FindMax

```
// Algorithm FindMax
// Max. element can be found at rightmost node of the tree.
FindMax(root):
    while root.right != null do
        root = root.right
    end while
    return root
end
```

↳ Running time of FindMax is T(n).

Hence, we visit a path from root to a leaf this path requires n (height of the tree) comparisons. in worst case.

$$T(n) \in O(n)$$

Insert a New Element

```
//Algorithm InsertBST
//The given item is inserted to the appropriate position by adding it from
//root and placing it down in the tree.

InsertBST(Tree, item):
npos        = Tree.root                    // npos is the position that item will be inserted  (npos → newpos)
npar        = null                         // npar is parent node of the item
while npos != null do
    npar    = npos
    if item.value < npos.value do
        npos    = npos.left
    else do
        npos    = npos.right
end while                                  // After the first line, we find the null(leaf node) and parent node of the inserted item.
item.parent = npar
if npar == null do                         //
    Tree.root = item                       // If the parent that we have found is null then, it means that we added the item to empty tree.
else if item.value < npar.value do
    npar.left = item                       // If the tree is not empty we place item as the right or left child of the its paren(npar) according to its value.
else do
    npar.right = item
```

→ Running time of InsertBST is T(n)

→ Assume that assignment and comparison operations take O(1) time.
Basic operation of the InsertBST is comparison in the while loop.

→ Hence, we go down through the tree in one direction (left or right) for each iteration, we have number of n (height of the tree) comparisons.

Thus, $T(n) \in O(n)$

Delete a Leaf Node

- For deleting a leaf node, we just need to traverse to the proper position and simply remove the node from the tree.
- Operations such as replacing the deleted node with its child are not required for this case of deletion.
- We need to make n (height of the tree) comparisons to go from the root to a leaf node.

↳ If the deletion operation and comparisons take $O(1)$ time, the total running time of deleting a leaf node $T(n) \in O(n)$.

Printing BST using the inorder traversal

- When we perform a inorder traversal we need to visit all the nodes in the tree. So, we need to find maximum number of nodes.
- The maximum number of nodes in a binary tree of height n is $2^{n+1}-1$. This tree must be a perfect binary tree. Thus, we need to visit $2^{n+1}-1$ nodes to print BST using the inorder traversal.

↳ If printing operation takes $O(1)$ time, total running time of inorder traversal for this tree is $O(2^n)$.

$2^{n+1}-1 = 2^n \cdot 2 - 1$

↓ eliminate the constants.

$$T(n) \in O(2^n)$$