Luis Angel Zavala Robles
A01706481

Act 3.4 - Actividad Integral de BST- Reflexión.

El problema resuelto como parte de la presente actividad solicitaba la obtención de los 5 IPs con más accesos de nuestro archivo bitácora.txt utilizando una estructura del tipo Árbol Binario de Búsqueda (BST).

EXPLICACION DEL ALGORITMO.

La solución propuesta parte de la clase BST.h realizándole unas pequeñas variaciones para adaptarla a nuestras necesidades. Primeramente, a nuestra clase Nodo se le agrego un atributo “key” que alojara el numero de veces que la IP guardada en “value” se repite. Posteriormente a nuestro método “find” se la agrego que en caso de encontrar el valor que recibe como parámetro, sume 1 al “key” de dicho “value”.

La implementación de nuestro algoritmo parte de leer nuestro archivo bitácora.txt y por cada línea del archivo se obtiene el IP sobre el cual se realizo el acceso. Dicho IP se envía como parámetro al método “add” de nuestro árbol. Si el IP ya se encuentra dentro de nuestro árbol, no se agrega como un nuevo nodo, sino que a su “key” se le suma 1. Caso contrario, es decir si el IP aún no se encuentra en nuestro árbol, el IP se agrega como un nuevo nodo con un “key” por defecto de 1.

Finalmente, cuando se ha terminado de leer el archivo bitácora, se hace un recorrido ByLevel de nuestro árbol, con la particularidad de que cada vez que encuentre un nodo con un key superior a 1, agregue dicho nodo a un vector. De esta manera al finalizar el recorrido ByLevel de nuestro árbol tendremos en un vector todas aquellas IP que se repiten más de 1 vez. Por ultimo se realiza un sort de dicho vector teniendo como parámetro de ordenamiento el valor de key para así tener ordenados los IP por su número de accesos. Dicha funcionalidad esta especificada en el método mostAccessed que regresa un string con los 5 IPs mas repetidos con su respectiva cantidad de accesos.

JUSTIFICACION Y EFICIENCIA.

Debido a que por cada línea contenida en nuestro archivo “bitácora.txt” se debe realizar una búsqueda de la IP en nuestra estructura de datos, es conveniente utilizar un Árbol Binario de Búsqueda, ya que realizar una búsqueda de un elemento dentro de un árbol de este tipo tiene complejidad algorítmica O(log(n)), representando una ventaja con respecto a otras estructuras como listas o vectores, donde el realizar una búsqueda es de complejidad O(n). Por su parte, el método de agregar en un BST tiene complejidad O(log(n)) por lo que su implementación se vuelve eficiente. Dentro de nuestro algoritmo, el método llamado mostAccessed() presenta una complejidad de O(n) ya que dentro de él realiza un recorrido del árbol por niveles. Por último, nuestro algoritmo está en cierta forma controlado por un while que termina cuando se han leído todas las líneas del archivo bitácora, por lo tanto, la complejidad final de nuestro algoritmo es 0(n).

CONCLUSION.

Para un problema de esta naturaleza es conveniente utilizar una estructura que realice la búsqueda y la inserción de elementos de manera eficiente, siendo un BST una gran alternativa al realizar dichas operaciones con una complejidad de O(log(n)). De esta manera podemos analizar las redes infectadas que se encuentran registradas en el archivo bitácora.txt y determinar cuales corren mayor riesgo o podrían encontrarse vulnerables basándonos en las veces que se han intentado acceder en un periodo de tiempo.