5-2 Assignment: Binary Search Tree

David Vega

Southern New Hampshire University

CS-300 DSA: Analysis and Design

Professor David Ostrowski

July 31, 2022

# [Title Here, up to 12 Words, on One to Two Lines]

# REFLECTION

The purpose of this program is to load a CSV file that contains bid information that can be displayed and manipulated by the user. Users can display, search, and remove bids. The program utilizes a binary search tree to store bid information. A binary search tree is a sorted data structure that utilizes a root node’s key to order the nodes. The root node is branches into a left subtree and a right subtree respectively. Any new nodes keys are compared to the root node and if less than the root node key will be pushed to the left subtree while nodes greater than the root node will be pushed to the right subtree. This method has proven very useful in searching.

This assignment was a little difficult because one must be comfortable with pointers and I still have much to learn on this topic. The fundamentals of pointers are easy to understand but in larger applications it can be somewhat convoluted. I struggled a bit with recursion and still can’t honestly say that I fully comprehend the mechanics of this, but I understand that it can be useful in breaking down complex problems into smaller chunks.

START PROGRAM

// Define a binary search tree to hold all bids

DEFINE a new BinarySearchTree() called “bst”

DECLARE a bid of type Bid

INITIALIZE int called “choice” // used to store the user input to traverse the menu

WHILE choice IS NOT equal to 9

EXECUTE

PRINT user menu:

1. Load Bids
2. Display All Bids
3. Find Bid
4. Remove Bid
5. Exit

READ IN user input and store to variable choice for menu selection.

Text

Description automatically generated

**CHOICE 1 - (Load Bids)**

SET variable ticks equal to clock() method. Stores starting clock tick.

INVOKE loadBids() and pass in csvPath and instance of binary search tree class

INITIALIZE csv file

PARSE csv file and stores data into bid object

DEFINE struct to store collection of bids

RETURN bids

CALCULATE elapsed time

Clock() – ticks

PRINT elapsed time

Text

Description automatically generated

**CHOICE 2 (Display All Bids)**

INVOKE bst->inOrder() method // prints all elements in order of bidId

BREAK;

Text

Description automatically generated

**CHOICE 3 - (Find Bid)**

SET variable ticks equal to clock() method. Stores starting clock tick.

SET bid equal to bst->Search() method and pass in bidkey as a parameter.

IF bidId IS NOT empty

INVOKE displayBid() method and pass in bid;

ELSE

PRINT “not found” message;

PRINT clock ticks

PRINT clock ticks per second

BREAK;

Text

Description automatically generated

**CHOICE 4 - (Remove Bid)**

INVOKE bst->Remove() method and pass in bidId;

// Remove() method invokes the removeNode() method which takes in a node and bidId

// and traverses down the tree to locate the bidId you want to delete

IF node is empty

RETURN node

IF bidId is less than the bidID of the node

SET node->left equal to removeNode() which takes in left node and bidId //recursive call

ELSE

SET node->right equal to removeNode() which takes in right node and bidId // recursive call

// if node has no children

IF node->left AND node->right are empty

DELETE node

SET node = null pointer

// if node has a left child

ELSE IF node->left is NOT empty and node->right is empty

INITIALIZE temp node and set equal to node // temporarily hold node

SET node equal to left node->left

DELETE temp node

// if node has a right child

ELSE IF node-> is empty and node-> right is NOT empty

INITIALIZE temp node and set equal to node // temporarily hold node

SET node equal to left node->right

DELETE temp node

// if node has two children

ELSE

INITIALIZE temp node and set equal to node->right // temporarily hold node

WHILE temp-> left IS NOT empty

SET temp equal to temp->left

SET node->bid = temp->bid

SET node->right equal to recursive call to removeNode() pass in node-> and bidId

RETURN node

Text

Description automatically generatedText

Description automatically generated

ELSE

PRINT “Goodbye”

Text

Description automatically generated

RETURN 0

END PROGRAM