Delete and object from a BST.

Node remove Item( Node root, Object toRemove) {

If(root == null) //base case 1

Else if(toRemove == root.data) // base case 2

Else if(toRemove < root.data)

Else //toRemove > root.data)

Node removeFromRoot(root)

CASE 1:

STEP 1

STEP 2

STEP 3

CASE 2:

CASE 3:

CASE 4:

Node removeLargest(Node root)

{

if(root.right != null)

Root.right = removeLargest(root.right)

Else // root.right == null // base case

Root = root.left

Return root;

}

For N Levels, this is the time complexity of the following operations

Insert: O(n)

Remove: O(n)

Search: O(n)

Cost of creating a BST by means of a series of insertions

* If elements inserted in order of ascending value it takes O(n^2) this is because for each value it takes O(n) complexity to perform the operation on the array, which is size N. Thus you multiply n\*n (one for operation time and one for the size of the array/times operation is iterated) to get a Big O complexity of O(n^2)
* If the elements inserted in Random we are good enough to create a balanced BST.
  + Insert cost of one elements (log n)
  + Total of n elements: O(logn)