Reflexión

Daniel Alfredo Barreras Meraz - A01254805

Reflexionando sobre el problema del Canal de Suez inicié con el primer paso que fue reconocer el problema: un gran volumen de datos de los buques que pasan por el Canal de Suez y la necesidad de buscar y ordenar eficientemente estos datos. Este problema se agravó con el incidente del Ever Given, que resaltó la importancia de tener sistemas robustos para manejar estos datos.

Para resolver este problema, tomé en cuenta e investigué sobre varias estructuras de datos, tales como las pilas, las filas, los arreglos, árboles binarios, listas enlazadas y mapas hash.

Pilas: Las pilas son estructuras de datos que siguen el principio de LIFO (Last In, First Out), es decir, el último elemento que se agrega a la pila es el primero en ser eliminado. Esto puede ser útil en ciertos escenarios, pero no proporciona la flexibilidad necesaria para nuestro problema.

Filas: Las filas siguen el principio de FIFO (First In, First Out), es decir, el primer elemento que se agrega a la fila es el primero en ser eliminado. Al igual que las pilas, no ofrecen la flexibilidad necesaria para nuestro problema.

Arreglos: Los arreglos son una estructura de datos simple que almacena elementos del mismo tipo en ubicaciones contiguas de memoria. Aunque los arreglos permiten un acceso rápido a los elementos, no son adecuados para nuestro problema debido a su tamaño fijo.

Árboles Binarios: Los árboles binarios son una estructura de datos jerárquica que puede ser útil para almacenar datos que tienen una relación natural de “menor que” o “mayor que”. En nuestro caso, los embarques no se hacen de manera jerárquica, por lo que es mejor apuntar hacia otra estructura de datos.

Listas Enlazadas: Las listas enlazadas son estructuras de datos lineales que contienen nodos, donde cada nodo tiene un valor y un puntero al siguiente nodo en la lista. Las listas enlazadas tienen una complejidad de tiempo de O(1) para las operaciones de inserción y eliminación (cuando se conoce el nodo), y una complejidad de tiempo de O(n) para las operaciones de búsqueda. Existen dos tipos de listas enlazadas, las simplemente y doblemente enlazadas.

Mapas Hash: Los mapas hash o tablas hash permiten un acceso rápido a los elementos a través de claves únicas. Lo cual los hace ideales para hacer la conversión entre formatos de fecha (01 a jan por ejemplo).

La elección de usar listas enlazadas en lugar de otras estructuras de datos como los arrays o los árboles binarios se debió a varias razones dada la naturaleza del problema. En primer lugar, las listas enlazadas permiten inserciones y eliminaciones

eficientes, lo cual es útil cuando se manejan grandes volúmenes de datos. En segundo lugar, las listas enlazadas no requieren un tamaño fijo, lo cual es beneficioso cuando no se conoce el número exacto de elementos por adelantado. Finalmente, las listas enlazadas permiten un recorrido secuencial eficiente de los elementos, lo cual es ideal para nuestro caso donde necesitamos buscar a través de todos los elementos. Así como el uso de mapas Hash para guardar las conversiones entre meses y años a formatos de texto en lugar de solo números.

Ahora bien, dentro de las listas enlazadas, tanto simplemente enlazadas como doblemente enlazadas, son estructuras de datos útiles que ofrecen varias ventajas. Permiten inserciones y eliminaciones eficientes, lo cual es útil cuando se manejan grandes volúmenes de datos. Además, no requieren un tamaño fijo, lo cual es beneficioso cuando no se conoce el número exacto de elementos por adelantado. Finalmente, permiten un recorrido secuencial eficiente de los elementos, lo cual es ideal para nuestro caso donde necesitamos buscar a través de todos los elementos.

Sin embargo, aunque las listas doblemente enlazadas ofrecen la ventaja adicional de permitir un recorrido bidireccional a través de la lista, también tienen un costo en términos de memoria ya que cada nodo en la lista debe almacenar un puntero adicional al nodo anterior. Además, las operaciones en listas doblemente enlazadas pueden ser más complejas y potencialmente más propensas a errores debido a la necesidad de mantener dos punteros por nodo. Por lo tanto, las listas simplemente enlazadas son una mejor opción. Aunque las listas doblemente enlazadas podrían haber proporcionado alguna funcionalidad adicional, las ventajas de las listas simplemente enlazadas en términos de simplicidad y eficiencia hacen que sean una mejor opción para este problema específico.

Para resolver este problema, investigué varios algoritmos de ordenamiento y búsqueda. Entre los algoritmos que consideré estaban Ordenamiento por Mezcla (Merge Sort), Ordenamiento Rápido (QuickSort), Ordenamiento por Inserción (Insertion Sort), Ordenamiento por Selección (Selection Sort) y Ordenamiento Burbuja (Bubble Sort). Cada uno de estos algoritmos tiene sus propias ventajas y desventajas en términos de complejidad temporal y espacial.

Ordenamiento Merge es un algoritmo eficiente con una complejidad temporal de O(n log n), lo que lo hace ideal para manejar grandes conjuntos de datos. Aunque este algoritmo requiere acceso aleatorio a los elementos para poder dividir la lista por la mitad en cada paso cuando se trabaja con arrays, con las listas enlazadas, este no es el caso. De hecho, Ordenamiento por Mezcla puede implementarse para listas enlazadas con requisitos de espacio auxiliar constante.

Ordenamiento Rápido, por otro lado, tiene una complejidad temporal promedio de O(n log n), pero su peor caso es O(n^2), lo cual puede ser problemático con ciertos conjuntos de datos. Al igual que Ordenamiento por Mezcla, Ordenamiento Rápido

también requiere acceso aleatorio a los elementos, lo cual no es eficiente con las listas enlazadas.

Ordenamiento por Inserción, Ordenamiento por Selección y Ordenamiento Burbuja son algoritmos más simples con una complejidad temporal de O(n^2). Aunque son menos eficientes que Ordenamiento por Mezcla y Ordenamiento Rápido para grandes conjuntos de datos, son compatibles con las listas enlazadas ya que sólo requieren acceso secuencial a los elementos.

Después de considerar todas estas opciones, decidí utilizar **Ordenamiento Merge** para ordenar los datos debido a su eficiencia con grandes conjuntos de datos y su compatibilidad con las listas enlazadas. Aunque otros algoritmos podrían haber sido más eficientes teóricamente, las características específicas de las listas enlazadas hicieron que **Ordenamiento por Merge** fuera una opción más adecuada para este problema específico.

Por la parte de la búsqueda se analizaron específicamente la búsqueda secuencial (implementada por defecto en las listas enlazadas) y la búsqueda binaria.

La **búsqueda secuencial** es un algoritmo simple que recorre cada elemento de una lista desde el principio hasta el final hasta encontrar el elemento objetivo. Este algoritmo es especialmente útil cuando se trabaja con listas enlazadas, ya que estas no permiten un acceso aleatorio a los elementos (es decir, no puedes acceder directamente a un elemento en una posición específica sin recorrer los elementos anteriores). Por lo tanto, la búsqueda secuencial es una elección natural para las listas enlazadas.

Por otro lado, la **búsqueda binaria** es un algoritmo que busca un elemento objetivo en una lista ordenada dividiéndola repetidamente por la mitad hasta que encuentra el objetivo o hasta que la sublista es vacía. Aunque este algoritmo es más eficiente que la búsqueda secuencial (con una complejidad temporal de O(log n) en comparación con la complejidad temporal de O(n) de la búsqueda secuencial), requiere que los datos estén ordenados y que se pueda acceder a los elementos de manera aleatoria. Estos requisitos no se cumplen con las listas enlazadas, lo que hace que la búsqueda binaria sea menos adecuada para este caso.

Para concluir, puedo decir que las listas enlazadas, debido a su flexibilidad y eficiencia en la inserción y eliminación de elementos, resultaron ser la estructura de datos más adecuada para este problema. El algoritmo de ordenamiento Merge fue seleccionado por su eficiencia con grandes conjuntos de datos y su compatibilidad con las listas enlazadas. Aunque la búsqueda binaria es generalmente más eficiente, la búsqueda secuencial fue la opción preferida debido a la naturaleza secuencial de las listas enlazadas. Por lo tanto, la elección de las herramientas y métodos adecuados depende tanto de su eficiencia teórica como del contexto específico del problema.

Referencias de investigación:

GeeksforGeeks. (2023). Data Structures. Recuperado de https://www.geeksforgeeks.org/data-structures/

Interview Cake. (2023). Sorting algorithm reference, for coding interviews and computer science classes. Recuperado de https://www.interviewcake.com/sorting-algorithm-cheat-sheet

GeeksforGeeks. (2023). Searching Algorithms. Recuperado de https://www.geeksforgeeks.org/searching-algorithms/ .