Instituto Tecnológico y de Estudios Superiores de Monterrey

Act 2.3 - Actividad Integral estructura de datos lineales: Reflexiones

---

Programación de estructuras de datos y algoritmos fundamentales (Gpo 570)

Integrantes:

Noreth Sofia Villalpando Saldaña A01368579

Héctor Calderón Reyes A01350637

Campus:

Toluca y Guadalajara

Profesores:

Dr. Eduardo Arturo Rodríguez Tello

Fecha de entrega:

14 de Julio de 2023

Reflexión Hector

En la actividad 2.3 , se utiliza una lista doblemente enlazada (DLL, por sus siglas en inglés) para almacenar y manejar los registros de log desde un archivo llamado “Bitacora.txt”. Una DLL es un tipo de estructura de datos lineal que consiste en nodos, donde cada nodo contiene un campo de datos y dos punteros que enlazan al nodo anterior y siguiente en la secuencia. Esto nos permite recorrer la lista en ambas direcciones.

Hay varias razones por las que una DLL es una opción adecuada para esta tarea. Primero, puede crecer y reducirse dinámicamente en tiempo de ejecución, lo que la convierte en una solución flexible para almacenar un número potencialmente grande e impredecible de registros de log. Segundo, proporciona operaciones eficientes para agregar y eliminar nodos en ambos extremos de la lista, lo cual puede ser útil al ordenar la lista o insertar nuevos registros de log (Weiss, 2014).

Comparada con una lista simplemente enlazada, una DLL requiere más memoria porque necesita almacenar un puntero extra para cada nodo. Sin embargo, este costo adicional puede verse compensado por la mayor flexibilidad y eficiencia que proporciona una DLL. Por ejemplo, en una SLL, no es posible recorrer la lista hacia atrás, y operaciones como eliminar el nodo anterior o insertar un nodo antes del nodo actual no son tan sencillas como en una DLL.

En términos de complejidad temporal, las operaciones de inserción y eliminación en una DLL son O(1) si se conoce la posición, y O(n) si no se conoce la posición porque necesitamos recorrer la lista para encontrarla. La operación de búsqueda también es O(n) porque podríamos necesitar recorrer toda la lista en el peor de los casos.

En esta tarea, la DLL se ordena utilizando un algoritmo de ordenamiento mergesort, que divide la lista en dos mitades, las ordena y luego las fusiona. Este algoritmo es adecuado para ordenar listas enlazadas porque no requiere acceso aleatorio a los elementos y es un ordenamiento estable, lo que significa que los elementos iguales permanecen en su orden original (Weiss, 2014). La complejidad temporal del ordenamiento por mezcla es O(n log n), que es más eficiente que otros algoritmos de ordenamiento comunes como el ordenamiento de burbuja o el ordenamiento por inserción que tienen una complejidad temporal de O(n^2).

Para buscar un rango de fechas en la lista ordenada, se utiliza un algoritmo de búsqueda binaria, que reduce a la mitad el espacio de búsqueda en cada paso. Esto es mucho más eficiente que una búsqueda lineal, especialmente para listas grandes, ya que su complejidad temporal es O(log n) en comparación con O(n) para una búsqueda lineal.

En conclusión, la elección de usar una DLL en esta tarea está justificada por los requisitos específicos del problema y las características de la DLL. Sin embargo, es importante destacar que la mejor estructura de datos siempre depende de las especificidades del problema en cuestión, y lo que funciona mejor en una situación puede no ser adecuado en otra

(GeeksforGeeks, 2022). Es crucial tener un profundo entendimiento de las diferentes estructuras de datos, sus ventajas y desventajas, y la complejidad de sus operaciones para elegir la solución más eficiente. El uso eficiente de las estructuras de datos puede mejorar enormemente el rendimiento de las aplicaciones de software, haciéndolas más rápidas y receptivas, lo que lleva a una mejor experiencia de usuario.

Reflexión Noreth

Durante el desarrollo de la solución de esta actividad, tuvimos que pasar un archivo de texto a una Doubly Linked List para tener una mejor manipulación sobre los datos en ella, que fueron las fechas y horas. Se requería ordenar y buscar datos de la DLL de una bitácora brindada.

Al hacer uso de una DLL, esto nos permitió aprovechar varias ventajas clave para manejar eficientemente la bitácora. Una de las principales ventajas fue el acceso bidireccional que proporciona la DLL. Al tener enlaces tanto al nodo anterior como al siguiente, pudimos realizar búsquedas en orden ascendente o descendente según la fecha y hora de los registros. Además, la inserción y eliminación de elementos en la DLL fueron operaciones eficientes, especialmente al agregar o eliminar registros al principio o al final de la lista. Estas operaciones tuvieron una complejidad constante O(1), lo que nos brindó un rendimiento óptimo al gestionar los registros.

Otra ventaja importante de la DLL fue la posibilidad de aplicar la búsqueda binaria para mejorar la eficiencia de la búsqueda de registros. Utilizando este algoritmo de búsqueda, pudimos reducir significativamente el número de comparaciones necesarias al dividir el espacio de búsqueda a la mitad en cada iteración. Esto resultó especialmente útil considerando la potencialmente grande cantidad de registros en la bitácora.

Sin embargo, también reconocemos algunas desventajas asociadas con el uso de una DLL. En primer lugar, se requiere un uso adicional de memoria para almacenar los punteros adicionales que enlazan los nodos en ambas direcciones (GeeksforGeeks, 2023). Además, la implementación de una DLL puede ser más compleja en comparación con otras estructuras de datos lineales, como un arreglo o una lista simplemente enlazada.

En cuanto a la complejidad computacional, la DLL demostró ser eficiente para nuestras operaciones requeridas. La inserción y eliminación tuvieron una complejidad constante O(1) en los extremos de la lista, mientras que la búsqueda utilizando la búsqueda binaria se redujo a una complejidad logarítmica O(log n), que la hace realmente eficiente y rápida.

Después de contrastar sus ventajas y sus desventajas que puede tener la implementación de una Doubly Linked List en una situación problema del estilo, así como la complejidad computacional de las operaciones solicitadas, concluimos que la elección de una Doubly Linked List (DLL) fue la mejor opción para nuestra situación problema. Sus características de acceso bidireccional, eficiencia en inserción y eliminación en los extremos de la lista, y la capacidad de aplicar la búsqueda binaria para optimizar las búsquedas, nos permitieron manejar eficientemente los registros de la bitácora (OpenDSA, s.f.).

Referencias

- GeeksforGeeks. (2022). Estructuras de Datos - GeeksforGeeks. https://www.geeksforgeeks.org/data-structures/

- GeeksforGeeks. (2023). Advantages Disadvantages and uses of Doubly Linked List. GeeksforGeeks. https://www.geeksforgeeks.org/advantages-disadvantages-and-uses-of-doubly-linked-list/

- OpenDSA. (s.f.) 5.6. Doubly linked Lists — CS3 Data Structures & Algorithms. https://opendsa-server.cs.vt.edu/ODSA/Books/CS3/html/ListDouble.html

- Weiss, M. A. (2014). Análisis de Estructuras de Datos y Algoritmos en C++. Pearson.