Problem 57

Ryan Burmeister

March 4, 2016

The expected worst case cost of insertion or search for a randomized binary search tree is $O(logn)$ as the height is expected to be somewhere on the order of the lg of the number of inputs. Deletion could simply occur by searching the tree until the node is found. Once the node is found, all the nodes in the subtree can be reinserted into a new subtree based on the insert algorithm we've already discussed. In this scenario, each value has an equivalent chance to be the root node of the subtree from which the value was deleted.

```
1: Start at root node
2: while node ! = null and node.value ! = searchValue do
3:     if searchValue > node.value then
4:         node = node.rightChild
5:     else if searchValue < node.value then
6:         node = node.leftChild
7:     else if searchVale == node.value then
8:         break
9:     end if
10: end while
11: if node ! = null then
12:     Delete node
13:     for node in oldSubtree do
14:         insert node into new subtree
15:     end for
16: end if
```