Jonathan Roberts
November 17, 2021
Binary Search Trees

Findings

Any of the search, deletion, or insertion operations were very efficient, none of them took even one second to run. In the second task where the overlapping values from BST1 needed to be inserted into BST2, this task took the longest, this is probably because I used search, deletion and insertion to perform the task, and the time between the different traversals didn't seem to vary much or rather the time difference was negligible. Even when comparing values between BSTs, they proved to be more efficient than running a sorting algorithm. Clearing the BSTs only took .0001 second, this would probably vary based on how someone chooses to clear them. Overall BSTs are extremely efficient even when performing consecutive searches, deletions, and insertions. The time table is listed below:

*timer for each traversal includes the insertion, deletion, and search where applicable

Binary Search Tree 1 height and inorder traversal

Height: 13
Time: .0012 seconds

Insert 10 unique values and insert them into binary search tree two where there is overlap with BST1.

Predoreder time: .002
Inorder time: .003
Postorder time: .002

Find and remove any values of BST2 from BST1. Print height, number of nodes, and inorder output of the modified BST1 tree.

Number of nodes: 96
Height: 13
Deletion of values in BSt2 from Bst1 took: .0005

Clear the binary search trees. Print whether trees are empty before and after clear operation.

Time to check whether tree is empty or not and to clear: .0001