Paul Kenaga

Professor Ostrowski

CS 300

26 Nov. 2021

Binary Search Tree Code Reflection

The code in this assignment demonstrates how to implement a binary search tree to hold data from a CSV file. Unlike other data structures, the time complexity for insert, search, and remove functions for a binary search tree is O(logN). While testing the code I noticed this structure outperformed the others from previous assignments. When a bid is added to the binary search tree, the code first checks to see if there is a root node. If there is not a root node, the bid data is assigned to the root. Otherwise, the code will check if the bid is greater than or less than the root and then traverse either the left or right subtree to find an empty node to place the bid. The use of left and right children allows the binary search tree to access nodes more easily. If the node you are looking for is less than the current node, move left. If the node is larger, move right. This concept is applied for all three basic functions; insert, search, remove.

While developing this code, I needed to redefine the node structure so that each node could have two children, a left and right one. Once I completed that part, I began to work on the Fix Me areas. I noticed that the private remove function was not included in the starter code so I also had to develop the logic for that. Using zyBooks and Stackoverflow as references helped me figure this part out. The most difficult logic to work out was removing a node with two children. For this case, we look for the leftmost node on the right subtree to replace the removed node. Then we need to remove the duplicate node. Another issue I ran into was with my Visual Studio IDE. SDL checks prevented me from running my code initially, just as it had during the hash table assignment. I needed to turn them off again and my code was able to run.