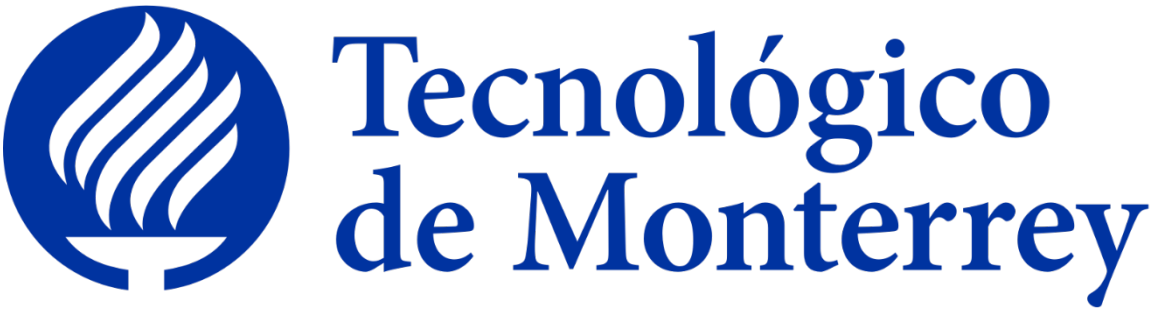
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey



Programación de estructura de datos y algoritmos fundamentales

TC1031, Grupo 601

Nombre del profesor: Dr. Eduardo Arturo Rodríguez Tello

Actividad 2.3 – Reflexión personal

Samuel Acosta Ugarte | A00833547

26 de abril 2022

Reflexión

A lo largo de esta actividad, aprendí y reforcé mi conocimiento en muchas de las áreas de la programación. En la actividad anterior realizamos un programa muy desorganizado, no orientado a objetos, que, a pesar de ser funcional, era muy ineficiente ya que no usábamos una estructura de datos bien definida. En esta actividad se realizo un programa orientado a objetos, modular contando con estructuras de datos bien definidas.

En esta actividad se utilizaron listas doblemente ligadas para la solución de este problema, una estructura de datos lineal. En la actividad anterior, usamos muchos vectores para separar la información, en un total de 5 vectores con la misma longitud de 16807 elementos. Para ordenar el vector anteriormente, realizamos un algoritmo de ordenamiento de complejidad O(n2) sumamente ineficiente, además para poder hacer uso de la lista ordenada, se necesitaba concatenar la información y para eso se hacían 2 búsquedas, una binaria y otra secuencial O(n2) también muy ineficiente.

La ventaja de poder usar listas ligadas es que podemos crear una lista de elementos de cualquier tipo, incluyendo objetos de clases creadas por el usuario, estos objetos pueden contener todos los atributos que necesitamos sin tener la necesidad de crear más listas y evitar realizar más búsquedas. Usando esa ventaja, se creó una estructura de datos llamada Registro que cuyos objetos tienen todos los atributos de un registro dentro del objeto (mes, día, hora, minuto, segundo, ip, puerto, error). Usando una sobrecarga de operadores para objetos de tipo registro, uno puede con mayor facilidad evitarse las múltiples conversiones y procesamiento a la hora de implementar algoritmos, que si uno trabajara con un vector de tipo string.

A pesar de que uno pudiera haber usado una lista ligada en lugar de una lista doblemente ligada, usar una lista doblemente ligada nos ayuda más. Con una lista doblemente ligada, uno tiene mas control de el manejo de los datos que contiene la lista, con el importantísimo apuntador que apunta hacia el nodo anterior. Usando esto, uno puede reducir la complejidad significativamente de O(n) a O(1), en algunos casos. Por ejemplo, en el método para particionar la lista a la hora de ordenar, se omite tener que buscar secuencialmente hasta tener los 2 apuntadores en el lugar correcto para saber cual es el elemento antes del pivote. Sin embargo, incluso los métodos básicos como addFirst y addLast, tienen ventajas de poder tener un apuntador hacia atrás.

A continuación, se muestra una tabla de la complejidad de los métodos actualizados contra los métodos anteriormente usados:

Método	**Complejidad**
SequentialSearch	O(n)
BinarySearch (Nuevo)	O(log n)
SwapSort	O(n2)
QuickSort (Nuevo)	O(log n)

A continuación, otros la complejidad de algunos de los otros métodos usados:

Método	Complejidad
Bitacora::print()	O (n)
Sobrecarga de operadores en clase registro	O (1)
Registro::getAll()	O (1)
Bitacora () constructor	O (n)

Al tener el código modular y toda la información estructurada, uno puede fácilmente crear modificaciones y reutilizar código sin tener que empezar de nada. Me dio mucho gusto poder entender lo que significa un programa orientado a objetos por completo. Además, esto me ayudara mucho a la hora de realizar las siguientes actividades que vienen y realizarlas de una mejor manera.