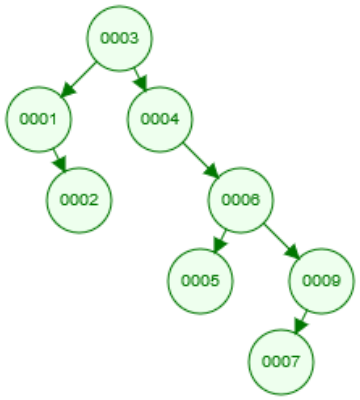
Homework #3 CECS 302-01

November 2, 2018

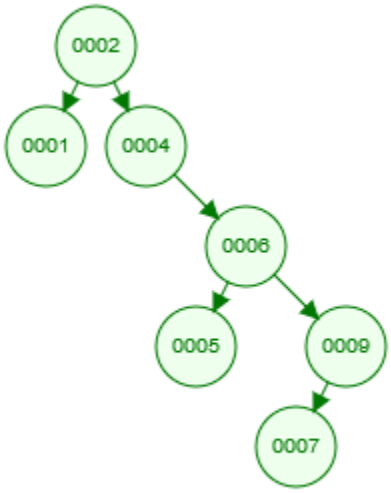
Nicholas Gittings

For problem one, the following picture is inserting numbers into the binary search tree.



Binary Search Tree with Numbers Given

Deleting the root would make all the other nodes not visible, unless you set the lowest number that's greater than the left child to root as shown in the next image.



For problem two, create a structure called node which contains an integer for data and two node pointers which will point to left and right nodes down the tree. All of the functions will use a queue which means they have O(n) which is linear time complexity. The functions to print all share the same method to search through the list, but returnNodes() has a counter which counts nodes that are added, returnLeaves() has a counter which increments by 0.5 every time a node is added and uses modulus to round up at the end which creates a leaf, and lastly returnFullNodes() counts the node like returnNode() except checks to see if the node contains data. The insert function checks to see if the data of the new node is greater than or less than the next nodes and sorts them on left side for less than, and right side for greater than.

For problem three, to create findKth(k) without sacrificing time bounds of other functions, use an integer in node to store information on the size of the node currently in the tree. Then use in order traversal for checking the node with the property that its traversal order is equal to k

For problem four, create a class called BSTiterator, which contains a constructor which stores the root and all the pointers through the root in a stack for in order traversal. Then, create two Boolean functions called begin() and end(), begin is true if stack is not empty, end is true if the stack is empty. Next, the ++ operator will pop the stack, and then lastly, the pointer operator will check the top of the node on top of the stack's data.