Actividad 3.4

Roberto Eduardo Cardona Luis
A00833365

Para este trabajo nosotros utilizamos el árbol AVL, este árbol nosotros lo escogimos debido a que este tipo de árboles siempre están equilibrados a que nos referimos con esto, con esto nos referimos que la altura de la rama derecha del árbol es la misma altura de la rama izquierda del árbol, también utilizamos este modelo de árbol AVL debido a que la situación problema que nosotros debíamos resolver teníamos que buscar entre toda la lista que teníamos de las IP ordenadas, desde un punto de inicio hasta un punto final.

¿Cómo podrías determinar si una red está infectada o no?

En la situación problema se nos menciona sobre los botnet, esta palabra se forma con las palabras en ingles de robot y network, este mencionado botnet es un virus que se ingresa en las redes para así entrar a dispositivos, estos "robots" que ingresan en las redes son controladas remotamente por un atacante, es decir por otra persona, y estos pueden ser utilizados para realizar actividades maliciosas. Casi todos los dispositivos que se conecten a las redes wifi pueden caer en estos botnet, desde ordenadores, hasta dispositivos de IoT, al igual que televisores y termostatos.

Estos llamados botnet al ingresar a un dispositivo pueden leer y escribir los datos que estén guardados en el sistema, al igual que pueden recopilar todos los datos personales que haya en estos dispositivos, también estos pueden espiar a las personas dueñas de los dispositivos ya que pueden acceder al sistema y ver mediante la cámara del dispositivo o ver todo lo que las personas hacen en este dispositivo, al igual pueden hacer spam sobre ciertas cosas, instalar aplicaciones que generen más malware al dispositivo, y como se mencionó anteriormente pueden infectar otros dispositivos esto mediante el wifi.

Muchas veces el wifi o los dispositivos de la red presentan muchas acciones que con estas podemos deducir o determinar si realmente una red esta infectada, entre estas se encuentra el mal funcionamiento de la red y la lentitud de esta, al igual que en muchas ocasiones estas no responden a las ordenes dadas por el usuario, también algo muy habitual es la mala velocidad de internet o cortes en esta.

Complejidad de los algoritmos utilizados para esta actividad:

Función para destruir algún nodo del árbol

```
// template<class T>
void AVLTree::destroyTree(NodoTree *root) { //Funcion para destruir el arbol
    if (root != nullptr) { //Si el valor es diferente a nulo se efectua la accion
        destroyTree(root->getLeft());
        destroyTree(root->getRight());
        delete root;//Elimina el arbol
        root = nullptr; //Toma el valor nulo
    }
}
```

Función para buscar en el árbol

```
NodoTree* AVLTree::search(accessList value)//Funcion utilizada para buscar algun balor en el arbol y regresar en donde se encuentra este nodo
{       NodoTree *aux = this->root;

        cout << "Buscando " << to_string(value.length) << endl;//Inicia la busqueda del nodo

        while ( aux != NULL )//Creamos un ciclo para cuando el valor a buscar sea diferente a nulo
        {       if (value.length < aux-> getData().length )//Si el valor es menor que el auxiliar imprimira que esta en la izquierda y seguira avanzando
                {       cout << " left " << endl;
                        aux = aux->getLeft();
                }
                else if(value.length > aux->getData().length )//Si es valor mayor a auxiliar seguira a la derecha
                {       cout << " right " << endl;
                        aux = aux->getRight();
                }
                else//Si el valor no es mayor ni menor, esto significa que el nodo se encontro y regresara donde está
                {       cout << " Se encontro " << to_string(value.length) << endl;
                        return aux;
                }
        }

        cout << " Aún no existe " << to_string(value.length) << endl;//Si no se encuentra nada imprime al usurio que no existe y regresara un valor nulo
        return NULL;
}
```

Función para contar los hijos

```
// template<class T>
int AVLTree::countChildren(NodoTree *node) {//Creamos un ciclo para contar los hijos del arbol

    int ch = 0;//Creamos una variable para el contador

    if (node->left != nullptr)//Si hay hijos en la izquierda  aumentaremos el contador
    {   ch++;   }
    if (node->right != nullptr)//Si hay hijos en la derecha  aumentaremos el contador
    {   ch++;   }
    return ch;


}
```

Función para rotar a la derecha

```
//template<class T>
NodoTree* AVLTree::rotaRight(NodoTree *Y) {//Funcion creada para realizar rotaciones a la derecha

    NodoTree *X = Y->left;
    NodoTree *T2 = X->right;
//Tomamos el nodo que queremos modificar
    X->right = Y;
    Y->left = T2;
    X->up = NULL;
    Y->level = max(height(Y->left), height(Y->right)) + 1;
    X->level = max(height(X->left), height(X->right)) + 1;

    return X; //Regresamos el valor del nodo ya modificado

}
```

Función para rotar a la izquierda

```
//template<class T>
NodoTree* AVLTree::rotaLeft(NodoTree *X) {//Funcion creada para realizar rotaciona a la izquierda

    NodoTree *Y = X->left;
    NodoTree *T2 = Y->right;
    //Tomamos el nodo ingresado por el usuario el cual moveremos a la izquiera
    Y->right = X;
    X->left = T2;
    Y->up = NULL;
    X->level = max(height(X->left), height(X->right)) + 1;
    Y->level = max(height(Y->left), height(Y->right)) + 1;

    return Y;//Regresamos el valor del nodo en la ubicacion deseada
}
```