Tecnológico de Monterrey

Campus Monterrey

Reflexión Individual

Act 3.3 - Actividad Integral de BST (Evidencia Competencia)

Paula Guerrero Martínez
A00839099

Dr. Eduardo Arturo Rodríguez Tello
Programación de estructuras de datos y algoritmos fundamentales
(Gpo 573)

28 de enero del 2025

En la actividad 3.3 desarrollamos una aplicación la cual analiza el archivo de bitácora, ordenando los registros por dirección IP y determinando cuáles registros tienen mayor número de accesos; para conseguir esto empleamos estructuras de datos jerárquicas como heap sort (ordenas) y binary heap (recuperar valores max). Utilizar estas estructuras resulta muy útil cuando se trabaja con mucha información, optimizando el tiempo de ejecución y asegurando que el sistema pueda mostrar eficiencia.

Importancia y eficiencia del uso de estructuras de datos jerárquicas en una situación problema de esta naturaleza

Las estructuras de datos jerárquicas (Árboles y Heaps) organizan la información de manera no lineal, permitiendo realizar operaciones de búsqueda, inserción o eliminación de una manera más eficiente que las estructuras lineales. En la actividad 3.3 binary heap fue de gran utilidad ya que siempre mantiene el elemento mayor en la raíz permitiendo recuperar ese elemento de manera constante.

Estas estructuras son constantemente utilizadas en programas donde se tiene que obtener un valor máximo o mínimo de manera repetida. De acuerdo a GeeksforGeeks el binary heap es una estructura completa que cumple con la propiedad heap, garantizando que cada nodo padre sea mayor que sus hijos; por eso el heap es óptimo para esta actividad y la búsqueda de IPs con mayor número de accesos.

El ordenamiento de la bitácora se realizó con heap sort, que construye un heap partiendo de los datos y luego extrae el valor máximo repetidamente para obtener la lista ordenada; También heap sort garantiza una complejidad de O(n log n) en el mejor, promedio y peor caso.

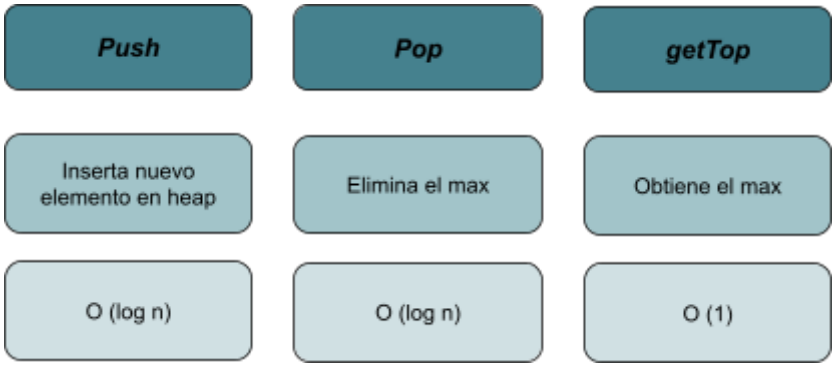
¿Por qué en la solución de este reto es preferible emplear un Binary Heap y no un BST?

Aunque un BST (binary search tree) es una estructura jerarquizada, no siempre es la mejor opción ya que no garantiza que el árbol esté balanceado, y si el árbol no está balanceado puede degenerar una lista causando que las operaciones de inserción, búsqueda y eliminación tengan complejidad de O(n).

Para evitar la degeneración de la lista se tendría que utilizar árboles balanceados como AVL o Red-Black Trees que son más complicados de implementar; en cambio el binary heap ayuda a garantizar que la altura del árbol sea de estructura logarítmica. Para esta actividad utilizar binary heap resulta más conveniente que un BST porque:

- Las operaciones de inserción (push) y eliminación (po) son O(log n).
- El max siempre está en la raíz.
- getTop() es O(1).

Analiza la complejidad computacional de las operaciones básicas de la estructura de datos empleada en tu implementación (push, pop, getTop, etc.) y cómo esto impacta en el desempeño de tu solución.



Utilizando estas propiedades, recuperar el top 10 de IPs con mayor acceso tiene un costo de aprox. O(10 log n). Haciendo una solución viable incluso cuando la bitácora tiene cantidades extensas de datos.

El uso de heap sort y binary heap permite que la aplicación funcione de manera adecuada en cada etapa:

- ***Ordenar bitacora*** → O (n log n)
- ***Contar accesos de IP*** → O (n)
- ***Inserción en binary heap*** → O (k log k) , k= num de diferentes IP
- ***Top 10 de IP*** → O (10 log k)

Garantizando que el programa sea eficiente en tiempo y recursos.

¿Cómo podrías determinar si una red está infectada o no?

Cuando una red presenta patrones anómalos en la bitácora, se considera infectada; algunos de los indicadores de que la red efectivamente se encuentra infectada son:

- IP con números de accesos excesivamente altos.
- Accesos repetitivos en tiempos cortos.
- Muchos intentos fallidos de autenticar la IP.
- Uso frecuente de “Illegal user”.

La aplicación que implementamos permite identificar las IP con mayor actividad, ayudando a detectar infecciones en la red; al enfocarnos en los números de accesos, un administrador de red puede revisar los casos sospechosos y tomar decisiones.

Como conclusión, esta actividad me ayudo a ver la importancia al elegir las estructuras de datos jerárquicas que se implementan en un programa, especialmente cuando se trabaja con muchos datos. Utilizar heap sort y binary heap garantiza que el programa sea eficiente. Usar un binary heap en vez de un BST ayuda a facilitar la recuperación rápida de los valores máximos. Este tipo de problemáticas se reflejan en el área de seguridad informática,

la cual me resulta muy interesante y poderla ver y aprender en cursos universitarios sobre aplicaciones en la vida cotidiana.

BIBLIOGRAFÍA

- GeeksforGeeks. (2025, December 22). Heap sort. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/heap-sort/
- GeeksforGeeks. (2026, January 19). Binary heap. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/binary-heap/
- GeeksforGeeks. (2025, July 23). Asymptotic analysis. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/asymptotic-notation-and-analysis-based-on-input-size-of-algorithms/
- priority_queue - C++ Reference. (n.d.). Cplusplus.com. https://cplusplus.com/reference/queue/priority_queue/