EECS 560: Lab 05 Report

Complexity Analysis of Functions in Binary Search Trees

Aubrey Bud Linville

2819130

1. Overall Organization

This experiment was written in C++, with the actually complexity analysis portion of the experiment written in .cpp files that were instantiated by the project's makefile. The only abstract data type needed was the binary search tree, although it would be possible to also include a queue to create a more efficient level order traversal in future implementations. My implementation of the binary search tree used templated binary nodes linked together using pointer variables to store data read in from an input file. Many of the functions in this lab required the use of *helper* functions for correct implementation. Being a tree data type, many of these helper functions were inherently recursive. One important such function was *getParentToDeleteNode()*, which returned the parent to a node that was about to be deleted. It was important to use this function so as to be able to set the parent's children pointers to *nullptr*, so as to avoid memory leaks.

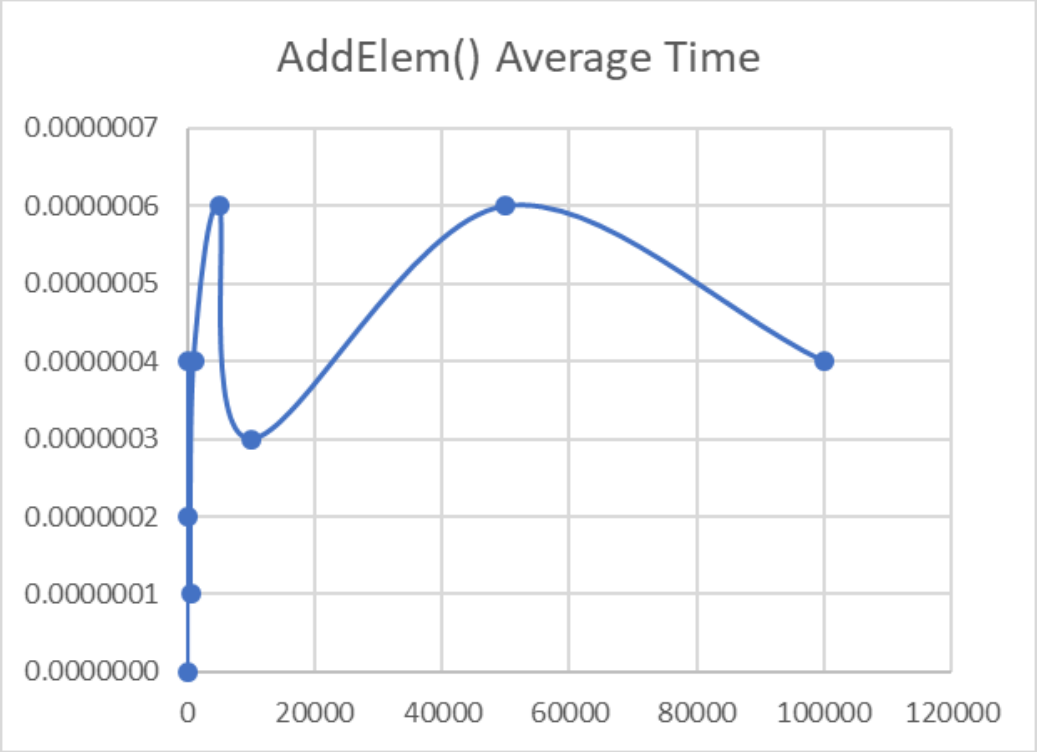
2. Tabulated Data

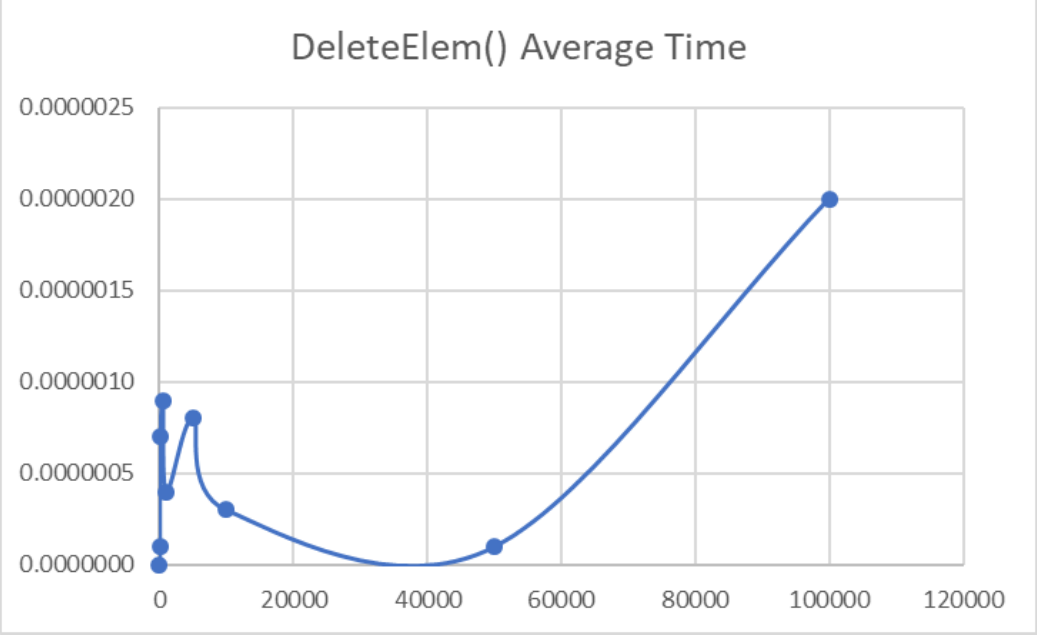
Adding Element																	
Size:	10		50		100		500		1000		5000		10000		50000		100000	
	Test	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*
	T1	0.000000	T1	0.000000	T1	0.000000	T1	0.000001	T1	0.000001	T1	0.000001	T1	0.000000	T1	0.000001	T1	0.000000
	T2	0.000000	T2	0.000000	T2	0.000000	T2	0.000000	T2	0.000000	T2	0.000000	T2	0.000001	T2	0.000001	T2	0.000000
	T3	0.000000	T3	0.000000	T3	0.000001	T3	0.000000	T3	0.000000	T3	0.000000	T3	0.000000	T3	0.000000	T3	0.000000
	T4	0.000000	T4	0.000001	T4	0.000000	T4	0.000000	T4	0.000000	T4	0.000001	T4	0.000000	T4	0.000001	T4	0.000000
	T5	0.000000	T5	0.000000	T5	0.000000	T5	0.000000	T5	0.000001	T5	0.000000	T5	0.000000	T5	0.000000	T5	0.000001
	T6	0.000000	T6	0.000000	T6	0.000001	T6	0.000000	T6	0.000000	T6	0.000000	T6	0.000000	T6	0.000000	T6	0.000001
	T7	0.000000	T7	0.000000	T7	0.000000	T7	0.000000	T7	0.000000	T7	0.000001	T7	0.000001	T7	0.000001	T7	0.000000
	T8	0.000000	T8	0.000000	T8	0.000000	T8	0.000000	T8	0.000001	T8	0.000001	T8	0.000001	T8	0.000001	T8	0.000001
	T9	0.000000	T9	0.000001	T9	0.000001	T9	0.000000	T9	0.000000	T9	0.000001	T9	0.000000	T9	0.000001	T9	0.000000
	T10	0.000000	T10	0.000000	T10	0.000000	T10	0.000000	T10	0.000000	T10	0.000000	T10	0.000000	T10	0.000000	T10	0.000000
	T(AVG)	**0.000000**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000000**
Deleting Element																		
Size:	10		50		100		500		1000		5000		10000		50000		100000	
	Test	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*
	T1	0.000000	T1	0.000000	T1	0.000001	T1	0.000000	T1	0.000001	T1	0.000001	T1	0.000000	T1	0.000000	T1	0.000003
	T2	0.000000	T2	0.000000	T2	0.000001	T2	0.000001	T2	0.000001	T2	0.000001	T2	0.000000	T2	0.000000	T2	0.000002
	T3	0.000000	T3	0.000000	T3	0.000000	T3	0.000001	T3	0.000000	T3	0.000000	T3	0.000001	T3	0.000001	T3	0.000002
	T4	0.000000	T4	0.000000	T4	0.000001	T4	0.000001	T4	0.000000	T4	0.000001	T4	0.000000	T4	0.000001	T4	0.000002
	T5	0.000000	T5	0.000000	T5	0.000001	T5	0.000001	T5	0.000000	T5	0.000001	T5	0.000001	T5	0.000000	T5	0.000002
	T6	0.000000	T6	0.000000	T6	0.000001	T6	0.000001	T6	0.000000	T6	0.000000	T6	0.000000	T6	0.000001	T6	0.000002
	T7	0.000000	T7	0.000000	T7	0.000000	T7	0.000001	T7	0.000001	T7	0.000001	T7	0.000000	T7	0.000001	T7	0.000002
	T8	0.000000	T8	0.000000	T8	0.000001	T8	0.000001	T8	0.000000	T8	0.000001	T8	0.000000	T8	0.000000	T8	0.000002
	T9	0.000000	T9	0.000000	T9	0.000000	T9	0.000001	T9	0.000000	T9	0.000001	T9	0.000000	T9	0.000000	T9	0.000002
	T10	0.000000	T10	0.000000	T10	0.000001	T10	0.000001	T10	0.000000	T10	0.000001	T10	0.000000	T10	0.000001	T10	0.000002
	T(AVG)	**0.000000**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000000**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000002**
Checking if Element Exists																		
Size:	10		50		100		500		1000		5000		10000		50000		100000	
	Test	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*	*Test*	*Time*
	T1	0.000000	T1	0.000001	T1	0.000001	T1	0.000009	T1	0.000011	T1	0.000079	T1	0.000213	T1	0.000788	T1	0.000965
	T2	0.000000	T2	0.000001	T2	0.000002	T2	0.000008	T2	0.000011	T2	0.000079	T2	0.000213	T2	0.000778	T2	0.000968
	T3	0.000000	T3	0.000001	T3	0.000001	T3	0.000008	T3	0.000010	T3	0.000080	T3	0.000213	T3	0.000777	T3	0.000964
	T4	0.000000	T4	0.000000	T4	0.000001	T4	0.000008	T4	0.000011	T4	0.000081	T4	0.000213	T4	0.000788	T4	0.000966
	T5	0.000000	T5	0.000001	T5	0.000001	T5	0.000008	T5	0.000010	T5	0.000078	T5	0.000213	T5	0.001074	T5	0.000981
	T6	0.000000	T6	0.000001	T6	0.000001	T6	0.000008	T6	0.000010	T6	0.000078	T6	0.000213	T6	0.000796	T6	0.000973
	T7	0.000000	T7	0.000001	T7	0.000001	T7	0.000009	T7	0.000010	T7	0.000081	T7	0.000213	T7	0.000787	T7	0.000968
	T8	0.000000	T8	0.000000	T8	0.000001	T8	0.000007	T8	0.000010	T8	0.000079	T8	0.000213	T8	0.000790	T8	0.000963
	T9	0.000000	T9	0.000000	T9	0.000001	T9	0.000007	T9	0.000010	T9	0.000079	T9	0.000213	T9	0.000784	T9	0.000967
	T10	0.000000	T10	0.000000	T10	0.000001	T10	0.000008	T10	0.000010	T10	0.000079	T10	0.000213	T10	0.000796	T10	0.000955
	T(AVG)	**0.000000**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000001**	**T(AVG)**	**0.000008**	**T(AVG)**	**0.000010**	**T(AVG)**	**0.000079**	**T(AVG)**	**0.000213**	**T(AVG)**	**0.000816**	**T(AVG)**	**0.000967**

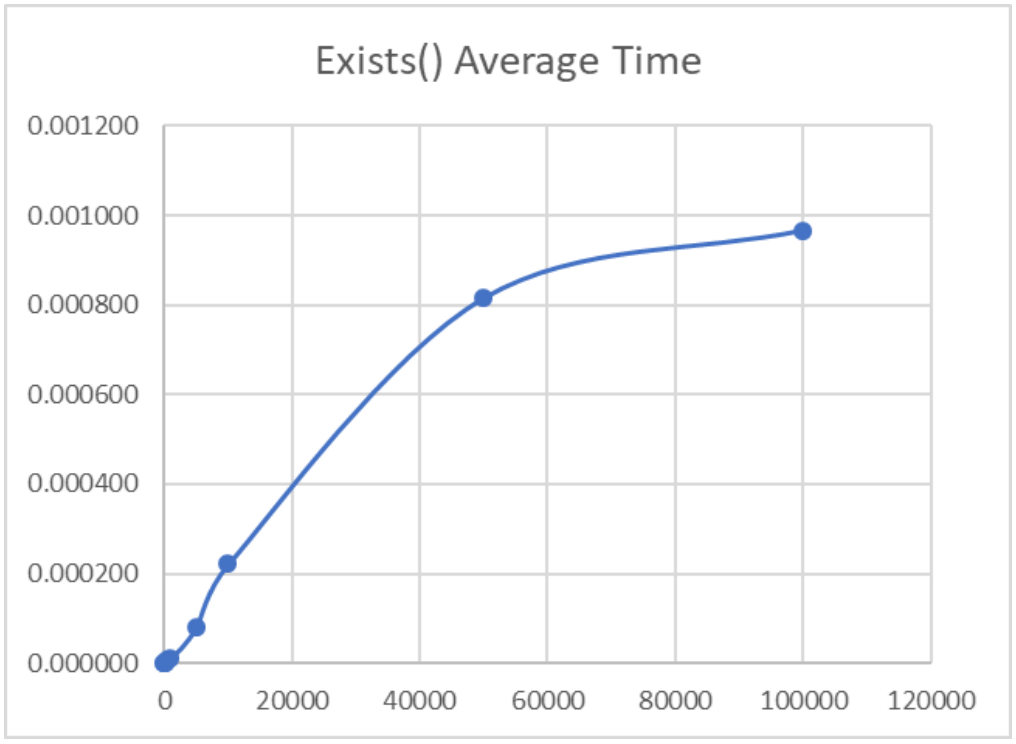
3. Graphs

The first two graphs are not accurate representations of the actual complexity of the functions they represent. Rather, the functions are efficient enough such that the sample sizes, as specified by the lab guidelines, are not big enough to truly show the O(log(n)) nature.

In the second picture, you can begin to see the logarithmic growth as it rises rapidly, but you never get to see it slow down.






Exists() Average Time
0.001200
0.001000
0.000800
0.000600
0.000400
0.000200
0.000000
0
20000
40000
60000
80000
100000
120000