

Project 5 Pseudocode

Insert:

newNode \leftarrow new node containing new data

If tree is empty then

 Set root to newNode

 Increment size of tree

 Return success status code

Else

 currentNode \leftarrow root node

 parentNode \leftarrow currentNode

 While currentNode is not null

 parentNode \leftarrow currentNode

 If new data < data of currentNode

 currentNode \leftarrow left child of currentNode

 Else if new data > data of currentNode

 currentNode \leftarrow right child of currentNode

 Else

 Return error status code (There is duplicate data!)

 (parentNode is now a leaf node)

 If new data < parentNode then

 Set left child of parentNode to newNode

 Else (the case where new data > parentNode)

 Set right child of parentNode to newNode

Increment size of tree

Return success status code

Contains:

currentNode \leftarrow root node

While currentNode is not null

 If data to be found < data of currentNode

 currentNode \leftarrow left child of currentNode

 Else if data to be found > data of currentNode

 currentNode \leftarrow right child of currentNode

 Else

 Return success status code (We found the data!)

Return error status code (We reached a leaf node without finding the data.)

Preorder Helper:

arr[index] \leftarrow data of root node

Increment index

Call preorder_helper() on left child of the root node

Call preorder_helper() on right child of the root node

Preorder Traversal:

arr \leftarrow empty array

index \leftarrow 0

Call preorder_helper() on the root node

Return the array of values

Duplicate Without:

dataArray \leftarrow Call preorder_traversal() on the root node to get an array of all of the values

newTree \leftarrow initialize a new BST

foundDataRemoved \leftarrow -1

For each value in dataArray

 If value = data_removed

 foundDataRemoved \leftarrow index of current value

 Else

 Insert a new node into newTree with the value

If foundDataRemoved \geq 0 (we found the data to be removed at some point)

 Destroy the data to be removed

Else

 Return NULL (The data to be removed is not in the BST.)

Destroy the original tree

Return the new tree

Destroy Node:

Call destroy_node() on left child of the node

Call destroy_node() on right child of the node

If destroy_data == 1

 Destroy the data

Destroy the node

Destroy Tree:

Call destroy_node() on the root node

Destroy the tree