamos bien que se seleccionaran los registros correctos dentro del intervalo.

**Referencias**


Merge Sort for Doubly Linked List – GeeksforGeeks
https://www.geeksforgeeks.org/merge-sort-for-doubly-linked-list/

Binary Search on Singly and Doubly Linked List – GeeksforGeeks
https://www.geeksforgeeks.org/binary-search-on-singly-and-doubly-linked-list/

Doubly Linked List Data Structure – Programiz
https://www.programiz.com/dsa/doubly-linked-list

Why is binary search inefficient on linked lists? – Stack Overflow
https://stackoverflow.com/questions/11153163/why-is-binary-search-inefficient-on-linked-lists