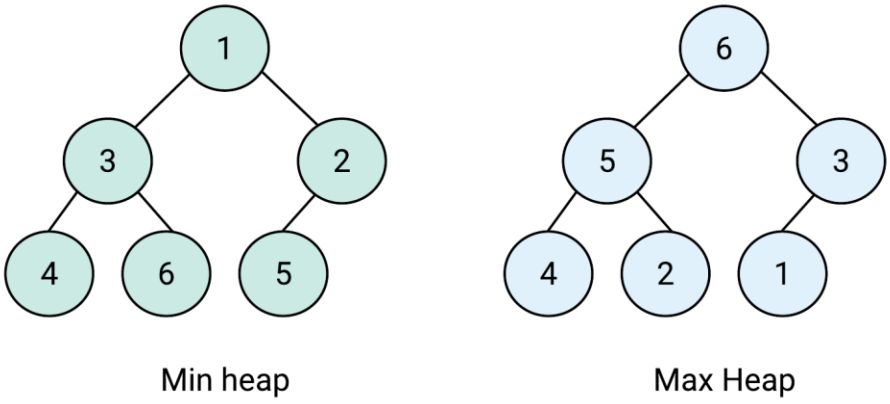
Samuel García Berenfeld A01642317

Reporte de Actividad 3.3

Parte 1: importancia del árbol HEAP en aplicaciones como esta.

En esta aplicación particular, nuestra meta era crear una estructura de datos capaz de almacenar los registros de nuestra bitácora, misma que contiene miles de datos de intentos de login a un servidor. En esta aplicación particular utilizamos una estructura del tipo MAX HEAP, gracias a sus cualidades particulares. Un árbol MAX HEAP consiste de un binary tree (lo cual significa que ningún nodo particular puede tener más de dos hijos) con una característica especial: ningún nodo puede tener un valor más grande que cualquiera de sus ancestros (Anand, 2023), como visto a continuación (en una comparación con la estructura MIN HEAP, que hace lo opuesto):



Para esta aplicación, esta estructura es particularmente útil, ya que estamos buscando IPs en la red con muchos intentos de acceso al servidor. Esta estructura garantiza que las IPs con más intentos de acceso siempre estarán en los primeros niveles del árbol, por lo que podemos acceso al nodo más grande de manera garantizada con nuestra consulta de Top().

Parte 2: uso de aplicación para detectar una red infectada.

Como mencionado en la sección anterior, nuestra estructura nos permite siempre saber la IP con más intentos de acceso, ya que organizamos los nodos respecto a la cantidad de intentos en nuestro MAX HEAP, de tal manera que nuestro primer registro siempre va a ser el IP con más ataques. En situaciones de ataque, conocer las cabezas (así como los nodos en el primer nivel) del árbol permitirían detectar con eficacia que IPs están generando el problema para proceder a prohibirlas. El caso más común de este tipo de ataques es aquellos llamados DDOS , que consisten en un gran número de ordenadores haciendo intentos constantes de acceso a algún servidor, con el fin de sobrecargar su poder de procesamiento para que no pueda operar correctamente (Toh, 2022). En estos casos, podríamos rápidamente detectar los IPs problema con el fin de bloquear su acceso al servidor, mitigando el ataque. De la misma forma, se puede hacer esto con cualquier IP subsecuente que se encuentre en la posición más alta de nuestro HEAP. Gracias a estas

características, podemos determinar que la estructura seleccionada funciona bien para nuestra aplicación.

Parte 3: reflexión general de lo aprendido.

Personalmente, me ha gustado mucho como cada entrega construye sobre la anterior, integrando nuevas áreas de conocimiento a la manipulación de la bitácora proporcionada de diferentes maneras. Hemos aprendido que los mismos datos pueden ser manipulados de dentro de diferentes estructuras, dependiendo de cómo los quiera tratar y manejar el programador. Con esta actividad particular todo hizo clic: me di cuenta que el dato que elegíamos para organizar nuestra información iba mano en mano con la estructura que queríamos usar. No es suficiente entender los beneficios de cada estructura, sino también que aspectos de nuestros objetos se acomodan mejor a existir dentro de cada una. Con esto, me siento preparado para explorar el final del curso, y llevar a esta bitácora aún más allá.

Referencias:

Toh, A. (2022, January 25). *Azure DDoS Protection—2021 Q3 and Q4 DDoS attack trends*. Microsoft Azure. Retrieved February 1, 2023, from https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/

Anand, G. (2023, January 15). *Code Studio*. Coding Ninjas. Retrieved February 1, 2023, from https://www.codingninjas.com/codestudio/library/applications-of-heap-data-structure