Mariana Aguilar Requena
A01569276

Tecnológico de Monterrey

Act 2.3 - Actividad Integral - Actividad Integral - Estructuras de datos lineales (Reflexión)

En esta actividad trabajamos con la misma bitácora de 16 mil registros, pero esta vez usando una estructura de datos diferente, una Doubly Linked List (lista doblemente enlazada) en lugar de un vector. El objetivo era entender por qué algunas estructuras de datos son mejores que otras dependiendo de lo que necesites hacer con la información.

Al principio no entendía bien la diferencia. Un vector es fácil de usar y puedes acceder rápido a cualquier posición, pero tiene un problema: si quieres insertar o borrar datos en medio del archivo, tienes que mover todos los elementos que vienen después. Esto hace que estas operaciones sean lentas, con complejidad O(n).

Vimos que una Linked List simple mejora esto porque los nodos están conectados entre sí, lo que hace que insertar o borrar elementos sea más eficiente. El problema es que solo se puede recorrer en una dirección. Por eso está Doubly Linked List, que tiene como ventaja que cada nodo apunta tanto al siguiente como al anterior, lo que permite recorrer la lista en ambos sentidos sin complicaciones. Esto resulta muy útil para ciertos algoritmos, como Merge Sort, que necesitan dividir y recorrer la lista de diferentes formas.

Complejidad de las operaciones en una Doubly Linked List

- **Inserción al inicio (addFirst): O(1)**
  Es muy rápida porque solo se actualizan los apuntadores del nodo inicial.
- **Búsqueda: O(n)**
  En el peor de los casos es necesario recorrer toda la lista para encontrar un elemento.
  Búsqueda binaria: O(log n)
  Solo se puede aplicar si la lista ya está ordenada, por eso primero se utiliza un algoritmo de ordenamiento.
- **Borrado: O(n)**
  Primero se debe buscar el elemento y después ajustar los apuntadores, lo cual hace que el costo principal sea la búsqueda.

El mayor impacto se nota en el ordenamiento. A diferencia de los vectores, las listas enlazadas no permiten acceso directo por índice, por lo que algunos algoritmos se comportan de manera distinta.

Comparación entre QuickSort y MergeSort

En la actividad anterior usamos QuickSort con vectores y funcionó muy bien. En esta ocasión implementamos también MergeSort para comparar cuál se adapta mejor a una lista doblemente enlazada.

QuickSort:

- Tiene una complejidad promedio de **O(n log n)**.
- Aunque funciona bien, no es tan eficiente en listas enlazadas porque necesita acceder a posiciones específicas para hacer las particiones.

MergeSort:

- Tiene una complejidad garantizada de **O(n log n)** en todos los casos.
- Es un algoritmo más estable y consistente.
- Se adapta mejor a las listas enlazadas porque solo necesita seguir los apuntadores, sin acceso aleatorio.
  Divide la lista en mitades usando el método de “tortuga y liebre” (punteros lento y rápido).

Conclusión

Esta actividad me ayudó a entender que no existe una estructura de datos que sea la mejor para todo. Todo depende de qué tipo de operaciones se van a realizar con mayor frecuencia. Si se necesita acceso rápido por índice, un vector es la mejor opción. Pero si se van a hacer muchas inserciones, eliminaciones o recorridos en ambas direcciones, una Doubly Linked List es más conveniente.

También aprendí que el algoritmo de ordenamiento debe elegirse de acuerdo con la estructura de datos que se esté utilizando. En este caso, MergeSort y Doubly Linked List hacen una muy buena combinación porque ambos funcionan bien sin necesidad de acceso aleatorio.

Referencias

GeeksforGeeks. (2025, 19 de septiembre). *Doubly Linked List Tutorial*. Recuperado de https://www.geeksforgeeks.org/dsa/doubly-linked-list/

GeeksforGeeks. (2025, 3 de octubre). *Merge sort*. Recuperado de https://www.geeksforgeeks.org/dsa/merge-sort/

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

Programiz. (s.f.). *Linked List Operations: Traverse, Insert and Delete*. Recuperado de https://www.programiz.com/dsa/linked-list-operations