# CS 300 Pseudocode Document

Gentian Hoxha

[Gentian.hoxha@snhu.edu](mailto:Gentian.hoxha@snhu.edu)

4/16/2024

**Tree Pseudocode**

// Define a structure for holding bid information

Structure Bid

Define bidId as String

Define title as String

Define fund as String

Define amount as Double

End Structure

// Define a node in the binary search tree

Structure Node

Define bid as Bid

Define left as Node

Define right as Node

End Structure

// Define a Binary Search Tree class

Class BinarySearchTree

Private:

Define root as Node

Public:

Constructor BinarySearchTree()

root = null

End Constructor

Destructor ~BinarySearchTree()

// Recursive delete all nodes

End Destructor

Method InOrder()

Call inOrder(root)

End Method

Method Insert(bid as Bid)

If root is null

root = new Node(bid)

Else

Call addNode(root, bid)

End If

End Method

Method Remove(bidId as String)

root = Call removeNode(root, bidId)

End Method

Method Search(bidId as String) as Bid

Define current as Node = root

While current is not null

If current.bid.bidId == bidId

Return current.bid

ElseIf bidId < current.bid.bidId

current = current.left

Else

current = current.right

End If

End While

Return new Bid

End Method

Private:

Method addNode(node as Node, bid as Bid)

If node.bid.bidId > bid.bidId

If node.left is null

node.left = new Node(bid)

Else

Call addNode(node.left, bid)

End If

Else

If node.right is null

node.right = new Node(bid)

Else

Call addNode(node.right, bid)

End If

End If

End Method

Method inOrder(node as Node)

If node is not null

Call inOrder(node.left)

Print node.bid

Call inOrder(node.right)

End If

End Method

Method removeNode(node as Node, bidId as String) as Node

If node is null

Return null

If bidId < node.bid.bidId

node.left = Call removeNode(node.left, bidId)

ElseIf bidId > node.bid.bidId

node.right = Call removeNode(node.right, bidId)

Else

If node.left is null and node.right is null

Delete node

node = null

ElseIf node.left is not null and node.right is null

Replace node with node.left

ElseIf node.left is null and node.right is not null

Replace node with node.right

Else

Find the smallest node in the right subtree

Replace node.bid with smallest bid

node.right = Call removeNode(node.right, smallest bid.bidId)

End If

End If

Return node

End Method

End Class

// Main program

Method main()

Define bst as BinarySearchTree

Define csvPath, bidKey as String

Define ticks as ClockTime

Menu operations for loading, displaying, searching, and removing bids

End Method

**References**

https://learn.zybooks.com/zybook/CS-300-R4804-OL-TRAD-UG.24EW4/chapter/5/section/1

https://learn.zybooks.com/zybook/CS-300-R4804-OL-TRAD-UG.24EW4/chapter/5/section/2

https://learn.zybooks.com/zybook/CS-300-R4804-OL-TRAD-UG.24EW4/chapter/5/section/8