Campus Guadalajara

Verano 2022

Reflexión Act 2.3 - Actividad Integral: Estructura de datos lineales

Fernando López Gómez | A01639715

Programación de estructuras de datos y algoritmos fundamentales

TC1031.570

Dr. Eduardo Arturo Rodríguez Tello

Fecha de entrega: 16/07/2022

Para la realización de la actividad 2.3 se utilizaron las clases bitácora y registro y los templates de Double Linked List y Double Linked list node.

Para abordar el problema sustituimos el vector dentro de la clase bitácora por una DLL de tipo registro, lo cual pudimos definir gracias a que DLL es un template. Ahora bien, el problema anterior se desarrollaba en torno al almacenamiento de registros en un vector; ahora que tenemos los conocimientos necesarios para la implementación de una lista ligada mudamos a este tipo de estructura debido a que nos permite trabajar con memoria dinámica, es decir que tenemos el control de los datos que se almacenan.

Las listas doblemente enlazadas tienen la característica de que podemos almacenar diferentes tipos de datos en diferentes lugares de la memoria disponible, no necesariamente se almacenan de manera secuencial, esto nos da la ventaja de poder hacer crecer y acortar la memoria que utiliza esta estructura. Al no estar secuencialmente guardados los datos, no hay manera de utilizar índices dentro de la lista, pero otra de las ventajas de las listas doblemente enlazadas es que "[...]no necesitan un nodo especial para acceder a ellas, pueden recorrerse en ambos sentidos a partir de cualquier nodo, esto es porque a partir de cualquier nodo, siempre es posible alcanzar cualquier nodo de la lista, hasta que se llega a uno de los extremos." (Pozo S, 2001). Es por esta misma razón que se usó una lista doblemente ligada en lugar de una lista ligada simple, porque la doblemente enlazada nos da la oportunidad de iterar en ambas direcciones mientras que la ligada simple no.

La complejidad computacional del algoritmo utilizado en el algoritmo de ordenamiento mergeSort es de nlogn, esta complejidad es una complejidad muy buena y constante al momento de utilizarlo en cualquier caso; es el algoritmo con la mejor eficiencia temporal dentro de los algoritmos de ordenamiento, lo cual hace que nuestro programa corra más rápido que si se hubiera utilizado cualquier otro método para ordenar la lista ligada.

Al eliminar el método de QuickSort como método de ordenamiento, eliminamos también el método con mayor complejidad temporal, teniendo una complejidad en su peor caso de O(n^2) y provocando así que todo el programa tenga esa complejidad. Con esa eliminación, la complejidad temporal se redujo a O(n), lo cual hace el código más eficiente.

Para la implementación del algoritmo de búsqueda contamos con 2 métodos, el secuencial y el de búsqueda binaria. La búsqueda binaria es el método con menor complejidad temporal O(n log n), lo que le ayuda al programa a correr mucho más rápido que si utilizáramos una búsqueda secuencial ordinaria con complejidad O(n)

Referencias:

Notas de clase

Pozo S. (2001) *Capítulo 5 Listas doblemente enlazadas.* Con Clase. https://conclase.net/c/edd/cap5#:~:text=Una%20lista%20doblemente%20enlazada%20es,siguiente%2C%20y%20otro%20al%20anterior.