Harper Fuchs

AVL Tree Documentation

Command: insert (NAME) (ID)

Function name: string insert(string name, string id)

Time Complexity: O(log(n)) where n is the number of nodes

Notes: The time complexity is O(log(n)) because the function insertNAMEID(string name, string id) which is the private function called by insert is O(log(n)) which determines where to insert the node based on the data value (like a normal BST tree) and this reduces the amount of nodes that need to be traversed. All of the functions encased in the insertNAMEID function are O(1), except one other O(log(n)) and they are called outside of any loop, and therefore can be dropped in the consideration of the time complexity of this function. The other functions called in insert are O(1) and O(m+l) where m is the chars in the name and l is the num chars in id. I am not considering these in my analysis because they are called in my main outside of insert, but I had to put them into insert when I had to get rid of main to run my test cases. My analysis is based on the time complexity when my main is included. I will comment them out of the insert function when I comment out the test cases to submit my files.

Command: remove (ID)

Function name: string removeID(string id)

Time Complexity: O(n) where n is the number of nodes

Notes: The time complexity is O(n) because it calls two functions with O(n) time complexities (both in relation to nodes still) so that becomes O(2n) which drops the 2. The function deleteCases(Node* node) has a time complexity of O(n) where it is the number of nodes because that was the highest time complexity of the two non-O(1) functions it calls. O(n) occurs because it may have to access every node.

Command: search (ID)

Function name: string searchID(string id)

Time Complexity: O(n) where n is the number of nodes

Notes: This function is O(n) because in the worst case it may have to touch every node.

Command: search (NAME)

Function name: string searchNAME(string name)

Time Complexity: O(n) where n is the number of nodes

Notes: This function is O(n) because in the worst case it may have to touch every node.

Command: printInorder

Function name: void printIn()

Time Complexity: O(n) where in is the number of nodes

Notes: This function is O(n) because every node must be accessed to print.

Command: printPreorder

Function name: void printPre()

Time Complexity: O(n) where in is the number of nodes

Notes: This function is O(n) because every node must be accessed to print.

Command: printPostorder

Function name: void printPost()

Time Complexity: O(n) where in is the number of nodes

Notes: This function is O(n) because every node must be accessed to print.

Command: printLevelCount

Function name: int printLevelCount()

Time Complexity: O(1)

Notes: Only one node needs to be accessed to determine the number of levels so this is a constant time complexity.

Command: removeInorder (N)

Function name: string removeInorderN(int num)

Time Complexity: O(n) where n is the number of nodes.

Notes: This function is O(n) because the two functions encased in it are both O(n) both also with n being the number of nodes. These are O(n) because in the worst case they may need to access every node. O(2n) is then simplified to just O(n).

What I Learned

I learned a lot from this project. One of the most notable things is recursion. I already knew what recursion was, and have used it before, but this project gave me a lot of practice looking at it and I feel that I truly understand the concept now. I have learned about trees and how to rotate them, which is self explanatory, but the tree itself has strengthened my understanding of pointers and has been a great review. I have also had to revisit time complexity a lot, and this was good practice for that. If I have made any mistakes in my analysis, that will just be another opportunity to learn as I look through the feedback.

What I would do Differently

I would start implementing test cases a lot earlier. This part of the project seemed very overwhelming to me at the beginning and I put it off. I had never learned how to make a test case so I was stressed that it would take up too much of my time and cut into the time I had to fix my code since that was worth more points. After learning how to set up test cases, I realize that it is not that difficult or overwhelming and could have been a great tool to have as I worked on the project from the beginning.