Design

Attributes:

We will create two classes, the node class and the binary search class.

Node Class:

Value: The value (key) in the node
Left Branch: Define what the left side of the node should be
Right Branch: Define what the right side of the node should be

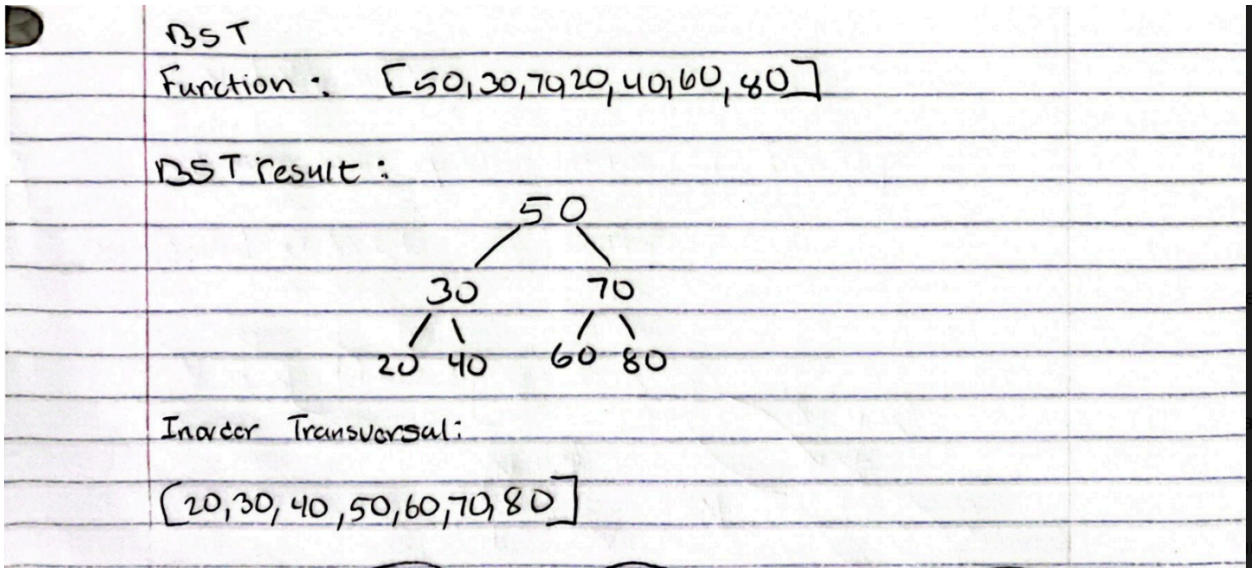
Binary Search Class:

Root Node: Establish a root node for the BST

Function needed:

Add function: Adds a value in the right location based on our BST rules
Remove function: Removes a value and replaces it appropriately.
Find minimum value function: A function to help us find the minimum value to enforce the BST rules
Tree transversal: Function to create an Inorder Transversal of our BST

How it would look like:



Testing

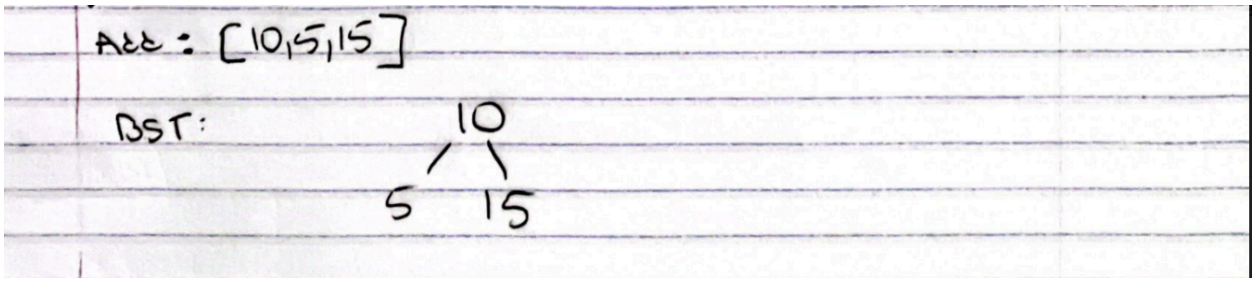
Add:

Test 1:

Add a value to an empty tree. The expected value should be the root value of that input.

Test 2:

Add three values [10,5,15]
Expected output:



Remove:

Test 1:

Remove a node with one child. If our list is [10,5,15,12] and we remove 15, our expected output should be a BST tree that consists of [10,5,12]

Test 2:

Remove a leaf. Using our list [10,5,15]. If we remove 5, our expected outputs should be the root 10 and a 15 on the right side.

Minimum:

Test 1:

We will create a tree, and find the minimum value in that tree. Then we will remove that value, then find the new minimum value. We will do that one more time. We are doing this multiple times to make sure that our function can adapt to the changing minimum value.

In order:

Test 1:

Find the inorder transversal of [10,5,15]. Expected outputs should be [5,10,15]

Test 2:

Find the inorder transversal of a bigger BST. We will add more valuest to the list and verify if the inorder transversal is it.