Seema Chavesta

Oct 6, 2024.

**Code Reflection:**

The purpose of this code is to implement a Binary Search Tree that efficiently handles bids by supporting operations like insertion, search, removal, and traversal. I used recursion for tree traversal (in-order, pre-order, and post-order) and node management, which made it easier to manage complex tree operations. One of the more challenging parts was implementing the removeNode function, especially handling cases where a node had one or two children while maintaining the tree's structure. Additionally, the destructor required careful recursive deletion of all nodes to avoid memory leaks. To overcome these challenges, I relied on modular methods and recursive design to keep the code organized and easy to follow.

**Binary Search Tree Pseudocode:**

1. **Destructor**

Function Destructor():

Call removeNode(root) to delete all nodes

End Function

1. **Recursively Remove Nodes (removeNode)**

Function removeNode(Node, bidId):

If Node is null:

Return null

If bidId is less than Node's bidId:

Node.left = removeNode(Node.left, bidId)

Else if bidId is greater than Node's bidId:

Node.right = removeNode(Node.right, bidId)

Else:

If Node.left is null:

Return Node.right

Else if Node.right is null:

Return Node.left

Else:

successor = FindMinimum(Node.right)

Node.bidId = successor.bidId

Node.right = removeNode(Node.right, successor.bidId)

Return Node

End Function

1. **Insert a Bid**

Function Insert(Node, Bid):

If Node is null:

Create new node with Bid data

Return new node

If Bid.id is less than Node.bidId:

Node.left = Insert(Node.left, Bid)

Else:

Node.right = Insert(Node.right, Bid)

Return Node

End Function

1. **Search for a Bid**

Function Search(Node, bidId):

If Node is null:

Return null (Bid not found)

If bidId matches Node.bidId:

Return Node

If bidId is less than Node.bidId:

Return Search(Node.left, bidId)

Else:

Return Search(Node.right, bidId)

End Function

1. **In-Order Traversal**

Function InOrder(Node):

If Node is not null:

InOrder(Node.left)

Print Node.data (Bid information)

InOrder(Node.right)

End Function

1. **Pre-Order Traversal**

Function PreOrder(Node):

If Node is not null:

Print Node.data (Bid information)

PreOrder(Node.left)

PreOrder(Node.right)

End Function

1. **Post-Order Traversal**

Function PostOrder(Node):

If Node is not null:

PostOrder(Node.left)

PostOrder(Node.right)

Print Node.data (Bid information)

End Function

1. **Find Minimum Value in Right Subtree**

Function FindMinimum(Node):

While Node.left is not null:

Node = Node.left

Return Node

End Function