**Instituto Tecnológico y de Estudios Superiores de Monterrey**
Escuela de Ingeniería y Ciencias


Programación de estructuras de datos y algoritmos fundamentales
Gpo 850

# Después de clase | Tarea individual: Act-Integradora-2 Estructuras de Datos Lineales

**Docente:** Dr. Eduardo Arturo Rodríguez Tello

Jesús Alonso Galaz Reyes (A00832930)

**21 de abril del 2023**

**Reflexión**

Bueno, en este código se trata un problema de ordenamiento y búsqueda de registros en una bitácora. La importancia y eficiencia de las estructuras de datos lineales en este tipo de problemas radica en que pueden almacenar y acceder a los elementos de manera secuencial y se adaptan a los requerimientos específicos de cada situación.

En el código, se eligió usar una Doubly Linked List (Lista Doblemente Enlazada) en lugar de una Linked List (Lista Enlazada Simple) porque ofrece ciertas ventajas en cuanto a la manipulación de los elementos. Con la Doubly Linked List, es más fácil recorrer la lista en ambas direcciones (hacia adelante y hacia atrás) y realizar inserciones y eliminaciones de elementos sin perder la referencia a los nodos adyacentes.

En cuanto a la complejidad computacional, la inserción y el borrado en una Doubly Linked List son O(1) cuando se tiene acceso directo al nodo de interés, mientras que la búsqueda en la lista es O(n) en el peor de los casos, ya que es necesario recorrer la lista en secuencia. Estas operaciones afectan el desempeño de la solución, pero hay que considerar que, en este caso, el foco principal está en el ordenamiento y la búsqueda de un rango de registros.

El código también compara dos algoritmos de ordenamiento, Quick Sort y Merge Sort, ambos con una complejidad temporal promedio de O(n log n). Aunque ambos algoritmos tienen la misma complejidad temporal, se observa que uno de ellos puede tener un mejor rendimiento en tiempo de ejecución, lo cual podría deberse a factores como el caso particular de los datos o la implementación específica en el código.

En resumen, la elección de una Doubly Linked List en este problema permite un manejo más eficiente de los elementos durante el proceso de ordenamiento y búsqueda. La comparación entre los algoritmos de ordenamiento sirve para identificar cuál de ellos se adapta mejor a la situación específica y ofrece un mejor rendimiento.