Tecnológico de Monterrey

Reflexión individual – Importancia y eficiencia de los árboles de búsqueda binaria

Franco De Escondrillas Vázquez | A01739410

Ingeniería en Tecnologías Computacionales

Dr. Daniel Pérez Rojas

TC1031.601

28 de octubre, 2025

Introducción

En fases anteriores de esta investigación, se sugirió el uso de algunas estructuras de datos para trabajar con los logs de acceso de usuarios. Dado que en una red de computadoras se genera una enorme cantidad de este tipo de datos, es indispensable encontrar una estructura de datos eficiente que sea capaz de realizar operaciones específicas con una baja complejidad.

Algunas acciones típicas que se llevan a cabo en este tipo de contextos son la inserción, búsqueda, ordenamiento y filtrado de logs. Especialmente si se trabaja con una red en la cual se sospecha que existen nodos infectados, tener acceso veloz a los datos permite implementar un sistema de seguridad lo suficientemente rápido como para proteger la red.

En un ejemplo anterior se usaron listas doblemente ligadas para manejar las entradas. Aunque la inserción y ordenamiento de información fue relativamente buena, el único tipo de búsqueda posible es el secuencial. Aunque una complejidad de $O(n)$ no es crítica, si se busca una cantidad k de datos, entonces la complejidad temporal aumenta a $O(kn)$, que resulta peligroso en este contexto. Por ello, en este reporte se analizará el desempeño de una estructura de datos diferente: el árbol de búsqueda binaria (BST).

Complejidad algorítmica de un BST

Inserción ordenada | O(log₂n)

A primera vista, insertar en un árbol de búsqueda binaria parece menos eficiente que insertar al final de una lista ligada $O(1)$. Sin embargo, cualquier tipo de inserción en un BST se hace de forma ordenada, por lo que una vez que el elemento se inserta no es necesario ordenar el conjunto de datos. Comparado con la inserción ordenada de la lista ligada $O(n^2)$ o la inserción simple y mergesort $O(n\ log_2 n)$ esta resulta una mejora significativa.

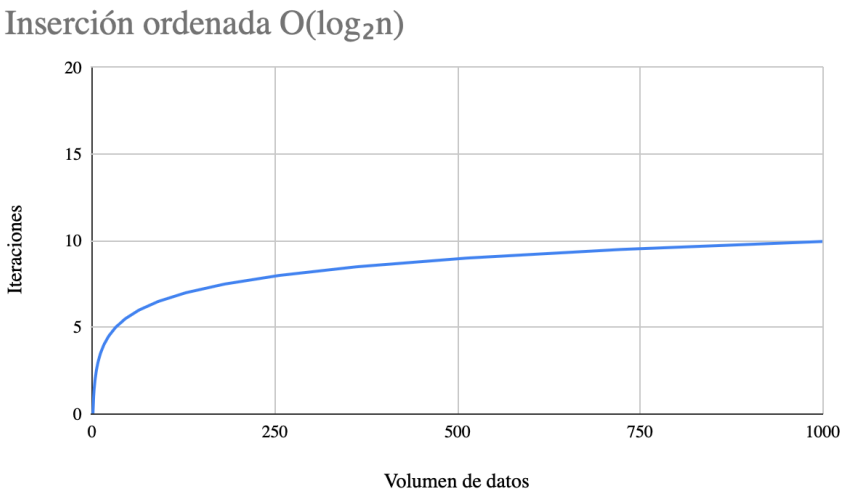

Figura 1. Complejidad inserción ordenada

Búsqueda binaria | O($\log_2 n$)

Una segunda mejora es que la búsqueda de elementos se realiza de forma binaria, lo que tiene una complejidad temporal mucho mejor que la búsqueda secuencial de las listas doblemente ligadas. El efecto negativo de esta forma de almacenar los datos es que, para hacer una impresión de los datos ordenados, se debe generar un recorrido de tipo inorden, que tiene como complejidad $O(log_2 n)$ en vez de $O(n)$.

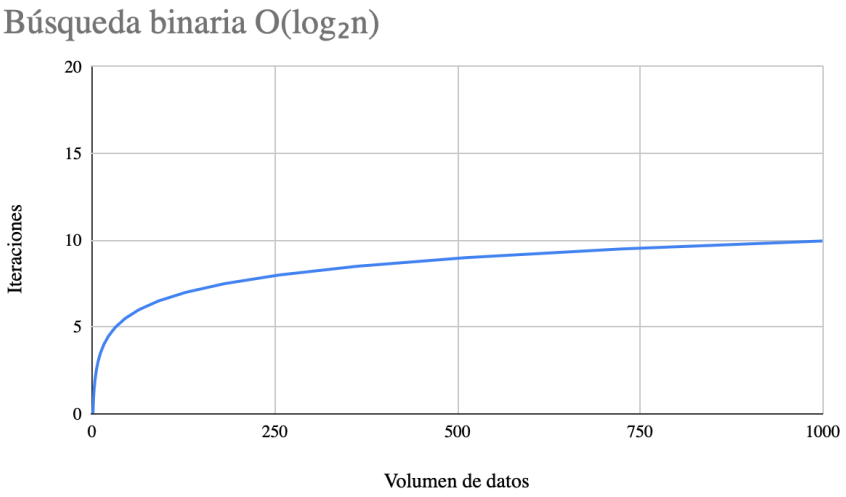

Figura 2. Complejidad de la búsqueda en un BST

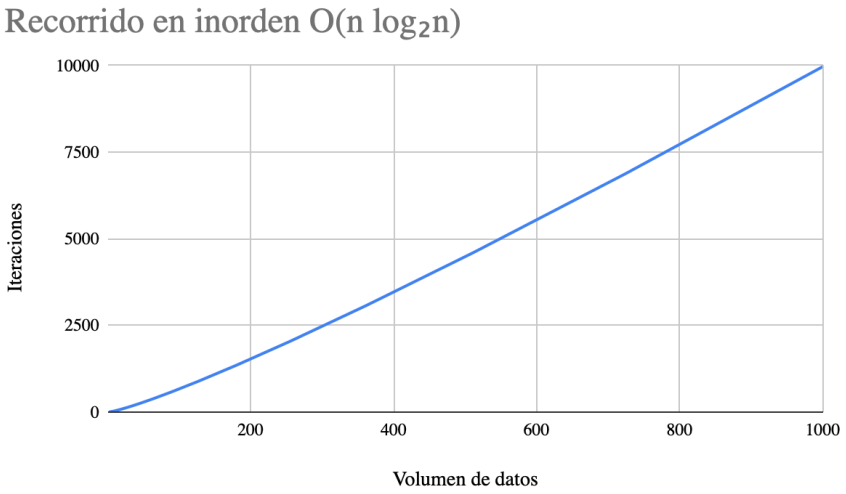

Figura 3. Complejidad del recorrido en inorden

Rotaciones AVL

Si bien los BST pueden tener las propiedades y complejidades anteriormente descritas, esto depende de que el árbol de búsqueda que se genere se encuentre bien distribuido. Si en una estructura de este tipo se insertan valores en orden (ascendente o descendente), entonces indirectamente se generará una lista ligada, donde todos los nodos desde la raíz solo contengan elementos hacia uno de sus lados. Por ello, es sumamente importante introducir rotaciones AVL para evitar que la diferencia de longitudes de los hijos de cualquier nodo supere la unidad (es decir, que el árbol se mantenga en equilibrio).

Un ejemplo claro de esto es que, cuando se desarrolló la práctica con una bitácora de 2^{18} datos, la altura del árbol sin rotaciones fue de 45 (2.5 veces $\log_2 n$), mientras que al introducir el AVL esta se redujo a 21 (prácticamente $\log_2 n$).

Conclusión e impacto en la situación problema

Detección de una red infectada

La estructura de datos con la que se trabajó en esta fase provee un enfoque adecuado para trabajar con logs de acceso de usuario. Esto, a su vez, permite analizar el conjunto de datos en busca de accesos sospechosos, repetitivos o denegados que permitan identificar nodos que se encuentren posiblemente infectados.

Sin embargo, esta información no es útil para prevenir la infección de otras computadoras o determinar si una red se encuentra completamente infectada. Para este propósito en particular, es necesario representar la red de conexiones para poder analizar la cercanía de un nodo aparentemente sano de otros infectados. Dado que un BST no permite este tipo de representaciones, en una siguiente etapa será necesario implementar una estructura de datos diferente que permita recorrer la red, identificando las zonas infectadas y la disposición de las conexiones.

En caso de que existan zonas críticas infectadas o que los nodos infectados se comuniquen con toda la red, entonces se puede valorar la red como infectada.