Programación de estructuras de datos y algoritmos fundamentales (Gpo 602)

Act 3.4 - Actividad Integral de BST (Evidencia Competencia)

investigación y reflexión de la importancia y eficiencia del uso de BST en una situación problema de esta naturaleza

Fecha

28/10/2025

Profesor

Daniel Perez R

Alumno

José Luis Gutiérrez Quintero– A01739337

Reflexión sobre el uso de BST en análisis de bitácoras

En esta actividad trabajamos con un archivo de bitácora que contiene registros de accesos asociados a diferentes direcciones IP. La tarea fue organizar toda esa información en una estructura tipo Árbol Binario de Búsqueda (BST), donde la clave principal fue la IP. Al trabajar con los datos entendí por qué este tipo de estructura es bastante útil en problemas donde se necesita buscar, insertar o ordenar datos de manera eficiente.

Un BST permite que, si está balanceado, tanto las búsquedas como las inserciones y los recorridos se puedan hacer en O(log n). Esto significa que, aunque el archivo tuviera miles de líneas con diferentes accesos, la búsqueda no se vuelve lenta conforme crece la cantidad de datos. Si hubiéramos utilizado una lista o un vector sin ordenar, tendríamos que recorrer todos los elementos cada vez que quisiéramos buscar una IP, lo cual sería O(n) y terminaría siendo muy ineficiente.

Algo interesante también es cómo definimos el criterio para ordenar las IPs. Aunque una IP está compuesta por números separados por puntos, se tiene que convertir en una representación numérica o comparar segmento por segmento para que el BST pueda ordenarlas correctamente. Además, si una IP se repetía, usamos la fecha y la hora como segundo criterio. Esto permitió que el árbol mantuviera un orden lógico completo, no solo de dirección IP, sino también del momento en el que ocurrió el acceso.

Al buscar las cinco IPs más altas, el recorrido es suficiente para obtenerlas sin necesidad de ordenar toda la estructura nuevamente. Si los datos hubieran sido manejados con un algoritmo de ordenamiento tradicional, tendríamos que ordenar toda la lista, aunque solo quisiéramos cinco valores, lo que sería menos eficiente.

¿Cómo ayuda esto en detectar si una red está infectada?

La idea principal es que cuando una red está siendo atacada o está infectada con algún malware, se generan patrones de acceso anormales. Por ejemplo:

Una misma IP conectándose demasiadas veces.

Varias IPs desconocidas intentando acceder al sistema.

Accesos registrados en horas inusuales.

Teniendo los registros almacenados en un BST, es posible identificar rápidamente cuáles son las IPs más activas o cuáles aparecen más arriba en el árbol (por frecuencia o valor). Con esta información se puede hacer una detección temprana, antes de que el daño sea mayor.

Si imaginamos esto en un entorno real, la eficiencia no es un lujo, sino una necesidad. Si la red está bajo ataque, no hay tiempo para “recorrer todo a ver qué está pasando”. Necesitamos una estructura que responda rápido, y el BST, gracias a su organización jerárquica, lo permite.

Conclusión

Más allá de simplemente almacenar y ordenar datos, el uso de un BST la elección de una estructura de datos puede cambiar por completo la manera en que resolvemos un problema. No se trata solo de programar “algo que funcione”, sino de pensar en la eficiencia y escalabilidad. En problemas de redes y seguridad, donde la velocidad de respuesta es clave, una estructura como el BST puede ser la diferencia entre detectar un ataque a tiempo o darnos cuenta demasiado tarde.

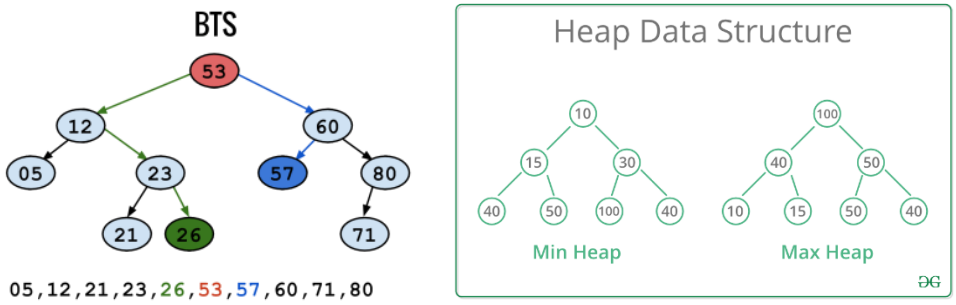

Figure 1. Binary trees and Heap data Structure

Referencias

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. *Introduction to Algorithms*. MIT Press.

- "Binary Search Trees." *GeeksforGeeks*. https://www.geeksforgeeks.org/binary-search-tree-data-structure/