Instituto Tecnológico y de Estudios Superiores de Monterrey

Act 3.4 - Actividad Integral de BST (Evidencia Competencia)

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

TC 1031.602

Carlos Adrián García Estrada -A01707503

21 de mayo de 2022

Reflexión

Durante este siglo el desarrollo tecnológico ha tenido un auge y esto ha llevado a que no solo nuestros teléfonos celulares y computadoras estén expuestos a ataques maliciosos, sino que ahora debido a la revolución del internet de las cosas muchos de nuestros objetos de uso común no solo están digitalizados sino también que utilizan internet para funcionar. Como se menciona en un artículo de Techtarget, estos objetos son los más expuestos pues sus tecnología al ser de funcionalidad básica no se le implementa medidas de seguridad fuertes (Hanna, K, s.f).

Uno de estos tipos de ataques son los llamados botnets, redes de software maliciosas que tienen el objetivo de infectar el mayor número de dispositivos posibles para obtener información personal y sensible como dirección, información de tarjetas de crédito, etc. Lamentablemente este problema parece ir a la alza pues coincide con la proliferación de dispositivos IoT, por ejemplo un artículo de Comparitech menciona que en Latinoamérica la cantidad de bots y C&C (centros de comando de botnets) es la más alta en todo el mundo y que la cantidad de ataques de botnets no solo incrementa bastante por año sino por cuartil, donde incrementaron un 23% en 2022. (Mccart, C, marzo,2022).

Los algoritmos de búsqueda y ordenamiento son herramientas de gran utilidad para enfrentar una crisis de botnets, pues son formas estandarizadas de ordenamiento y filtrado de información y en el caso de la situación problema nos ofrecen la estandarización de datos para una lectura y búsqueda de datos más rápido, permitiéndonos identificar con facilidad la ubicación de los ataques provenientes así como la causa de los mismos. Es por eso que es de máxima importancia no utilizar un algoritmo de búsqueda u ordenamiento, sino tomar en cuenta la complejidad computacional dependiendo del tipo de datos que estaremos manejando así como la cantidad.

Para esta evidencia tenemos un caso similar a los pasados, donde se nos otorga una bitácora con la fecha de entrada de ciertas computadoras a un servidor, sin embargo, en este caso nuestro enfoque no se encuentra en fechas sino en las ip 's que intentaron ingresar. De este modo podemos hacer un análisis cuantitativo por IP y determinar cuál de ellas es potencialmente más peligrosa; por otro, ahora se estará trabajando con *Binary Search Tree.*

Los *BST´s* similarmente a las doubly linked list, tienen dos apuntadores a otros datos, sin embargo en los BST estos apuntas hacia sus `hijos" y estos cumplen la regla de ser menores si están a la izquierda y mayores a la derecha. Esta condición nos permite sin hacerle ninguna modificación al BST un arreglo mucho más conveniente ya que los datos ya tienen incompleto ordenamiento} por otro lado, si nosotros decidimos utilizar el BST *Max Heap* nos permite ordenarlos a través de *heap sort* con una complejidad de O(n(log(n)).

Para esta actividad se nos pide que ordenemos la bitácora por ip´s, analizando podríamos concluir que el uso de listas ligadas sería más conveniente porque sus algoritmos de ordenamiento tienen complejidades menores a O(n(log(n)) pero recordemos que el acceso a

los datos es de O(n) mientras que para los árboles binarios es de O(log(n). La mayor ventaja radica en el ordenamiento por repetición, ya que para implementar un método como max heap sería necesario una complejidad de O(n^2) ya que sería necesario recorrer toda la lista n veces por cada dato diferente a contar. Por otro para el ordenamiento de mayor a menor dependiendo del número de accesos se facilita en un BST gracias a la propiedad de los *Max heap* que una vez contados, serían ordenados automáticamente con complejidad O(log(n)), mientras que para una lista ligada tendría que ordenar manualmente con complejidad mínima de O(n).

A través de estos procesos, podríamos concluir si una computadora se encuentra infectada o no, ya que podemos hacer un conteo rápido del número de accesos dependiendo la IP, si podemos visualizar que hay accesos fuera del promedio por una IP en específico, podemos concluir que la red se encuentra infectada, ya que la funcionalidad de los bots radica en infectar el mayor número de computadoras en el menor tiempo posible. Una verdadera ventaja requeriría del uso de una tabla hash, donde su complejidad es de O(1) y esto nos permitiría obviar del uso de cualquier ADT previa para el ordenamiento por repetición. Esta será la nueva ruta que se tomará en el curso.

En conclusión, los árboles binarios ofrecen una gran ventaja para el ordenamiento por valor en su modelo maxheap y el acceso a los datos que el de las listas ligadas en su modelo BST, lo cual nos permite acceder y visualizar los datos más eficientemente y en caso de un potencial ataque estrategizar mucho más tempranamente.

Referencias:

1. Mccart, Craig. (15 de marzo de 2022). *15 Shocking botnet statistics. Comparitech.* Recuperado de: https://www.comparitech.com/blog/information-security/botnet-statistics/
2. Geeks for Geeks (2022).Binary Search Tree data structure. Recuperado de: https://www.geeksforgeeks.org/binary-search-tree-data-structure/
3. Geeks for Geeks (2022). Time Complexity of all sorting algorithms. Recuperado de:https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/
4. Woltman, S. (agosto, 2020). Heapsort – Algorithm, Source Code, Time Complexity. *Happy Coders.* Recuperado de: https://www.happycoders.eu/algorithms/heapsort/#:~:text=Heapsort%20is%20an%20efficient%2C%20unstable,less%20commonly%20encountered%20in%20practice.
5. Geeks for Geeks (2022).Complexity of different operations in Binary Tree, Binary Search Tree and AVL. Recuperado de: https://www.geeksforgeeks.org/complexity-different-operations-binary-tree-binary-search-tree-avl-tree/