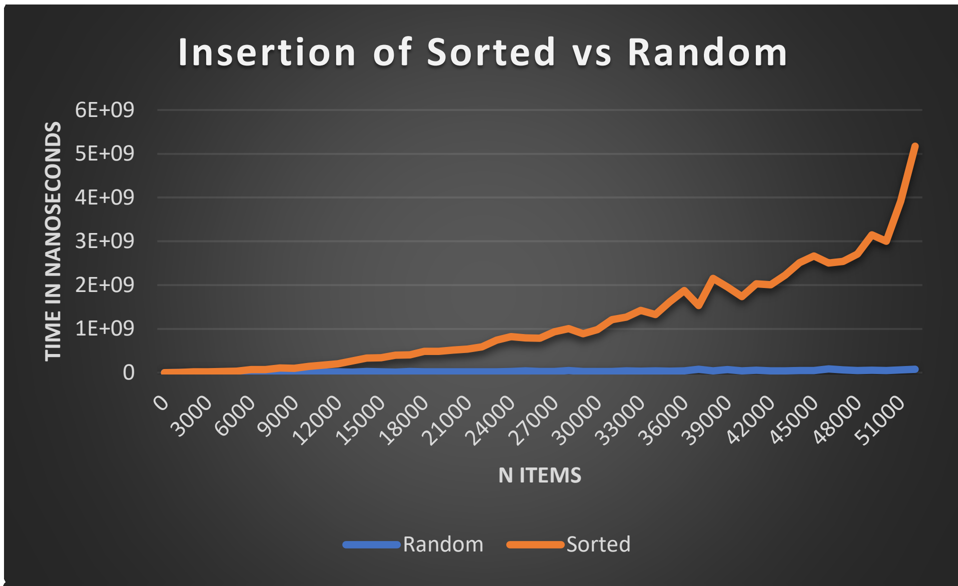
Harrison Smith
Ben Malohi

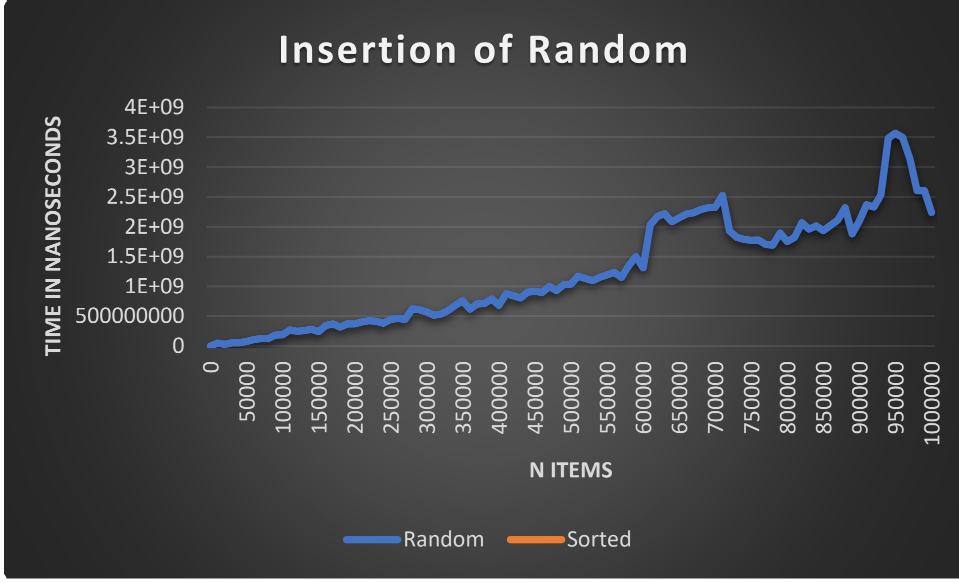
Analysis Doc: Assignment 7 – Spellchecker/BST

October 31, 2019
Prof. Swaroop Joshi

1. Experiment: Sorted Insert vs Randomly Insert of Integers converted to Strings via .toString() method call. We randomly generate the Integers and hold them within an ArrayList of type Integer, and then loop through the ArrayList to add them to the BST. While we add from the ArrayList to the BST, we take the time before that loop, then the time after that loop; compute the total time it took to run, and then continue this process for each N size of list, each time it complete it is for N size. However for the entire list inserting at each location it will take N^2 times because it has to touch N * (N-1) resulting in about N^2.
We found that when we add from a Sorted ArrayList of Integers, once we run into 53,000 items, the Stack would overflow, that is why our first graph is only to 52,000 items. However, when we added randomly we were able to hold more data, because the root data was not stuck adding all of the data to the right of the initial node; Random insert was more balanced, that it would be able to add nodes with data on both sides of the tree.

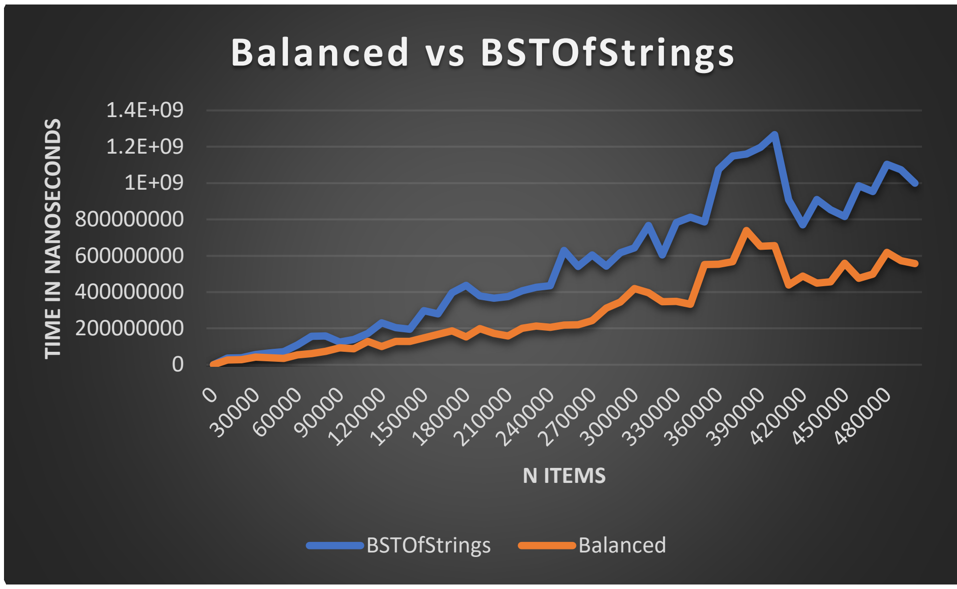
Sorted Insert runtime is N, this is because all items will be sorted directly to the right.
Random Insert runtime is logN, this is because it will not touch each item before it inserts the data to the BST, preforming more efficiently than the Sorted Insert.





2. We repeat the way we are approaching the first experiment, however, instead of inserting a sorted and random, we are just inserting a list randomly. Comparing the run time of how long it takes to insert in the provided BalancedBST and our implementation of BST, where it is not balanced.

Balanced – updates the root, resulting in a more balanced and evenly spread out BST. This will allow for the insertion or finding of the item to be quicker as it will have the potential to look at less items, the ultimate runtime is logN for this.
Our Implementation – does not update root after initially picking root. The runtime will still be logN however, as time progresses it will begin to approach a worse runtime as it will have the possibility of being unbalanced, since the initial root is a random choice, and it may not be the best root for the entire time.

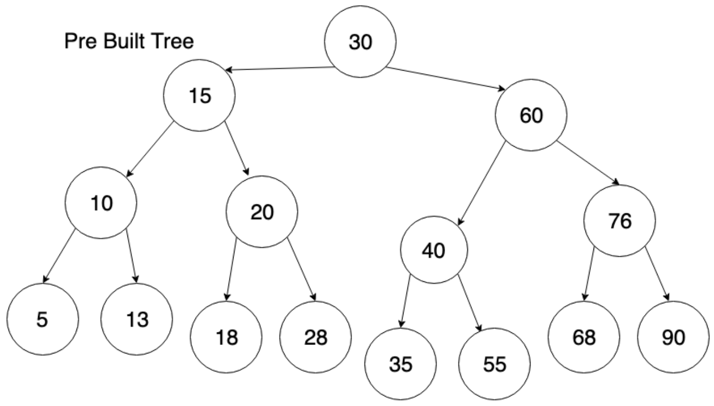


Harrison Smith
Ben Malohi

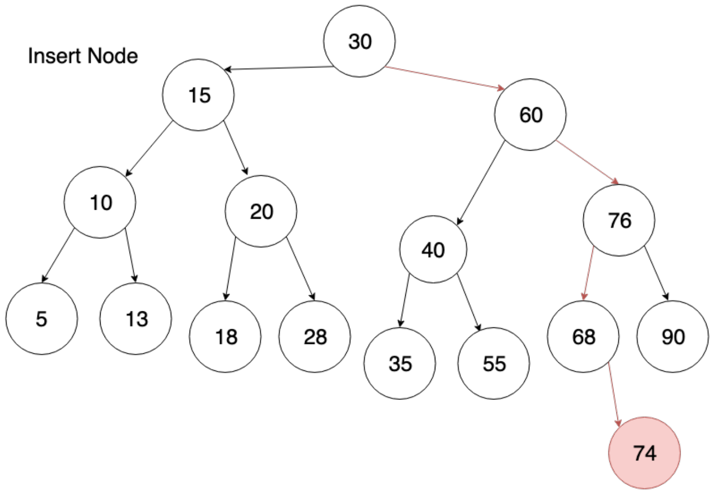
Analysis Doc: Assignment 7 – Spellchecker/BST

October 31, 2019
Prof. Swaroop Joshi

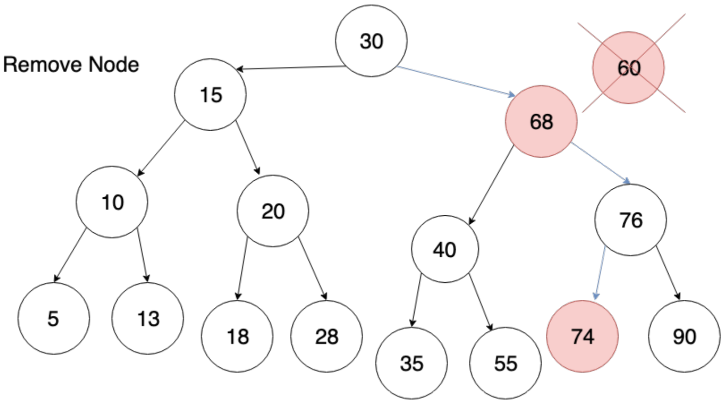
Pre Built Tree



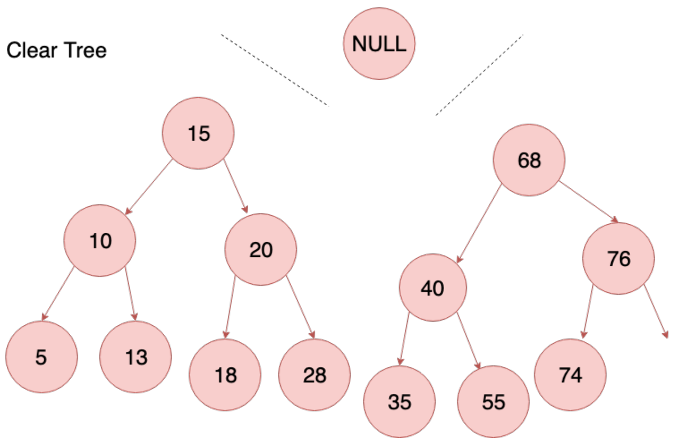
Insert Node



Remove Node



Clear Tree



NULL
15
68
10
20
40
76
5
13
18
28
35
55
74