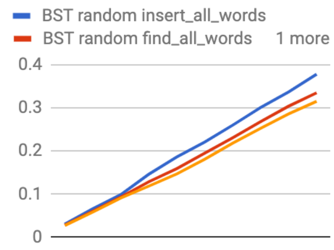
Binary Search Tree (BST) Unsorted:



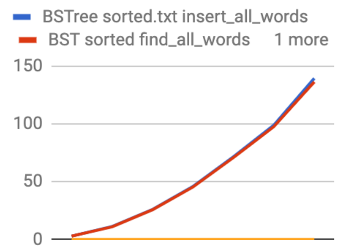
Observed time complexity:

Insert: O(log n). Inserting n words takes O(log n) * n = O(nlogn) time, as observed in the graph.

Find: O(log n). Finding n words takes O(log n) * n = O(nlogn) time, as observed in the graph.

Remove: O(log n). Removing n words takes O(log n) * n = O(nlogn) time.

Binary Search Tree (BST) Sorted:



Observed time complexity:

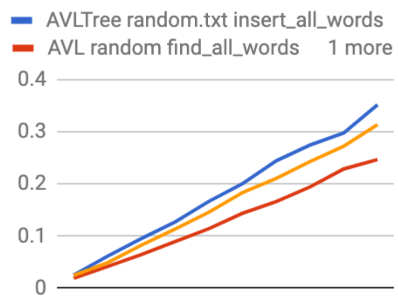
Insert: O(n). Inserting n words takes O(n) * n = O(n^2) time, as observed in the graph.

Find: O(n). Finding n words takes O(1) * n = O(n^2) time, as observed in the graph.

Remove: O(1). Removing n words takes O(1) * n = O(n) time, as observed in the graph.

Analysis: Insert and find take O(n) time because an unbalanced tree can degrade to O(n) performance when inserting in a sorted order. Remove likely takes O(1) time because removing is in the same order as inserting, and so it is effectively removing the first element each time.

AVL Tree Unsorted:



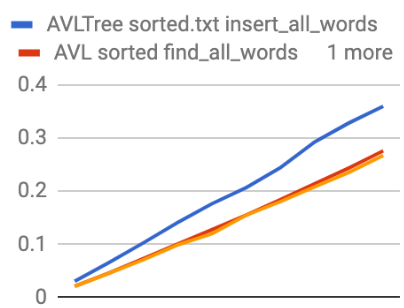
Observed time complexity:

Insert: O(log n). Inserting n words takes O(log n) * n = O(nlogn) time, as observed in the graph.

Find: O(log n). Finding n words takes O(log n) * n = O(nlogn) time, as observed in the graph.

Remove: O(log n). Removing n words takes O(log n) * n = O(nlogn) time.

AVL Tree Sorted:



Observed time complexity:

Insert: O(log n). Inserting n words takes O(log n) * n = O(nlogn) time, as observed in the graph.

Find: O(log n). Finding n words takes O(log n) * n = O(nlogn) time, as observed in the graph.

Remove: O(log n). Removing n words takes O(log n) * n = O(nlogn) time.

Analysis: As expected, keeping the tree balanced makes the most significant impact when inserting into the tree in a sorted order. The AVL tree was able to perform much better.