Dylan Dunagan

CS-300 DSA: Analysis and Design

8 June 2025

Module 5 Assignment Binary Search Tree Code Reflection

The purpose of this program is to allow for managing individual bid information uploaded from a CSV file into a binary search tree. This code allows the user to search for specific bids, insert bids, remove bids, and display all bids. For the program to work correctly, you must know how a binary search tree functions regarding which route it will take down the tree and how to manipulate it into taking the route you want it to. I struggled a lot with this assignment and how to apply what ZyBooks taught. To complete the assignment, I kept the ZyBooks pages open and utilized them significantly to try and get the correct code in place. It was hard to figure out which objects I needed to be using for specific functions to work, since ZyBooks of course does not use the same names for similar functions and objects. After enough trial and error, I feel that I managed to put together the correct code.

The pseudocode for the program is:

//Constructor  
BinarySearchTree()  
 Initialize root to null

//Destructor  
BinarySearchTree()  
 If node is equal to null  
 return  
 Else  
 delete nodes down the left of tree  
 delete nodes down the right of tree  
 delete root node

Insert()  
 if root is equal to null  
 root equal new node  
 else  
 add node

Remove()  
 call removeNode on root bidId

Search()  
 initialize currentNode to root  
 loop while currentNode does not equal null  
 if currentNode equals searched bidId  
 return bid  
 else if  
 searched bidId is less than current node  
 move to left  
 else  
 move to right

addNode()  
 if bidId is less than node’s  
 if left node equals null  
 left node equals new node  
 else  
 call addNode(left node, bid)  
 else  
 if right node equals null  
 right node equals new node  
 else  
 call addNode(right node, bid)

inOrder()  
 if node does not equal null  
 call inOrder(left node)  
 display bid  
 call inOrder(right node)

postOrder()  
 if node does not equal null  
 call postOrder(left node)  
 call postOrder(right node)  
 display bid

preOrder()  
 if node does not equal null  
 display bid  
 call preOrder(left node)  
 call preOrder(right node)

removeNode()  
 if node equals null  
 return null  
 if bidId is less than node’s  
 remove left node  
 else if bidId is greater than node’s  
 remove right node  
 else  
 if left node and right node equal null  
 delete node  
 return  
 else if left node does not equal null and right node equals null  
 temp equal left node  
 delete node  
 return temp  
 else if left node equals null and right node does not equal null  
 temp equal right node  
 delete node  
 return temp  
 else  
 temp equals right node  
 while left node does not equal null  
 temp equals left node  
 node’s bid equals temp’s bid  
 remove right node

main()  
 while choice is not 9  
 output menu  
 get user input  
 case 1  
 start timer  
 load CSV path  
 output time  
 case 2  
 output bids  
 case 3  
 start timer  
 search for bid  
 if bid is not empty  
 output bid  
 else  
 output error message  
 stop timer  
 output time  
 case 4  
 remove bid