Tim M. Lael
tmlz5d

Binary Search Trees

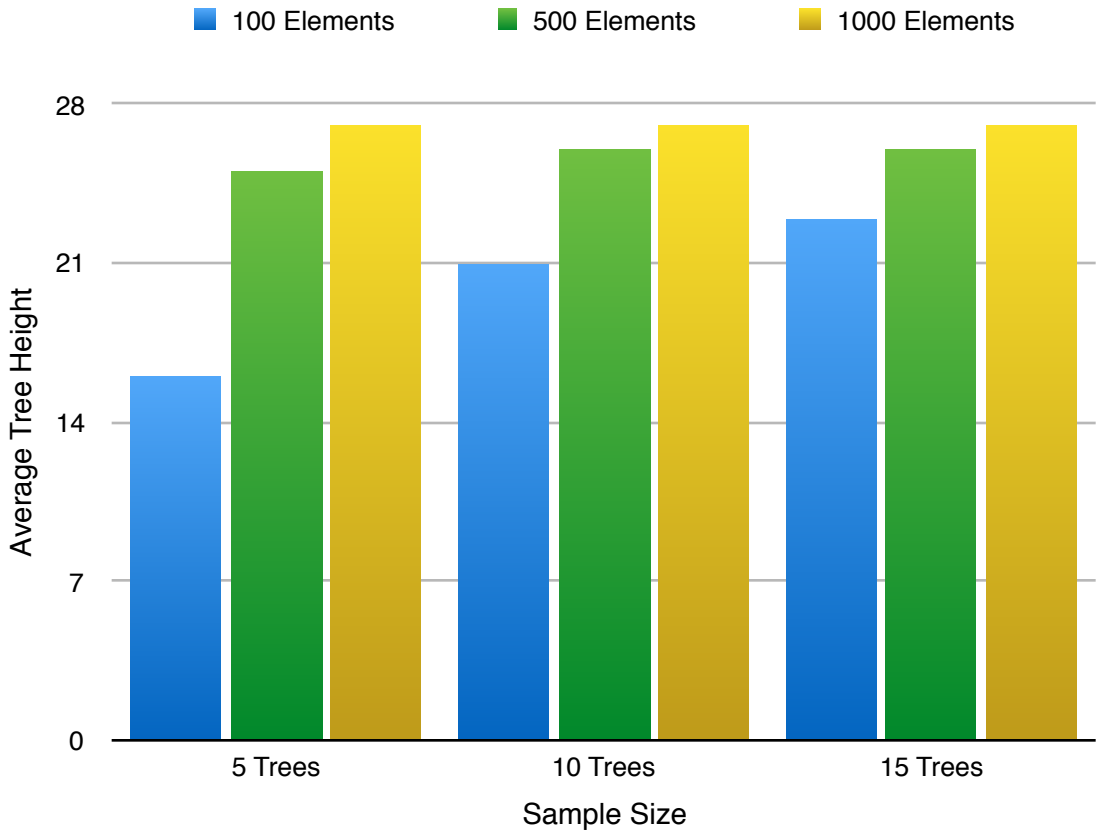
Duplicate Handling:

In my implementation of binary search trees, I initially ignored duplicate values in part A. This was because, in the definition of BSTs, there are normally no duplicates allowed. In the second part, Part B, I accounted for duplicated keys by implementing a count property in the structures used to represent the tree nodes. If a duplicate is encountered, it is not inserted, but the counter for the node containing the duplicate value would be incremented. while this property (node->counter) could be accessed, it was not displayed (or accessed) since the display was beyond the scope of this project.

Tree Height Calculation Efficiency:

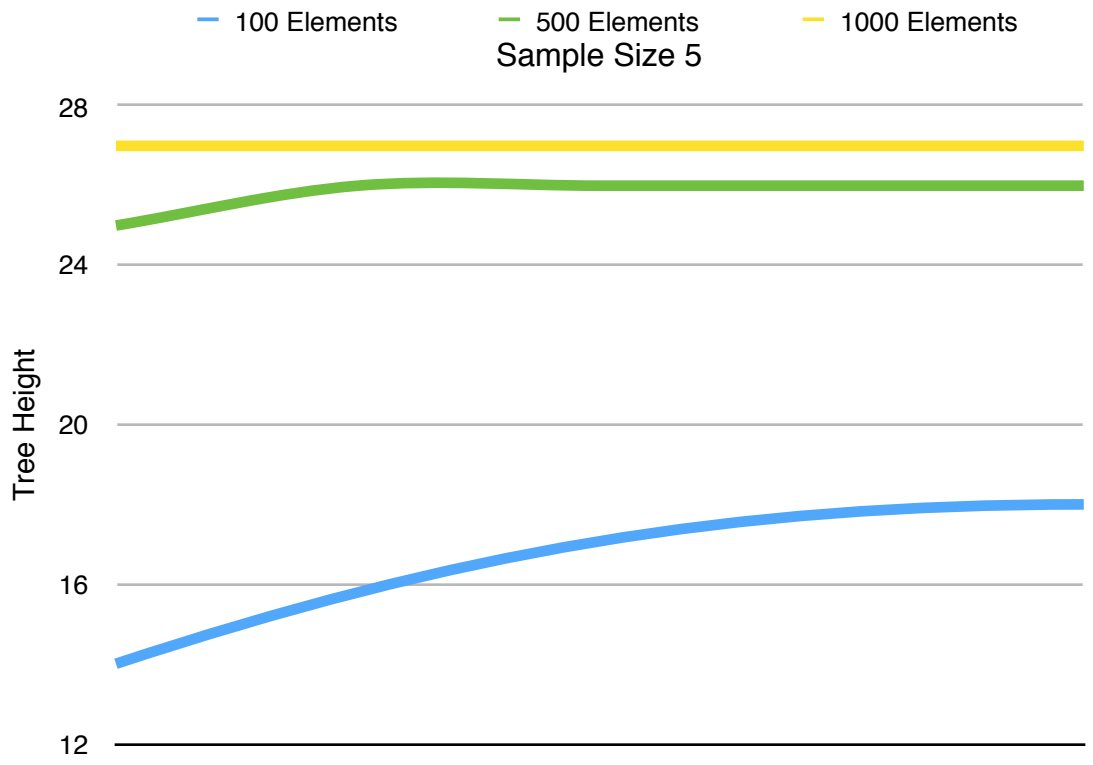
The calculation and display of tree height has O(n) time complexity since, in order to calculate height, each node is visited until we reach the bottom of the tree. This operation nearly identical to each of the traversal types. The difference is that, instead of displaying values (as we would in traversal), we increment the height value for each recursive call. So, in the end, we could visit each node leading to O(n) running time.

In the charts below, you can see how tree height grows in relation to the number of nodes inserted. Since the height is logarithmic, you can notice the height often changes more dramatically in trees with fewer nodes.



Tim M. Lael
tmlz5d

Sample	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Tree Size															
100	14	16	17	18	18										
500	25	26	26	26	26										
1000	27	27	27	27	27										
100	19	19	20	21	21	22	22	22	22	22					
500	26	26	26	26	26	26	26	26	26	26					
1000	27	27	27	27	27	27	27	27	27	27					
100	22	23	23	23	23	23	23	23	23	24	24	24	24	25	25
500	26	26	26	26	26	27	27	27	27	27	27	27	27	27	27
1000	27	27	27	28	28	28	28	28	28	28	28	28	28	28	28



Tim M. Lael
tmlz5d