# Act 3.4 - Actividad Integral de BST (Evidencia Competencia)

Programación de estructuras de datos y algoritmos fundamentales
Grupo 12

Marisol Rodríguez Mejía A01640086

6 de noviembre de 2021

Marisol Rodríguez Mejía – A01640086

# Reflexión

Un árbol de búsqueda binaria es un árbol binario donde cada nodo contiene una clave y un valor asociado opcional. Permite una búsqueda, adición y eliminación de elementos particularmente rápida debido a que sus elementos tienen un orden específico:

- El subárbol izquierdo de un nodo en particular siempre contendrá nodos con claves menores que la clave de ese nodo.
- El subárbol derecho de un nodo en particular siempre contendrá nodos con claves mayores que la clave de ese nodo.
- El subárbol izquierdo y derecho de un nodo en particular también serán, a su vez, árboles de búsqueda binarios.

Para esta actividad, el BST que elegí usar fue un heap. Éste es una estructura de datos tipo BST en la que el árbol es un árbol binario completo. Para tener un heap correcto, éste debe de cumplir con las siguientes reglas:

1. Todo nodo padre del árbol contiene un elemento que tiene un valor de mayor prioridad que los valores de sus nodos hijos.
2. El árbol está completamente balanceado.
3. Los nodos hojas del nivel inferior están lo más a la izquierda posible.

Generalmente, los heaps pueden ser de dos tipos Min-Heap y Max-Heap. En esta actividad se usa un Max-Heap. Elegí el max debido a que su nodo raíz debe ser la mayor entre las claves presentes en todos sus elementos secundarios y, como queremos obtener las IPs con mayor número de ataques, me pareció adecuado utilizar esta estructura para que su prioridad fuera de mayor a menor.

Por otra parte, dado que un heap es un BST completo, se puede representar fácilmente como una lista en la cual el nodo padre se almacena en el índice i. Su hijo izquierdo se puede calcular con 2 * i + 1 y el hijo derecho con 2 * i + 2.

La importancia de utilizar un BST en una situación problema de esta naturaleza se debe a que nos permiten mantener un flujo de datos ordenado, y considerando que tenemos una base de datos relativamente grande que queremos manejar, utilizar un BST para lograrlo es algo que nos beneficia, de una manera en la cual podemos modificar la aplicación que le damos dependiendo del problema a resolver.
Asimismo, el heap me permitió tener una estructura en la cual se considera con una prioridad en específico, en este caso, el número de ataques a cada dirección IP. El Max-Heap hizo que se volviera algo poco complejo el proceso de obtener las IPs con mayor número de ataques, utilizar este tipo de estructura de datos hace que podamos tener nuestros datos ya ordenados y listos para analizar.

Los algoritmos utilizados para la actividad fueron principalmente:

- **Heap sort:** usé este algoritmo para ordenar la bitácora de menor a mayor. En éste repetidamente elegimos el elemento más grande y lo movemos al final del vector, convirtiendo el vector en un Max-Heap para acelerar el proceso. Tiene una complejidad temporal linearítmica en todos sus casos. Dicha complejidad se debe a que la altura de un árbol binario de tamaño n es $\log_2(n)$ como máximo, es decir, si el número de elementos se duplica, el árbol se vuelve solo un nivel más profundo. Para ordenar con heap sort, examinamos cada elemento del árbol y lo movemos hacia abajo hasta que sea más grande que sus hijos. Dado que la altura de nuestro árbol es O(logn) podríamos hacer hasta logn movimientos. Aplicando lo anterior a el número n de nodos, obtenemos la complejidad temporal y general de O(nlogn).
- **Push:** fue el método utilizado para ir agregando elementos al Max-Heap. Su complejidad puede ser calculada considerando un escenario en el cual insertamos un nuevo nodo en el heap, y la prioridad del padre del nodo es mayor que la del nodo recién insertado. En

tal caso, no necesitamos hacer nada y no se necesita ningún cambio en el Max-Heap ya que cumple con las reglas necesarias; por ello, la complejidad sería de O(1). Sin embargo, en otros casos no es así; en el peor caso, como esta estructura es un árbol completo y balanceado, el nuevo nodo insertado tiene que ser intercambiado en cada nivel desde el último hasta la raíz para mantener las propiedades de los heap. Al tener que intercambiar en cada nivel del árbol y sabiendo que tenemos una altura de O(logn), recorrer todo el árbol hace que nuestro algoritmo push cuente con una complejidad temporal de O(logn).

- **Pop:** este método se encarga de eliminar y retornar la raíz del Max-Heap, es decir, el elemento de mayor prioridad. Su complejidad es de O(logn) debido a que para obtener la raíz tarda O(1), pero, para eliminar dicha raíz, hay un proceso de reemplazo desde la raíz hasta la parte inferior del montón, ese recorrido del árbol toma O(logn) tiempo.

Considerando todo lo anterior, también me parece pertinente haber utilizado un BST gracias a las ventajas que nos brindan sus funciones de push y pop al tener una buena complejidad temporal. Debido a que en la actividad se deben de hacer varias inserciones, el heap se vuelve una estructura eficiente para lograrlo. De igual manera, para obtener las IPs con mayor número de ataques, el Max-Heap lo vuelve tan eficiente como realizar 5 pops de complejidad logarítmica.

El uso que tiene esta actividad integral es que nos permite ver si una red está infectada o no.
Los ataques cibernéticos son asaltos que utilizan una o más computadoras contra una o varias computadoras o redes. Un ciberataque puede deshabilitar computadoras de forma maliciosa, robar datos o usar una computadora violada como punto de lanzamiento para otros ataques. Los ciberdelincuentes utilizan una variedad de métodos para lanzar un ciberataque, que incluyen malware, phishing, ransomware, denegación de servicio, entre otros.
Un bot es una pequeña pieza de software que automatiza las solicitudes web con varios objetivos. Los bots se utilizan para realizar tareas sin intervención humana, que incluyen todo,

desde brindar soporte de servicio al cliente hasta probar los números de tarjetas de crédito robadas. Por ello, un bot puede usarse tanto de manera beneficiosa como dañina.

Los ataques de bots son ataques cibernéticos que utilizan solicitudes web automatizadas destinadas a manipular un sitio web, una aplicación o un dispositivo.

Un ataque de bot es el uso de solicitudes web automatizadas para manipular, defraudar o interrumpir un sitio web, aplicación, API o usuarios finales. Los ataques de bot comenzaron como simples operaciones de spam y se han ramificado en empresas criminales multinacionales complejas con sus propias economías e infraestructuras.

En esta actividad integral, la manera en que podemos detectar si una red ha sido infectada es considerando el número de ataques registrados para cada dirección, si es un número grande, podemos asumir que la red se encuentra bajo ataque.

**Referencias:**

Doubly Linked List | Set 1 (Introduction and Insertion) - GeeksforGeeks. (2014, March 12). Retrieved October 15, 2021, from GeeksforGeeks website: https://www.geeksforgeeks.org/doubly-linked-list/

Qué son las estructuras de datos y por qué son útiles. (2019, October 18). Retrieved October 15, 2021, from OpenWebinars.net website: https://openwebinars.net/blog/que-son-las-estructuras-de-datos-y-por-que-son-tan-utiles/

Listas enlazadas vs vectores: una gran decisión a nivel de programación. (2020). Retrieved October 15, 2021, from Delfino.cr website: https://delfino.cr/2020/10/listas-enlazadas-vs-vectores-una-gran-decision-a-nivel-de-programacion