EECS 560: Lab 06 Report

Complexity Analysis of Functions in a 3-ary Min Heap

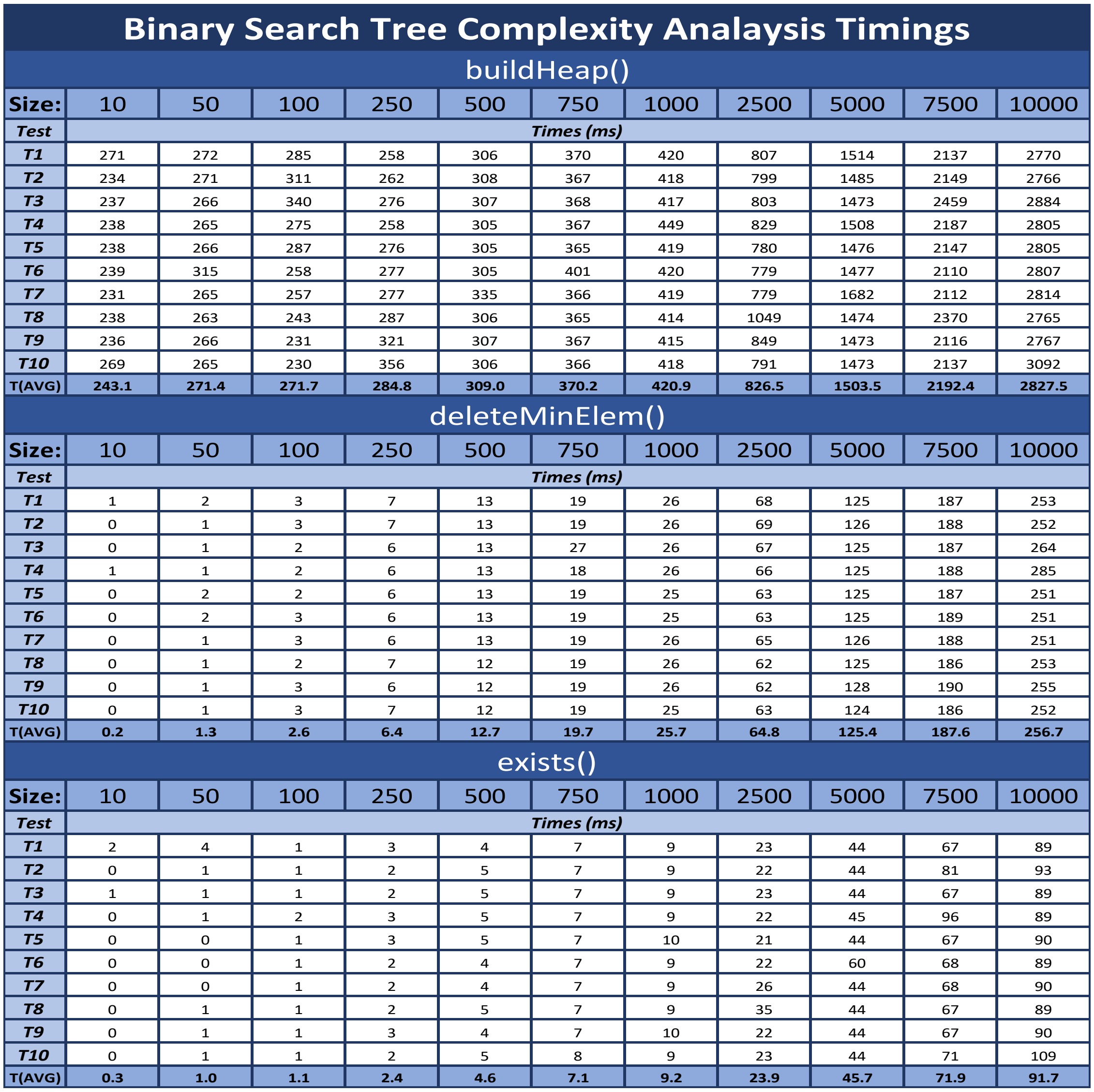
Aubrey Bud Linville

2819130

1. Overall Organization

This experiment was written in C++, with the actual 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 a 3-ary min-heap. This heap was implemented using an array of 15000 elements, but could be expanded by changing one line of code if larger heaps are needed. Data is read in from a text file specified by the user from a command line parameter. If no input file is give, my program used a default *Data.txt* file included in the package. The data is stored in an array that is then passed to the heap’s constructor so that it can build a 3-ary min heap. All typical heap functionality is implemented including deletion, insertion, and searching methods. The analysis files calculate average times that their respective functions take and displays it to the terminal.

2. Tabulated Data



Binary Search Tree Complexity Analaysis Timings

buildHeap()

Size:	10	50	100	250	500	750	1000	2500	5000	7500	10000
Test	***Times (ms)***										
T1	271	272	285	258	306	370	420	807	1514	2137	2770
T2	234	271	311	262	308	367	418	799	1485	2149	2766
T3	237	266	340	276	307	368	417	803	1473	2459	2884
T4	238	265	275	258	305	367	449	829	1508	2187	2805
T5	238	266	287	276	305	365	419	780	1476	2147	2805
T6	239	315	258	277	305	401	420	779	1477	2110	2807
T7	231	265	257	277	335	366	419	779	1682	2112	2814
T8	238	263	243	287	306	365	414	1049	1474	2370	2765
T9	236	266	231	321	307	367	415	849	1473	2116	2767
T10	269	265	230	356	306	366	418	791	1473	2137	3092
T(AVG)	**243.1**	**271.4**	**271.7**	**284.8**	**309.0**	**370.2**	**420.9**	**826.5**	**1503.5**	**2192.4**	**2827.5**

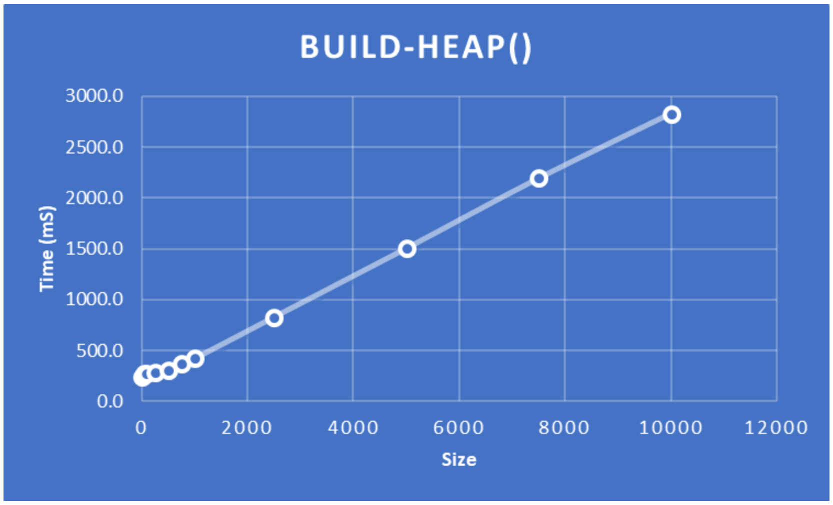
deleteMinElem()

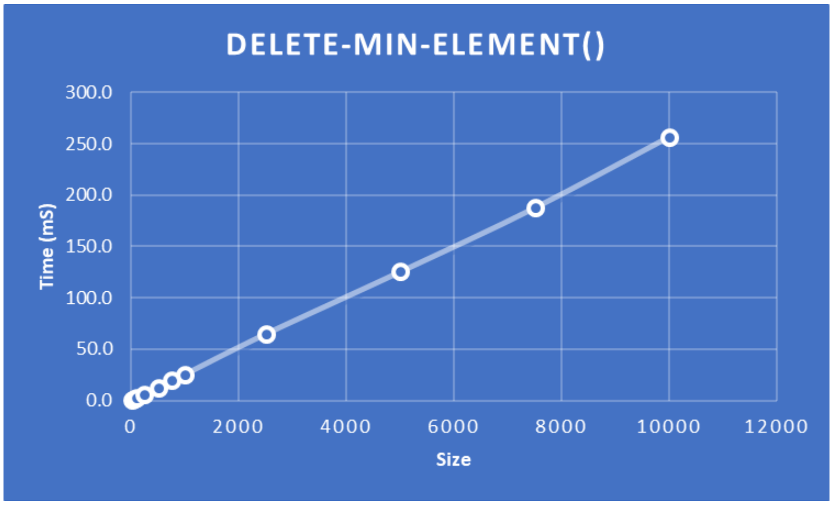
Size:	10	50	100	250	500	750	1000	2500	5000	7500	10000
Test	***Times (ms)***										
T1	1	2	3	7	13	19	26	68	125	187	253
T2	0	1	3	7	13	19	26	69	126	188	252
T3	0	1	2	6	13	27	26	67	125	187	264
T4	1	1	2	6	13	18	26	66	125	188	285
T5	0	2	2	6	13	19	25	63	125	187	251
T6	0	2	3	6	13	19	25	63	125	189	251
T7	0	1	3	6	13	19	26	65	126	188	251
T8	0	1	2	7	12	19	26	62	125	186	253
T9	0	1	3	6	12	19	26	62	128	190	255
T10	0	1	3	7	12	19	25	63	124	186	252
T(AVG)	**0.2**	**1.3**	**2.6**	**6.4**	**12.7**	**19.7**	**25.7**	**64.8**	**125.4**	**187.6**	**256.7**

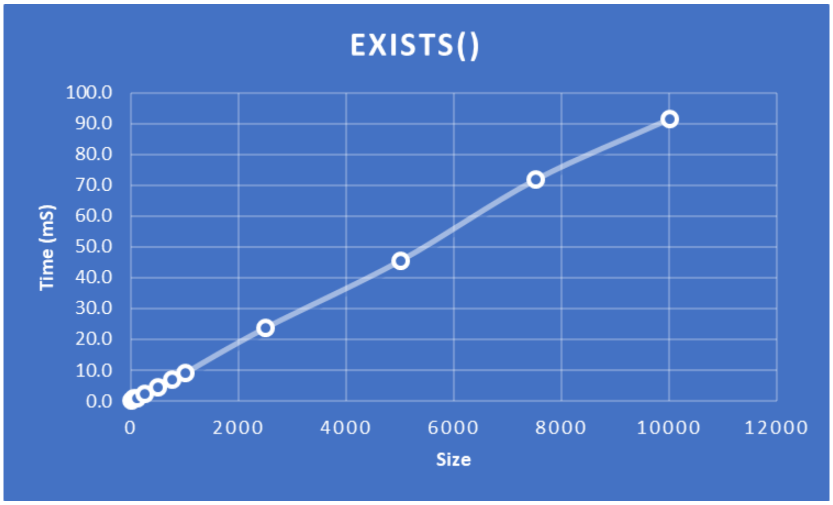
exists()

Size:	10	50	100	250	500	750	1000	2500	5000	7500	10000
Test	***Times (ms)***										
T1	2	4	1	3	4	7	9	23	44	67	89
T2	0	1	1	2	5	7	9	22	44	81	93
T3	1	1	1	2	5	7	9	23	44	67	89
T4	0	1	2	3	5	7	9	22	45	96	89
T5	0	0	1	3	5	7	10	21	44	67	90
T6	0	0	1	2	4	7	9	22	60	68	89
T7	0	0	1	2	4	7	9	26	44	68	90
T8	0	1	1	2	5	7	9	35	44	67	89
T9	0	1	1	3	4	7	10	22	44	67	90
T10	0	1	1	2	5	8	9	23	44	71	109
T(AVG)	**0.3**	**1.0**	**1.1**	**2.4**	**4.6**	**7.1**	**9.2**	**23.9**	**45.7**	**71.9**	**91.7**

3. Graphs


BUILD-HEAP()
3000.0
2500.0
2000.0
1500.0
1000.0
500.0
0.0
Time (mS)
0
2000
4000
6000
8000
10000
12000
Size


DELETE-MIN-ELEMENT()
300.0
250.0
200.0
150.0
100.0
50.0
0.0
Time (mS)
0
2000
4000
6000
8000
10000
12000
Size


EXISTS()
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
Time (mS)
0
2000
4000
6000
8000
10000
12000
Size