Brandon Parkerson

CS-300

Professor Chowdhury

29 Nov 2024

5-2 Code Reflection and Pseudo-Code

The program implements a binary search tree to manage bid data. It uses in-order, pre-order, and post-order traversal to display data, search functionality for efficient retrieval, and a deletion mechanism for bid removal. The recursive logic for removing a node had me struggling a little bit. However, after careful handling to ensure proper tree restructuring, the function worked perfectly. I leveraged helper functions for recursion to keep the APIs simple and I also double checked traversal when completing the node removing function.

Pseudo-Code:

Insert(bid):

If root is null, make a new node with bid and assign to root.

Else, call addNode

addNode(node, bid):

If bid.bidId < node.bid.bidId:

If node.left is null:

Make new node and assign to left.

Else: recurse addNode(node.left, bid).

Else:

If right node is null:

Make new node and assign to right.

Else: recurse addNode(node.right, bid).

search(bidId):

Start at root

While current node is not null:

If current.bid.bidId == bidId:

Return current.bid.

Else if bidId < current.bid.bidId:

Traverse left.

Else: traverse right.

Return empty bid.

removeNode(node, bidId):

If node is null, return null.

If bidId < node.bid.bidId:

Recurse removeNode(node.left, bidId).

Else if bidId > node.bid.bidId:

Recurse remove node(node.right, bidId)

Else:

If node does not have children, then delete it.

If node has one child, replace node with child.

If node has 2 children, find min in right subtree,

Replace node.bid with minimum bid, and delete the min node.