Eric Powers

2/18/2023

AVL Report

Insert and Remove: The insert and remove commands are going to have a time complexity of O(log(n)). Since the AVL tree automatically balances itself through O(1) rotations after insertion and deletion, it maintains a "bushy" structure. This prevents it from having a worst time complexity of O(n) like you would see in a BST insertion or deletion because BST's can have an almost linear structure, like a linked list, in the worst case.

Search ID: My search ID function is going to have a time complexity of O(log(n)) because of the "bushy" nature of an AVL tree. Since each node is unique, there is no need to traverse over the entire tree which is why it has this time complexity.

Search Name: The search name process is a little bit different than the search ID process and has a time complexity of O(n). This is because the function uses a preorder traversal to visit all nodes in the tree since more than one node can have the same name.

Print Inorder, Preorder, and Postorder: All three of these traversals have a time complexity of O(n) just because you have to traverse over every single node in the tree.

Print Level Count: The print level count operation has a time complexity of O(n). This is because it requires me to find the height of each subtree that exists in the tree to find the largest one so I will be traveling to every node once.

Remove Inorder: This operation has a time complexity of O(n) because my implementation uses an inorder traversal to push each node into a vector. It then accesses the nth node of this vector and removes it. The inorder traversal is the more demanding operation with a time

complexity of O(n) compared to the deletion's time complexity of O(log(n)) which is why this process has an overall time complexity of O(n).

What I learned: I learned from this assignment that creating an AVL tree is no joke and takes a lot of time and dedication to not only grasp the concepts, but then to implement them in your own code. Also through this project I learned why we might use an AVL tree over a BST and the advantages that come with it. If I had to start this process over I would start just a little bit earlier than I did to learn the rotations earlier. This was the hardest thing for me to grasp conceptually and took me a lot of time to implement.