Tecnológico de Monterrey

# Programación de estructuras de datos y algoritmos fundamentales

## Act 3.3 - Actividad Integral de BST

Maestro: Dr. Eduardo Arturo Rodríguez Tello

Ana Regina Rodríguez Múzquiz | A01286913

**Enero 2026**

## *Reflexión – Actividad 3.3*

En esta actividad trabajé con una bitácora que tiene un montón de registros de accesos a una red. Al principio parece solo un archivo gigante con fechas, horas e IPs, pero la idea no era solo leerlo, sino sacar información útil de ahí, como saber qué IPs se conectan más y si eso puede significar algo raro en la red.

Lo primero que tuve que hacer fue ordenar toda la bitácora por IP, sin tomar en cuenta el puerto. Aquí me di cuenta de que no cualquier algoritmo sirve. Por ejemplo, algoritmos como Bubble Sort o Insertion Sort sí ordenan, pero son lentísimos cuando hay muchos datos porque su complejidad es de $O(n^2)$. O sea, con un archivo grande sería súper lento.

También están otros algoritmos como Quick Sort y Merge Sort, que ya son más rápidos y trabajan en O(n log n). El problema con Quick Sort es que en el peor de los casos se puede volver lento, y Merge Sort necesita memoria extra. Por eso, en esta actividad usamos Heap Sort, que también es O(n log n) pero es más estable y no depende de cómo vengan los datos. A parte, usa una estructura jerárquica llamada heap, lo cual va muy de la mano con lo que se pedía en la actividad.

Una vez que la bitácora quedó ordenada por IP, contar los accesos fue mucho más fácil, porque todas las entradas de la misma IP quedaron juntas. Literal fue recorrer el vector una sola vez y contar cuántas veces se repetía cada IP.

Después, para saber cuáles eran las IPs con más accesos, usé un Binary Heap, implementado con una priority_queue. Aquí es donde se nota por qué esta estructura es tan útil. El Binary Heap está hecho justo para manejar prioridades, entonces sacar la IP con más accesos es súper rápido. Operaciones como meter un dato (push) o sacar el más importante (pop) cuestan O(log n), y ver cuál es el más grande (getTop) es O(1), o sea, super rápido.

Aquí también entendí por qué no conviene usar un BST para este problema. Aunque un árbol binario de búsqueda puede funcionar bien para buscar datos, no es tan directo cuando lo que quieres es saber cuál es el mayor todo el tiempo. Además, si el árbol no está balanceado, el rendimiento puede empeorar bastante. En cambio, el Binary Heap siempre mantiene la prioridad clara, por eso es mejor opción cuando solo te importa el “top”.

Sobre la pregunta de cómo saber si una red está infectada, yo creo que una forma sencilla sería justo analizar este tipo de bitácoras. Por ejemplo, si una IP tiene muchísimos accesos en muy poco tiempo, o muchos intentos fallidos de inicio de sesión, eso puede ser una señal de ataques como fuerza bruta o intentos de hackeo. Si varias IPs muestran comportamientos raros, ahí ya se prende el foco rojo. Con estructuras como los heaps, detectar estas IPs es mucho más rápido y eficiente.

Puedo decir que esta actividad me ayudó a entender que las estructuras de datos no son solo teoría o algo que se memoriza para un examen. Sirven para resolver problemas reales y para que los programas sean más rápidos y útiles. Usar Heap Sort y Binary Heap hizo que todo el proceso fuera más eficiente y claro, y me quedó muy claro que elegir bien el algoritmo y la estructura de datos hace toda la diferencia.

**Referencias**


GeeksforGeeks. (2025f, diciembre 22). *Heap sort*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/heap-sort/

GeeksforGeeks. (2026, 19 enero). *Binary heap*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/binary-heap/

Prezi, A. C. O. (s. f.). *La importancia de la estructura de datos del «árboles»*. prezi.com. https://prezi.com/p/acmpqhkzx6im/la-importancia-de-la-estructura-de-datos-del-arboles/

Piper. (2024, 9 enero). *What's the difference between a binary search tree and a binary heap?* Computer Science Stack Exchange. https://cs.stackexchange.com/questions/27860/whats-the-difference-between-a-binary-search-tree-and-a-binary-heap#:~:text=Advantage%20of%20BST%20over%20binary,O(log(n))%20.

*Why is binary heap preferred over BST for priority queue?* (2015, septiembre 7). GeeksforGeeks. https://www.geeksforgeeks.org/dsa/why-is-binary-heap-preferred-over-bst-for-priority-queue/

GeeksforGeeks. (2025a, julio 23). *Difference between Binary Search Tree and Binary Heap*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/difference-between-binary-search-tree-and-binary-heap/