Programación de estructuras de datos y algoritmos fundamentales (Gpo 602)

Act 2.3 - Actividad Integral estructura de datos lineales (Evidencia Competencia)

investigación y reflexión de la importancia y eficiencia del uso de las listas doblemente ligadas en una situación problema de esta naturaleza

Fecha

15/10/2025

Profesor

Daniel Perez R

Alumno

José Luis Gutiérrez Quintero– A01739337

Marco Teórico:

Comparación de estructuras:

Array: Un array proporciona acceso aleatorio rápido a cualquier elemento por índice y suele ocupar posiciones contiguas en memoria, lo que beneficia la localidad. Sin embargo, su tamaño puede ser fijo o requerir redimensionamiento si es dinámico. Las operaciones de inserción o borrado en el medio de un array son costosas, ya que implican desplazar múltiples elementos. En contraste, una lista enlazada crece o disminuye dinámicamente reservando memoria nodo a nodo y no necesita espacio contiguo; esto evita copias masivas de datos, aunque cada nodo incurre en un sobrecoste de memoria por almacenar punteros adicionales. En nuestro caso, donde el número de registros de bitácora puede variar y se requieren inserciones frecuentes ordenadas, la flexibilidad de la lista es una ventaja sobre un arreglo estático.

Lista simplemente enlazada: En una lista simplemente ligada, cada nodo solo conoce a su sucesor, por lo que no es posible retroceder fácilmente ni acceder directamente al nodo anterior. Esto complica ciertas operaciones; por ejemplo, para eliminar un elemento dado es necesario recorrer la lista para hallar su predecesor. La lista doblemente ligada resuelve esto almacenando el puntero al nodo anterior, facilitando operaciones de borrado o inserción en cualquier posición sin recorrer desde la cabeza cada vez. No obstante, la lista simple utiliza ligeramente menos memoria (un puntero menos por nodo) y su implementación es más simple.

Lista Doblemente ligada:

Una lista doblemente ligada (o lista doblemente enlazada) es una estructura de datos dinámica formada por nodos enlazados secuencialmente. Cada nodo almacena un dato (en este caso, un registro de bitácora) y dos punteros: uno al nodo siguiente y otro al nodo anterior. A diferencia de una lista simplemente ligada, que solo tiene referencias hacia adelante, la lista doblemente ligada permite recorrer los elementos en ambas direcciones – hacia adelante y hacia atrás – facilitando operaciones de retroceso sin estructuras auxiliares. Esta característica agrega versatilidad, aunque con un pequeño coste en complejidad de implementación y uso de memoria (cada nodo mantiene un puntero extra).

En cuanto a operaciones básicas y complejidad, las listas enlazadas presentan un comportamiento distinto a los arreglos. Por ejemplo, la búsqueda de un elemento por valor o posición es una operación de tiempo lineal O(n), ya que en el peor caso hay que recorrer la lista nodo a nodo. En cambio, las inserciones y eliminaciones en los extremos (al inicio o al final) pueden realizarse en tiempo constante O (1) al actualizar pocos punteros, siempre que tengamos referencias directas a esas posiciones. Insertar o eliminar en posiciones intermedias también evita mover todos los elementos posteriores (como ocurriría en un array), aunque requiere primero navegar hasta la posición deseada, lo cual es O(n) en tiempo. En resumen, las listas doblemente ligadas ofrecen eficiencia en modificaciones locales, a cambio de un acceso más costoso por índice.

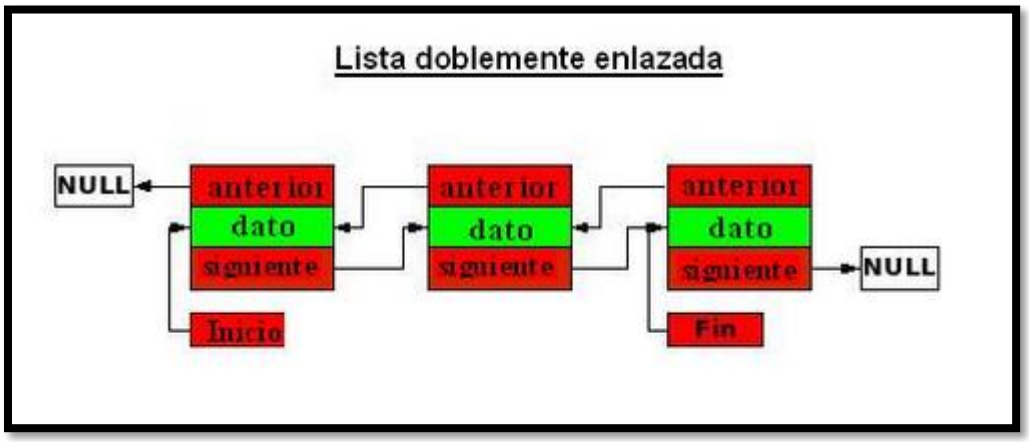

Figura 1. Listas doblemente Ligadas

Reflexión sobre el uso de listas doblemente ligadas en la bitácora

Relevancia de la lista doblemente ligada en el proyecto

Para el proyecto de bitácora para ordenar e imprimir registros por dirección IP, fecha y mensaje, me pidieron el uso de listas doblemente ligadas. Al principio dudaba si realmente era necesaria una estructura así en lugar de un array o una lista simple. Trabajando con ellas y después de ver algunos videos comprendí su relevancia: necesitaba manejar muchos registros y mantenerlos ordenados de forma eficiente. Con la lista doblemente ligada pude insertar nuevos registros en el lugar correcto según su IP y fecha sin tener que reordenar toda la colección, a pesar de que lleguen a ser más pesadas debido a los bytes que utilizan el hecho de que no sean estáticas facilita su uso.

Al inicio me abrumaba manejar dos punteros por nodo, mas que nada por que trabajaba con malloc y syntax de C y no C++, pero con la práctica entendí cómo enlazar los nodos correctamente y considerar casos especiales (como una lista vacía o insertar al inicio/final) además de que el trabajar con lenguaje de C al inicio me ayudo a su mayor comprensión y como se utilizan punteros para su estructura. Tuve algunos Segmentation fault al principio, pero al arreglarlos aprendí mucho sobre el funcionamiento interno de estas estructuras.

Ventajas frente a otras estructuras

Una de las ventajas más claras fue la facilidad para insertar y eliminar elementos en cualquier posición. A diferencia de un array, donde al insertar en medio hay que mover muchos elementos, en la lista doblemente ligada solo ajusté los punteros de los nodos vecinos. Esto hizo más sencillo agregar un registro en el lugar adecuado dentro del orden. Además, a diferencia de una lista simplemente ligada, esta estructura me permitió recorrer los registros en ambos sentidos. Si necesitaba revisar los registros en orden cronológico inverso, podía recorrer la lista hacia atrás sin problema. Esa flexibilidad no es posible con una lista simplemente ligada.

Desventajas

En términos de memoria, cada nodo guarda dos referencias (siguiente y anterior), lo que implica un ligero overhead. El código se volvió más complejo por el manejo de dos punteros; esta estructura no permite acceso directo por índice, así que para encontrar un elemento a la mitad hay que recorrer la lista secuencialmente.

Conclusión personal

En conclusión, el uso de una lista doblemente ligada resultó ser adecuado para el proyecto de gestión de bitácoras por las características del problema. Técnicamente, esta estructura permitió ordenar los registros por IP, fecha y mensaje de error de manera eficiente, facilitando inserciones ordenadas y recorridos bidireccionales para impresiones o búsquedas específicas. Si bien no es la solución universal para todos los casos, en esta situación concreta la lista enlazada demostró su Re para manejar datos dinámicos y ordenar por múltiples criterios sin un costo excesivo.

Referencias

- *Doubly linked list tutorial*. (2024, febrero 23). GeeksforGeeks. https://www.geeksforgeeks.org/dsa/doubly-linked-list/
- *Operations of doubly linked list with implementation*. (2020, mayo 21). GeeksforGeeks. https://www.geeksforgeeks.org/dsa/doubly-linked-list-tutorial/