**Árboles Binarios**

    Un árbol es una estructura de datos jerárquica, es decir la forma en la que se conectan los elementos, permite identificar niveles de importancia. Los árboles se componen de nodos; cada nodo contiene un dato y una o más referencias a otros nodos. En un árbol binario, cada nodo tiene a lo sumo dos nodos. Existen árboles **n-arios**, en los que los nodos pueden tener más de dos nodos.

    Estructura básica de un árbol binario:

    Un árbol binario de búsqueda (bts) cumple las condiciones:

* El hijo derecho es mayor que la raíz
* El hijo izquierdo es menor que la raíz
* Estas condiciones se cumplen en cualquier parte del árbol

    Un bts provee búsquedas rápidas y eliminaciones /  inserciones eficientes

¿Cómo se define un nodo?

    class nodo {

    int element;

    node right;

    node left;

    }

¿Cómo se define un árbol?

    class bts{

    nodo root = null;

    boolean isEmpty( ){

    return root = = null;

    }}

¿Cómo se busca en el árbol?

    boolean contains (int e){

    return this.contains (e, root)

    }

    private boolean contains (int element, nodo current){

    if (current = = null){

return false;

}else if (element < current.element){

return contains ( e, current.left);

}else if (element > current.element){

return contains(e , current.right);

else{

return true;

}

¿Cómo encontrar el mayor/menor?

    public nodo findmin ( ){

return this.findmin(this.root);

}

private nodo findmin(nodo current){

if ( current = = null){

return null;

}else if (current.left = = null){

return current;

}else{

return findmin (current.left);

}

}

¿Cómo se inserta en un árbol?

    public void insert ( int e ){

root = this.insert ( int e, nodo current)

}

private nodo insert ( int e, nodo current){

if (current = = null){

return new nodo(e);

}else if ( e < current.element ){

current.left = insert( e, current.left);

}else {

current.right = insert (e, current.right);

}

return current;

}