Act 3.4 - Actividad Integral de BST
Carlos Arturo Ferat Torres | A01739190

La situación problema de la Act 3.4 volvió a requerir clasificar datos de una bitácora que lleva registro de entradas a un servidor, en este caso con árboles. Clasificar datos con árboles es extremadamente eficiente pues de forma ideal el tiempo de inserción y búsqueda es de $O(\log n)$. Existen múltiples tipos de árboles, aunque en el caso de la situación problema donde resulta necesario clasificar y ordenar las entradas de bitácora, únicamente los árboles de búsqueda pueden ser usados, en particular los arboles binarios que permiten comparar y ordenar de forma eficiente dos datos, el que entra con algún otro ya almacenado, múltiples veces hasta encontrar su lugar.

De los arboles binarios, aun hay más, pues se puede ordenar tal cual como entran los datos, o se puede implementar el tipo de árbol que usamos, llamado *AVL* (Adelson-Velski & Landis) el cual no solo ordena el dato al entrar, sino que posteriormente reordena el árbol de tal manera que siempre quede balanceado. Se decidió usar un árbol *AVL* ya que prevé los peores casos posibles del árbol binario y por ello garantiza una búsqueda e inserción en tiempo O $(\log n)$, pues en el árbol binario si siempre entrasen datos ordenados de mayor a menor lo que nos quedaría es una lista ligada, arruinando por completo la eficiencia de búsqueda que otorgan los árboles quedando un caso $O(n)$, lo no ocurre en los *AVL*. En específico tanto buscar como ordenar en un *AVL* es de complejidad $O(\log_2 n)$ la base dos se debe a que cada que se va avanzando se dividen a la mitad la cantidad de datos en los que buscar.

Dado que ahora la estructura de datos es la que por defecto ya ordena los datos, no fue necesario implementar algoritmos de ordenamiento de ningún tipo, insertar datos es al mismo tiempo ordenarlos, lo que lo vuelve extremadamente eficiente. Aquí lo único que ocurre es la comparación del dato de entrada en la búsqueda de donde insertarlo, lo cual como en cualquier otro tipo de ordenamiento ocurre en tiempo constante $O(1)$.

Por último, planteando brevemente la pregunta: *¿Cómo podrías determinar si una red está infectada o no?* Este tipo de ordenamiento permitiría de forma un poco rudimentaria identificar alguna red infectada, pues los árboles tienen la característica de que no pueden repetir entradas, sin embargo, si se tuviese una sola IP que está intentando entrar múltiples veces, pero probando puertos diferentes, entonces sí que aparecería en el árbol, ya que la entrada de bitácora seria distinta (el puerto y hora cambian) y como se compara por puerto van a quedar agrupadas, quedando prácticamente juntas y sería muy fácil observar que es una sola IP la que esta intentando vulnerar la seguridad de la red.