Joshua Silva
ECE 2230
Spring 2024
MP5

Performance Evaluation

To evaluate the performance of our Binary Search Tree, we ran tests with large generated trees that ranged from having levels 1 to 20, with one million accesses done to test the successful search time of each search policy. From reading the Standish Textbook, we know that Binary Search Trees are intended to have a search time of O(logn), with the intention of decreasing search times for the user, as we consistently decrease the amount of nodes to search by half. However, there is a possibility that the tree can become heavy on its left or right side, decreasing the efficiency of search times, which is seen when randomly inserted keys into a tree over time. Typically, randomly inserting into a tree doesn't result in deep skinny trees & stays mostly balanced, but the efficiency of search times is decreased by around 38.6%. Potentially, a tree can become deep and skinny, especially if the user keeps inserting keys that are less and less, or more and more than earlier keys. This can become the worst case scenario, where the tree operates more like a sequential list, not utilizing its O(logn) nature and instead approaching O(n), or flat out O(n) in the absolute worst case.

An optimally constructed tree has a search time of O(logn), due to being made in a way to be as balanced as possible, having leaves on one level or on two adjacent levels. With this in mind, we see in our graph for Optimal Insertion Times, that both BST and AVL search policies are identical. Since the tree was made to be as balanced as possible, AVL balancing is not necessary to decrease search times, so both BST and AVL will be the same.

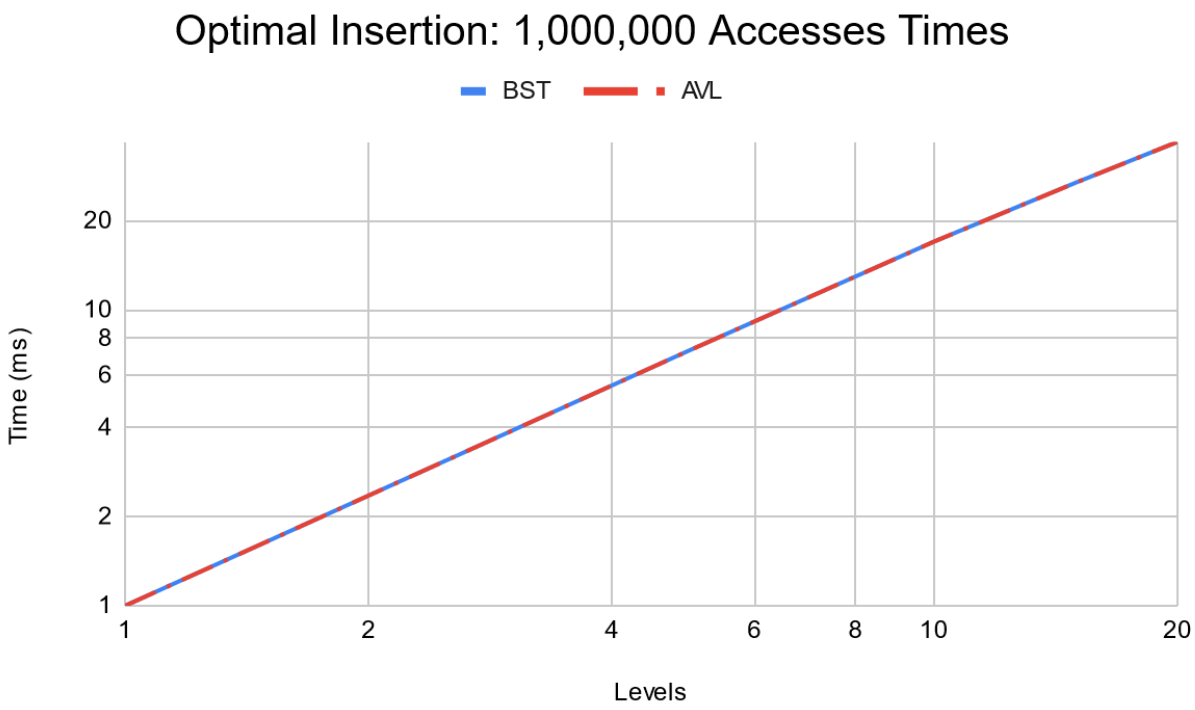
A randomly constructed tree still has a search time of O(logn), due to the rarity of deep skinny trees but expected to have a decreased efficiency of 38.6%. With our data collected matching this result. Comparing our BST & AVL at 20 levels, we see that 49.81/37.77 = 1.32, with BST being about 32% less efficient at 20 levels than AVL. AVL has barely changed search times, which will also be seen with poor trees, as AVL balances the tree after each insert, preventing the decrease in efficiency with a randomly constructed tree.

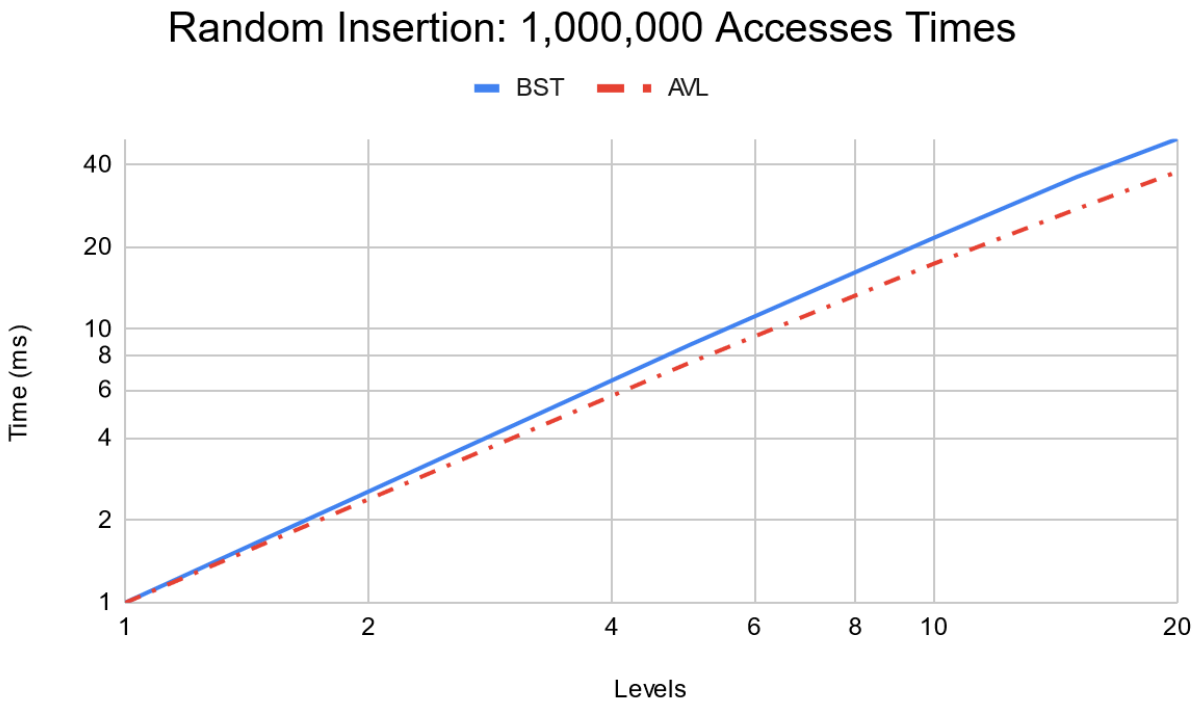
A poorly constructed tree can approach a search time of O(n), due to how common deep skinny trees have become. Our data collected matches this result, as BST search times rapidly increase at higher levels. However, AVL balances the Tree at each insert before deep skinny subtrees form. So search times are still similar to Optimal and Random Insertions for AVL.

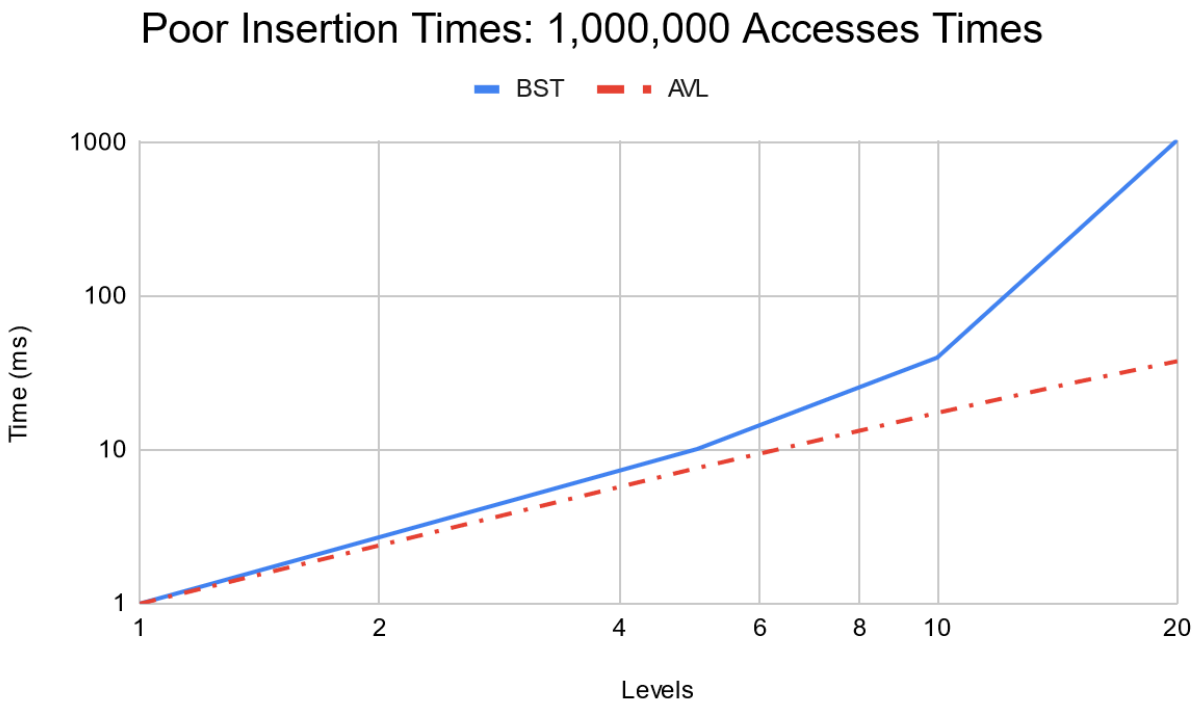
Overall, looking at our raw data, we see that our search times match the expected amount by 3 significant figures. Unsuccessful search times tend to be longer, which is expected due to

reaching the end of the tree before concluding the key is not there, unlike successful which can stop anywhere in the tree. Also, when looking at our graphs, we can conclude that our implementation does give successful searches a complexity of O(logn). To prove this, we plotted both levels(#nodes) and time searched in logarithmic scale, meaning a straight line shows a logarithmic relationship between time(y) and levels of our tree(x). For both Optimal and Random, a logarithmic pattern is shown for BST and AVL, with BST having worse performance in random, while still maintaining a straight line. For Poor, AVL still shows a logarithmic pattern, however BST starts to lose its logarithmic pattern at very high levels of our tree.

Graphs (Successful Searches):




Random Insertion: 1,000,000 Accesses Times
BST
AVL
Time (ms)
40
20
10
8
6
4
2
1
1
2
4
6
8
10
20
Levels


Poor Insertion Times: 1,000,000 Accesses Times
BST
AVL
Time (ms)
1000
100
10
1
1
2
4
6
8
10
20
Levels

Raw Data:

		BST Optimal Insertion Times (ms)			
		Accesses			
		1,000,000			
		Expected		Actual	
		Successful	Unsuccessful	Successful	Unsuccessful
Levels	1	1	2	1	2
	5	7.32258	10	7.32593	10
	10	17.0196	20	17.0231	20
	15	27.0009	30	26.9934	30
	20	37	40	36.9974	40

		AVL Optimal Insertion Times (ms)			
		Accesses			
		1,000,000			
		Expected		Actual	
		Successful	Unsuccessful	Successful	Unsuccessful
Levels	1	1	2	1	2
	5	7.32258	10	7.32593	10
	10	17.0196	20	17.0231	20
	15	27.0009	30	26.9934	30
	20	37	40	36.9974	40

		BST Random Insertion Times (ms)			
		Accesses			
		1,000,000			
		Expected		Actual	
		Successful	Unsuccessful	Successful	Unsuccessful
Levels	1	1	2	1	2
	5	8.80645	11.4375	8.80468	11.4361
	10	21.6432	24.6191	21.6467	24.6319
	15	36.1075	39.1063	36.107	39.1198
	20	49.8099	52.8099	49.8141	52.8324

		AVL Random Insertion Times (ms)			
		Accesses			
		1,000,000			
		Expected		Actual	
		Successful	Unsuccessful	Successful	Unsuccessful
Levels	1	1	2	1	2
	5	7.58065	10.25	7.5789	10.2482
	10	17.4184	20.3984	17.4162	20.4009
	15	27.572	30.5711	27.5744	30.5714
	20	37.7698	40.7697	37.7738	40.7735

		BST Poor Insertion Times (ms)			
		Accesses			
		1,000,000			
		Expected		Actual	
		Successful	Unsuccessful	Successful	Unsuccessful
Levels	1	1	2	1	2
	5	10.1613	12.75	10.1535	12.7504
	10	39.9795	42.9375	39.9684	42.9819
	15	268	270.992	268.102	270.927
	20	1042	1045	1042.95	1046.48

		AVL Poor Insertion Times (ms)			
		Accesses			
		1,000,000			
		Expected		Actual	
		Successful	Unsuccessful	Successful	Unsuccessful
Levels	1	1	2	1	2
	5	7.64516	10.3125	7.64357	10.3153
	10	17.4712	20.4512	17.4677	20.452
	15	27.8023	30.8014	27.8049	30.8025
	20	37.8828	40.8827	37.8868	40.8808