Tecnológico de Monterrey

Nombre: Rodrigo Castillo Francisco

Matricula: A01799191

Materia: Programación de estructuras de datos y algoritmos fundamentales

Profesor: Eduardo Arturo Rodríguez Tello

1- Importancia y eficiencia del uso de estructuras de datos lineales

Las estructuras de datos líneas ya sea vectores, listas enlazadas además de listas doblemente ligadas, son importantes para el procesamiento de grandes volúmenes de datos en archivos. En el caso de este ejercicio de la bitácora de intentos de ataque, donde se almacenaron y procesaron muchos registros deben de contar con una estructura de datos que sea adecuado para permitir almacenar los datos y acceder a ellos de forma secuencial cuando se realiza una operación de filtrado o búsqueda además de que con la estructura de datos correcta impacta directamente en la eficiencia de operaciones como la inserción, la eliminación y la búsqueda. Por otra parte, algunas estructuras como los vectores ofrecen acceso aleatorio de forma natica por lo que es muy útil para algoritmos de búsqueda binaria. En su contra las listas enlazadas ofrecen inserciones y eliminaciones en tiempo O (1) cuando se tiene acceso al nodo, pero sacrificando el acceso aleatorio. Es aquí donde entra en juego la parte de tener una estructura de datos ordenada y eficiente cuando existe una situación determinante.

2 ¿Por qué es preferible emplear una Doubly Linked List en este reto en lugar de una simple LinkedList

Para el procesamiento de la bitácora y la realización de búsquedas y ordenamientos en un rango de fechas, la Double Linked List resulta ser mejor porque permite recorrer la lista tanto hacia adelante como atrás lo que facilita la implementación de algoritmos que requieren moverse en ambas direcciones. Por otra parte, cuando se realizan algoritmos de ordenamiento como fue el caso de Merge Sort en una lista doblemente ligada, es importante poder actualizar tanto los punteros al siguiente nodo como al nodo anterior. Eso hace que la fusión de dos sublistas ordenadas sea más sencilla y evitar recorrer todo nuevamente la lista para actualizar los punteros. Una cosa que también considero fundamental a la hora de programar con lista doblemente ligada es que tiene una gran flexibilidad, me refiero a que si se requiere eliminar o insertar elementos en cualquier posición de la lista, una lista doblemente ligada lo realiza

en O (1) mientras que en una lista simplemente ligada recorre la lista para encontrar al nodo previo.

Por lo tanto, podría decir que en este trabajo el uso de una lista doblemente ligada mejora la eficiencia y simplifica la implementación de los algoritmos de ordenamiento y búsqueda.

3 - Complejidad computacional de las operaciones básicas

En la parte de inserción, la inserción final mediante el push_back tiene una complejidad de O (1), ya que se mantiene un puntero al final de la lista.

El impacto que tiene esta operación es que es muy eficiente cuando se cargan muchos registros, ya que cada inserción se realiza en un tiempo podría decirse constante.

En la parte del borrado, una vez localizado el nodo, se realiza en O (1) ya que se actualiza el puntero anterior y siguientes de forma inmediata.

En la parte de búsqueda, la búsqueda secuencial en una lista ligada es de complejidad O(n). Sin embargo, en este trabajo se utiliza un enfoque combinado; cuando se requiere la búsqueda binaria en una lista ordenada, se construye un vector que permite realizar la búsqueda en O (n log n) una vez que se convierte a lista.

Su impacto es que la conversión a vector tiene un costo de O(n) pero si se realizan múltiples búsquedas, el beneficio de la búsqueda binaria puede compensar ese costo inicial.

Ordenamiento (Merge Sort)

El algoritmo Merge Sort en una lista ligada tiene una complejidad de O(n log n) debido a que a la ausencia de acceso aleatorio, se utiliza recursivamente una técnica que es de división en mitades para dividir y fusionar la lista.

El impacto que tiene es que es eficiente para grandes cantidades de información de datos por lo tanto resulta adecuado para el procesamiento de bitácoras que son muy gigantes.

4. Retos al implementar algoritmos de ordenamiento y búsqueda en una Doubly Linked List y su adaptación en orden descendente.

Para mi yo creo que el mayor reto fue el manejo de punteros esto es porque por ejemplo cuando se usa en Merge Sort se debe de actualizar con mucho cuidado tanto el puntero next como el prev da cada nodo. Cuando me equivocaba provocaba muchos errores en el código además de varios minutos en estar analizando y buscando cual es el error.

Otro reto fue el implementar como se leerían los datos. Esto es porque estaba tomando mal la redirección de lo que quería que se tomara del dato por lo tanto provoco muchos ajustes en el código, así como analizar todos los prev y next de los archivos para ver si ya podía correr afortunadamente si se logró el cometido

El reto de la adaptación de la estructura ordenada descendentemente fue en una lista ordenada de forma descendente, la comparación es inversa a lo que hace un orden ascendente. Por ejemplo, cuando se fusionan dos sublistas en Merge Sort se coloca primero el nodo que tenga el valor mas alto. Eso requiere ajustar varias cosas de comparación en el algoritmo de fusión que además me equivoque muchas veces.

Otro reto fue la conversión a vector auxiliar, bueno dado que la lista en sí no tiene acceso aleatorio, se crea un vector auxiliar para aplicar la búsqueda binaria. Este proceso agrega sobrecarga de O(n) pero es compensable si el número de búsquedas es mucho.

5. Conclusión

En conclusión, el uso de estructuras de datos lineales es muy importante en varios problemas como el procesamiento de bitácoras donde el manejo de datos y la eficiencia afectan el desempeño de la solución.

En este trabajo estuvo interesante la parte de implementar la doblemente ligada porque al se vio como al emplear esta ligada con la posibilidad de realizar recorridos bidireccionales facilitaba la implementación de los algoritmos de búsqueda además de hacer mucho más eficiente el trabajo.