Ryan Hulke

Report

Time Complexities:

(for all of these, n represents the number of nodes in the tree)

Insert: O(log(n)) because you only compare ID at each level of the tree to find where it fits, which takes log(n) time. Then you rebalance your way back up the tree, also taking up to log(n), but these added together just gives you 2log(n) and the constant can be dropped.

remove: O(log(n)) It first calls the search-with-ID function in order to make sure the node exists, which takes log(n) (see below), then you have to delete which is O(1), and then trace your way back up the tree recalculating the heights, which also takes log(n). So, reducing (O(2log(n) + 1)) gives O(log(n)

search with ID: O(log(n)) because you only compare at each level of the balanced tree until you find it

search with Name: O(n) because it searches in a preorder traversal, and worst case it could be the farthest right node, thus having to hit every node before it.

printPreorder, printPostorder, printInorder: O(n). They have to visit every node in the tree in order to print them.

removeInorder: O(n). Worst case, if the last index is called, it would have to traverse through every node in order, which is an O(n) operation. It also rebalances which is O(log(n))

printLevelCount: O(n), it goes through level by level hitting each node, queueing up each node's children, and tracking when it goes down a level. Therefore since it hits every node it is O(n)

I learned a ton about practical implementation of an AVL tree. I learned things like how to implement several different kinds of traversals, and the use cases for each. I thought it was really cool how easy it is to do a level order traversal with the queue, as well as the other traversals with very simple recursion. The most interesting things I learned were the remove and insert methods. These were the most challenging tasks for me, as I struggled to conceptualize what was happening and trace the recursion stack of which nodes ended up pointing where when I tried to do rotations. If I could go back, I would have spent more time beforehand understanding the rotations and how and why they work. I had a weak grasp of the concept going into the project, and I ended up paying for it by spending a lot of time debugging those functions.