Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Programación de estructura de datos y algoritmos fundamentales

TC1031, Grupo 601

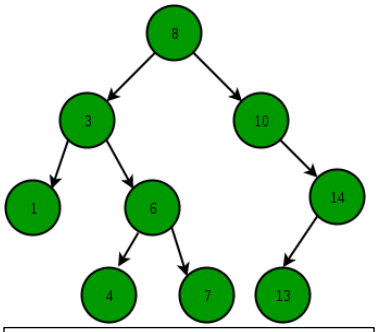
Nombre del profesor: Dr. Eduardo Arturo Rodríguez Tello

Actividad 3.4 – Reflexión personal

Samuel Acosta Ugarte | A00833547

22 de mayo 2022

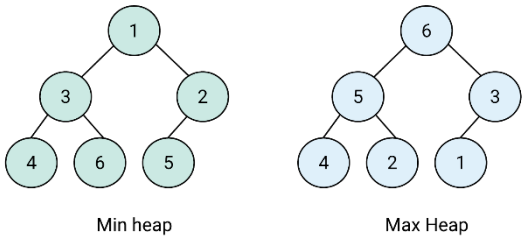
BST y Heaps



BST – GeeksforGeeks, 2022

La estructura de datos BST desde que lo vimos en clase, se me hizo bastante útil para los programas realizados últimamente al igual que los Heaps, que se pueden representar por arboles con dos hijos similarmente a la representación de un BST. La idea de un BST se basa en el algoritmo de búsqueda binaria, cada comparación de elementos permite eliminar la mitad del árbol, creando una complejidad O(logn), siguiendo la filosofía de divide and conquer (freeCodeCamp, 2019). Las aplicaciones de un BST varían, pero principalmente se usan para implementar algoritmos de búsqueda, mantener un orden en los datos e indexación (AfterAcademy, 2020). En este problema se tomo provecho de algunas de estas características para la solución, para mantener control de los datos y a su vez poder procesarlos cuando sea necesario.

Un heap es un árbol binario completo, que ordena loas datos dependiendo de dos tipos de heap: min-heap o max-heap. En un max heap, el valor de cada nodo debe ser mayor que sus hijos, y para un min heap, el valor de cada nodo debe ser menor que sus hijos (Anand, 2022). Esto nos permite conocer el valor de un valor máximo o un valor mínimo (dependiendo del tipo de heap), en una complejidad O(1), constante. En la solución de este problema creamos un max heap, basado en las ip con mas accesos. De esta manera podemos tener el control de las ips con mas accesos en una estructura de árbol binario y a su vez usando la función pop 5 veces, podemos conocer las 5 ips con más accesos. Podemos usar Heaps en cualquier tipo de problema donde se requiera procesar datos con prioridad, lo cual es muy común (Anand, 2022).



Min heap y Max heap, Applications of Heap – Data Structure, codestudio, 2022

¿Como saber si una red está infectada o no?

Yo pienso que para saber si una red infectada, podríamos representar cada nodo con un atributo de cantidad de accesos al nodo, que representa una computadora. Necesitaríamos conocer cuáles son los accesos normales de cada nodo en general, podríamos implementar una función que mande aviso si mas de 3 nodos (que representan computadoras), si hay una anormalidad de la cantidad de accesos a una computadora. Además, podríamos tener un control de los accesos totales, así podríamos corroborar si el número de conexiones a la red es anormal. Usar BST, puede ser útil para buscar cada nodo, pero también se podrían usar grafos para tener mas relaciones, puede ser que podemos conocer el origen del ataque con sus relaciones y actuar sobre ello.

Referencias bibliográficas

FreeCodeCamp. (2019). Binary Search Trees: BST Explained with Examples. 2022, de freeCodeCamp Sitio web: https://www.freecodecamp.org/news/binary-search-trees-bst-explained-with-examples/

Anand, G. (2022). Applications of Heap - Data Structure. 2022, de CodeStudio Sitio web: https://www.codingninjas.com/codestudio/library/applications-of-heap-data-structure#:~:text=heap%20data%20structure.-,Applications%20of%20the%20Heap%20Data%20Structure,time%20using%20a%20priority%20queue.

AdminAfterAcademy. (2020). Binary Search Tree: Introduction, Operations and Applications. 2022, de AfterAcademy Sitio web: https://afteracademy.com/blog/binary-search-tree-introduction-operations-and-applications#:~:text=BSTs%20are%20used%20for%20a,a%20sorted%20stream%20of%20data.