A01713396 - Santiago Martin del Campo

Reflexión Sobre el Uso de Estructuras de Datos Jerárquicas y Binary Heap

En esta actividad 3.3, el objetivo fue analizar una bitácora de accesos a un sistema para identificar las direcciones IP con mayor número de accesos. Para lograrlo, fue necesario procesar una gran cantidad de registros, ordenarlos correctamente y después aplicar una estructura de datos que permitiera obtener de forma eficiente la información más relevante.

En esta actividad no fue únicamente leer y ordenar los datos, sino que seamos capaces responder preguntas concretas como cuáles son las IPs con más accesos, algo que en un entorno real puede estar relacionado con tráfico sospechoso o posibles ataques a la red.

Importancia de las estructuras de datos jerárquicas

Durante el desarrollo de esta actividad aprendí que las estructuras de datos jerárquicas son especialmente útiles cuando se necesita priorizar información. En este caso, no era suficiente con saber cuántas veces aparecía cada IP, sino poder obtener rápidamente aquellas con mayor número de accesos.

Las estructuras jerárquicas permiten organizar los datos de forma que las consultas más importantes se resuelvan en el menor tiempo posible, lo cual es fundamental cuando se trabaja con grandes volúmenes de información como una bitácora de red.

Ventajas de un Binary Heap sobre un BST

Aunque un Binary Search Tree puede almacenar los datos de forma ordenada, su desempeño depende de que el árbol se mantenga balanceado. En situaciones donde los datos se insertan de forma desordenada o repetitiva, el BST puede perder eficiencia y volverse similar a una lista.

En cambio, el Binary Heap nos garantiza siempre una estructura balanceada y está diseñado específicamente para manejar prioridades. En esta actividad, el Binary Heap permitió almacenar pares de datos (IP, número de accesos) y recuperar de manera inmediata la IP con mayor número de accesos. Esto lo convierte en una mejor opción que un BST, ya que el enfoque de esta problemática está en obtener el valor máximo de forma rápida y constante.

Complejidad computacional y desempeño

Las operaciones del Binary Heap tienen una complejidad temporal que impacta positivamente en el desempeño del programa:

- **push:** O(log n), al insertar un nuevo elemento manteniendo la propiedad del heap.
- **pop:** O(log n), al remover el elemento con mayor prioridad.
- **getTop:** O(1), ya que el elemento más importante siempre se encuentra en la raíz.

Gracias a estas complejidades, fue posible obtener las diez IPs con mayor número de accesos de forma eficiente, incluso cuando la cantidad de registros era elevada. Esto demuestra que el Binary Heap es una estructura adecuada para este tipo de análisis.

¿Cómo podríamos determinar si una red está infectada?

Para determinar si una red está infectada, es necesario analizar patrones anormales en la bitácora. Por ejemplo, una IP con un número excesivamente alto de accesos en un periodo corto de tiempo puede indicar un ataque de fuerza bruta o un comportamiento automatizado. Al identificar las IPs con mayor número de accesos mediante el Binary Heap, es posible detectar rápidamente este tipo de anomalías y tomar decisiones preventivas.

Reflexión final

En conclusión, esta actividad mostró la importancia de seleccionar correctamente las estructuras de datos para analizar grandes volúmenes de información. El Binary Heap permitió identificar de forma eficiente las IPs con mayor número de accesos, lo cual resulta fundamental en la detección de patrones sospechosos dentro de una red. Este aprendizaje resalta el papel que tienen las estructuras de datos en la ciberseguridad para mejorar el análisis, la prevención y la toma de decisiones ante posibles amenazas.

Referencias Bibliográficas:

Bose, A., & Bose, A. (2025, 14 febrero). *Malware Detection: How to detect and remove malware ?* Seceon Inc. https://seceon.com/malware-detection-how-to-detect-and-remove-malware/

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/

GeeksforGeeks. (2025b, julio 23). *Why is Binary Heap Preferred over BST for Priority Queue?* GeeksforGeeks. https://www.geeksforgeeks.org/dsa/why-is-binary-heap-preferred-over-bst-for-priority-queue/

GeeksforGeeks. (2025c, 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/

www.naukri.com. (s. f.). *Code 360 by Coding Ninjas*. 2024 Naukri.com. https://www.naukri.com/code360/library/why-binary-heap-is-better-than-binary-search-tree-bst-for-priority-queues