Tecnológico de Monterrey

Actividad 2.3 Actividad Integral estructura de datos lineales

15.10.2021

—

Programación de estructuras de datos y algoritmos fundamentales
Grupo 12

Alumno

Fernando Cerriteño Magaña - A01702790

Profesor

Dr. Eduardo Arturo Rodríguez Tello

Reflexión

Durante la solución de la actividad es evidente que el uso de datos lineales es más preferible, esto es debido a que facilita la implementación de funciones como la de ordenamiento y búsqueda. Aparte de que es más fácil imaginarse a los datos una unidad segmentada que como varias unidades, permitiendo una mayor facilidad en el momento de generar soluciones en la actividad.

Para la actividad se trabajo con listas doblemente ligadas (Double Linked Lists) o Dll para abreviar. Esto es debido a que al ser doblemente ligadas permiten una manipulación mas flexible y sencilla de los datos que almacena, a diferencia de una lista ligada (Linked List) la cual solo puede avanzar a lo largo de la lista sin poder obtener el Nodo previo, lo cual es fundamental en la implementación de la función Quick Sort, debido a que se necista evaluar datos con respecto al nodo anterior y siguiente.

En la implementación del método de ordenamiento se presento con un reto adicional, la conversión a una forma iterativa del método QuickSort, esto con el fin de mejorar el código, ya que realiza el método principal en un bucle while en vez de llamarse a si mismo, ahorrando tiempo.

La implementación del método QuickSort de forma iterativa represento un gran desafío, debido a que no sabía por dónde empezar, por lo que primero empecé ordenando la lista de manera recursiva, para poder observar el que tantas modificaciones se le tienen que hacer al algoritmo que usa vectores. Una vez implementada el algoritmo en forma recursiva, se procedió a realizarlo de manera iterativa, basándome en el código presentado en el artículo “Iterative Quick Sort” contribuido por el usuario/usuaria rathbhupendra en la página de GeeksforGeeks. En la versión iterativa del método se hace el uso de Pilas (Stacks) para poder almacenar datos de forma temporal y pode compararlos, la dificultad del código llego cuando se tenía que implementar un comparador entre el nodo previo a una partición, con respecto al primer nodo de la pila y posteriormente, con respecto al último nodo de la pila. El problema radicaba en que el algoritmo aportado por rathbhupendra, utilizaba enteros para realizar la comparación, por lo que se opto

por generar una nueva función de tipo booleana, la cual recibe dos nodos, y compara si el segundo nodo dado se encuentra después que el primero, si lo está, se regresa un true, de lo contrario significa que se encuentra antes que el nodo, por lo que regresa un false. Con este método se logro ejecutar los condicionales a la par que se ordena la bitácora.

Para el algoritmo de búsqueda binaria resulto ser mas fácil, debido a que se tenían las funciones necesarias para realizarlo en la DLL, lo único que se tuvo que hacer fue cambiar el dato de entrada vector por una DLL y utilizar la función getData para obtener los datos necesarios.

Creo que esta actividad a representado un verdadero reto, principalmente por la implementación de la iteratividad en vez de recursividad, no obstante, me he logrado llevar buenos conocimientos de implementación, asimilación y de búsqueda de archivos útiles. En cuanto a la iteratividad, personalmente creo que es mejor la implementación de recursividad, debido a que la recursividad es comúnmente más consistente que la iteratividad, aparte de que permite reducir el tamaño del algoritmo de forma significativa, permitiendo ver a un código más limpio y elegante, no obstante, la implementación de la iteratividad me ayudo a ver más en detalle el cómo funciona el método de ordenamiento, además de que me ayudo a ver más formas del cómo puedo usar los datos de la DLL.

Bibliografía

Briceño, G. (2017, 5 octubre). Recursividad o Iteración. Club de Tecnología. Recuperado 15 de octubre de 2021, de https://www.clubdetecnologia.net/blog/2013/recursividad-iteracion/#:%7E:text=Usualmente%2C%20el%20c%C3%B3digo%20con%20recursividad,en%20el%20m%C3%A9todo%20con%20iteraciones.&text=La%20recursividad%20es%20m%C3%A1s%20clara,problemas%20complejos%20en%20piezas%20manejables.

Diferencia entre iteración y recursividad. (s. f.). CódigoFacilito. Recuperado 15 de octubre de 2021, de https://www.codigofacilito.com/articulos/articulo_16_10_2019_16_22_35

GeeksforGeeks. (2021a, septiembre 6). Iterative Quick Sort. Recuperado 15 de octubre de 2021, de https://www.geeksforgeeks.org/iterative-quick-sort/

GeeksforGeeks. (2021b, septiembre 21). QuickSort on Doubly Linked List. Recuperado 15 de octubre de 2021, de https://www.geeksforgeeks.org/quicksort-for-linked-list/