Shane Doyle 2/3/2023

Professor Thoma Assignment 5-2: Binary Search Tree

Code Reflection

For the binary search tree assignment, I was given starter code that defined a Bid structure, a tree node structure, and a BinarySearchTree class. In the BinarySearchTree class, there were several member functions which included InOrder, PostOrder, PreOrder, Insert, Remove, and Search. The purpose of this assignment was to finish writing each function and implement a binary search tree using data from the given CSV files. When the bid data is stored in the binary search tree, the user can display the bids, remove a bid, and search for a bid.

The only issue I had while working on this assignment was implementing the removeNode function. The part of that function that confused me was writing the 4 different scenarios that can happen and then restructuring the tree after deletion. To overcome this, I looked at the zybooks remove algorithm to better understand it.

Pseudocode

Function inOrder(Node\* node)

IF node does NOT equal nullptr

Call inOrder(node points to left)

OUTPUT Bid info

Call inOrder(node points to right)

Function postOrder(Node\* node)

IF node does NOT equal nullptr

Call postOrder(node points to left)

Call postOrder(node points to right)

OUTPUT Bid info

Function preOrder(Node\* node)

IF node does NOT equal nullptr

OUTPUT Bid info

Call preOrder(Node points to left)

Call preOrder(node points to right)

Function Insert(Bid bid)

IF root equals nullptr

root equals new Node(bid)

ELSE

Call addNode(root, bid)

Function addNode(Node\* node, Bid bid)

IF bid id is less than node bid id

IF left child equals nullptr

Left child equals new node

ELSE

Call addNode(node points to left, bid)

ELSE

IF right child equals nullptr

Right child equals new node

ELSE

Call addNode(node points to right, bid)

Function removeNode(Node\* node, string bidId)

IF node equals nullptr

Return node

IF bidId is less than node bidId

Left child equals removeNode(node points to left, bidId)

ELIF bidId is greater than node bidId

Right child equals removeNode(node points to right, bidId)

ELSE

IF node has no children

Delete node

Node equals nullptr

ELIF node has only left child

Declare Node pointer temp equal to node

node equals left

Delete temp

ELIF node has only right child

Declare Node pointer temp equal to node

node equals right

Delete temp

ELSE

Declare Node pointer temp equal to right

WHILE temp points to left does NOT equal nullptr

temp equals temp points to left

Set nodes bid equal to temps bid

node points to right equals removeNode( node points to right, temp bidId)

Function Search(string bidId)

Declare Node pointer cur equal to root

WHILE cur does NOT equal nullptr

IF bidIds are equal

Return cur bidId

IF bidId is less than cur bidId

cur equals cur left

ELSE

cur equals cur right