Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación de estructuras de datos y algoritmos fundamentales

Act 6.2 - Reflexión Final de Actividades Integradoras de la Unidad de Formación TC1031

Salvador Alejandro Gaytán Ibáñez A01730311

ITC

30 de noviembre del 2020.

Reflexión:

A lo largo del semestre se nos fue requerido realizar distintas actividades tomando como base un conjunto de datos con distintas características que simulaban errores de distintas direcciones IPs con sus redes, host, fechas, motivo de fallo, entre otras cosas. Con esta información implementamos distintas estructuras de datos a la par que las íbamos viendo en clase, por lo cual desde la primera actividad empezamos a trabajar con las subcompetencias SICT0301, SICT0302 y SICT0303 en nivel B, donde la SICT301 habla sobre el desarrollar la capacidad de integrar conocimiento de las ciencias de la ingeniería para reconocer variables que tienen desviaciones sobre el estado normal de funcionamiento de sistemas computacionales, donde las variables con desviaciones eran formadas dependiendo de cada uno de los casos específicos a analizar, por ejemplo, cuando se implementó primeramente un método X para leer los datos de la bitácora que los contenía para posteriormente en las últimas 2 actividades integradoras cambiar el formato de la misma, teniendo de esta manera que implementar un método distinto.

Asimismo, la subcompetencia SICT0302 describe que deberíamos de ser capaces de integrar información disponible, modelos seleccionados y herramientas de la ingeniería para obtener soluciones de problemas asociados con tecnologías de la información así como identificar información que cumpla con criterios de calidad para la generación de soluciones al problema planteado, donde esta subcompetencia la desarrollamos en todas las actividades integradoras ya que a pesar de tener la misma información disponible (aunque en diferente formato) era necesario integrarla con los modelos requeridos por la actividad, desde listas ligadas en la primera actividad hasta Hash Tables, probándolas con distintos testcases para asegurarnos que la calidad del código presentado se atenga tanto a los requerimientos de la actividad como a los estándares de programación.

Finalmente, con la subcompetencia SICT0303, adquirimos la habilidad de integrar conscientemente en las acciones el análisis de priorización de soluciones basadas en criterios de significancia, completitud y optimalidad, siendo esta competencia a mi parecer una de las más importantes ya que como ingenieros de software es nuestro trabajo el buscar siempre soluciones significativas, completas y sobre todo óptimas que permitan no solo resolver el problema planteado sino permitir reusar del código o escalarlo de ser necesario.

Debido a que en todas las actividades integradoras se nos requería hacer cosas distintas dado el mismo conjunto de datos, hablar de eficiencia es un tema complicado, sin embargo, a mi parecer, una buena métrica para saber cual algoritmo es más eficiente es analizarlos por su complejidad computacional ya que esto da una idea general del tiempo

de ejecución del algoritmo en su peor escenario. Basándome en esta métrica me atrevo a decir que el árbol binario de búsqueda sería el más eficiente para buscar un dato en específico ya que su complejidad es de O (log n), sin embargo, si se va a hacer uso constante de los datos con los valores más altos, la mejor opción sería un heap, si lo que quisiéramos es seguridad de los datos se podría usar una HashTable con un algoritmo de Hasheo de la familia SHA 2. Entonces no se puede dar una respuesta definitiva a cuál es el más eficiente ya que, como en toda situación, la eficiencia de la estructura de datos depende de para qué la quieras utilizar.

Si me dieran a elegir libremente las estructuras de datos para las distintas actividades, utilizaría los árboles binarios de búsqueda para los casos en lo que se tuviera que encontrar un dato único, para hacer ordenamientos de datos a partir de un número con sentido de mayor o menor, usaría el método Merge Sort ya que es uno de los que tiene la menor complejidad, y para actividades donde me pidieran encontrar las IPs con mayor ocurrencia, lo que yo haría seria ordenarlas en un árbol binario de búsqueda donde el atributo que defina en donde van las IPs fuera el número de ocurrencias de la misma, formando una llave única primero con las ocurrencias y después con la IP para evitar choques con la estructura, (ej; 3.196.45.55.582, donde el primer número antes del punto es el número de ocurrencias y los demás son la IP).

En la unidad de formación TC1031 desarrollé distintas subcompetencias que estoy seguro de que me serán de gran utilidad a lo largo no solo de la carrera sino también de la vida laboral ya que el poder manejar la misma información disponible de distintas maneras dependiendo de las necesidades del programa es algo muy valioso, no solo para poder resolver el problema en sí, pero también para hacerlo de la manera más eficiente, ordenada y escalable que no comprometa eficiencia con funcionalidad.