Luis Cano Irigoyen - A00827178

Act2.3 Reflexión

A lo largo del bloque estuvimos trabajando con un nuevo concepto para mí, las Linked Lists. Antes de eso nunca se me había ocurrido el realizar listas con apuntadores, no había visto la necesidad, pero ahora que la conozco estas listas de nodos junto a todas sus funciones empiezo a entender más su utilidad y aplicaciones, como el poder asignar memoria dinámicamente. De igual manera las linked lists ofrecen métodos muy eficientes, como el de inserción y eliminación de data en cierto lugar de la lista, los cuales tienen una complejidad de tan solo O(n) para encontrar el lugar de inserción y una vez con la posición correcta de solo O(1) . Adicionalmente, el uso de Nodos en una lista permite la creación de stacks y queues, los cuales pueden tener muchas aplicaciones en el mundo real.

En la situación problema se nos presentó la necesidad de utilizar Doubly Linked Lists, las cuales adicional al apuntador a la next data de cada nodo, contaban con un apuntador a la previous data, ofreciendo aún más eficiencia en sus procedimientos, y permitiendo utilizar métodos como queue de manera más sencillas en la Lista de Nodos.

Una observación que realicé a lo largo de la actividad fue la duración de el ordenamiento de la lista. Siendo una doubly linked list de 16807 datos, a esta le tomó bastante tiempo el reordenar todos sus Nodos con sus apuntadores al anterior y siguiente elemento, a pesar de haber sido ordenado con un algoritmo recursivo, Ordena Merge, el cual es más rápido que los otros ordenamientos iterativos para cantidades de datos muy altas. Originalmente, utilizando un método de addBack(T data) en el ordenamiento por emrge, me tomó aproximadamente entre 43.8 a 45.4 segundos el que se almacenen los datos en su struct, que se ordenen todos estos, y que sean impresos en orden en un archivo de salida. Más tarde, conforme fui modificando el código a mejor y agregue el método de queue a este proceso, logré que el tiempo bajara a aproximadamente de 41.6 a 42.5 segundos. Aún así, el tiempo de ordenamiento para una doubly linked list de muy alta cantidad de datos continúa siendo bastante alto.

En situaciones problemas de esta naturaleza definitivamente es bueno conocer las diferentes estructuras de datos lineales que se pueden utilizar, unas serán más eficientes que otras, y con más de 16mil datos el ser un poco más eficientes puede llegar a salvar varios segundos. Cada estructura de datos tendrá sus propias ventajas y desventajas, las cual es necesario conocer junto al conocer que tipos de datos se tienen, para elegir la más adecuada a la situación, realizando soluciones eficaces y óptimas para los problemas.