Amory Weinzierl and James Huang

Dr. Mark Fontenot

CS 2341 - 801

9 December 2019

Search Engine Report

The AVL tree data structure is a self-balancing binary search tree, which is a node based data structure where the left subtree must be a lesser value than the parent node, the right subtree must be greater value than the parent node, and each subtree must be a binary search tree. This consistent structure makes traversing the tree and searching for an element an efficient and quick process. A primary function of the AVL tree is tree rotation, which changes the structure of the tree without changing the order of the elements.

The hash table data structure stores data through a process called hashing. Hashing is a process that converts a range of key values into a range of indexes of an array. A hash table utilizes an array as storage and uses hash technique -- which typically involves the modulo function -- to generate an index where an element is to be inserted or is to be located from.

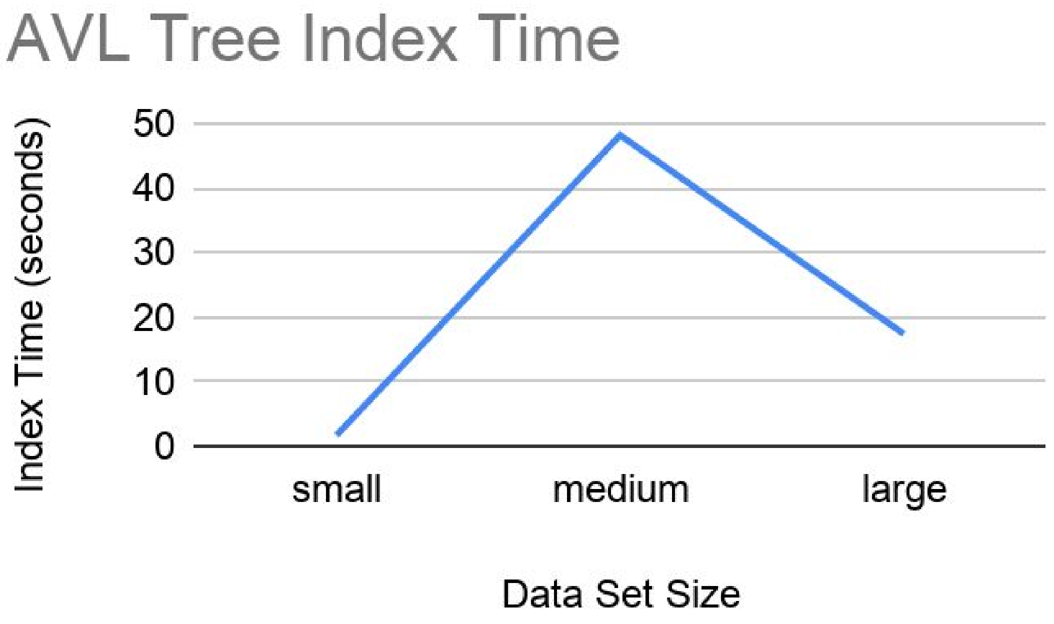
In terms of each data structures' time complexity, an AVL tree has the time complexity of O(logn) while a hash table has the time complexity of O(1) for the functions of searching, inserting, and deleting. Therefore, in terms of these selected functions, a hash table is more efficient than an AVL tree. With these three functions being of great importance for the search engine, that deems the hash table data structure as more efficient and, therefore, useful in this project. However, the data below demonstrates that, as the data set gets bigger, hash table gets less efficient, as hash tables tend toward a more linear nature as the data set increases. Our AVL tree has a bit of extreme data when it comes to the five hundred file data set, however, according to research, AVL trees tend to be more efficient and reliable when it comes to large data sets, as its searching time tends toward a more logarithmic nature, although our data does not demonstrate that. Generally, our hash table function is more reliable than our AVL function, thus, we believe that the hash table is the better data structure to utilize in a project such as this.

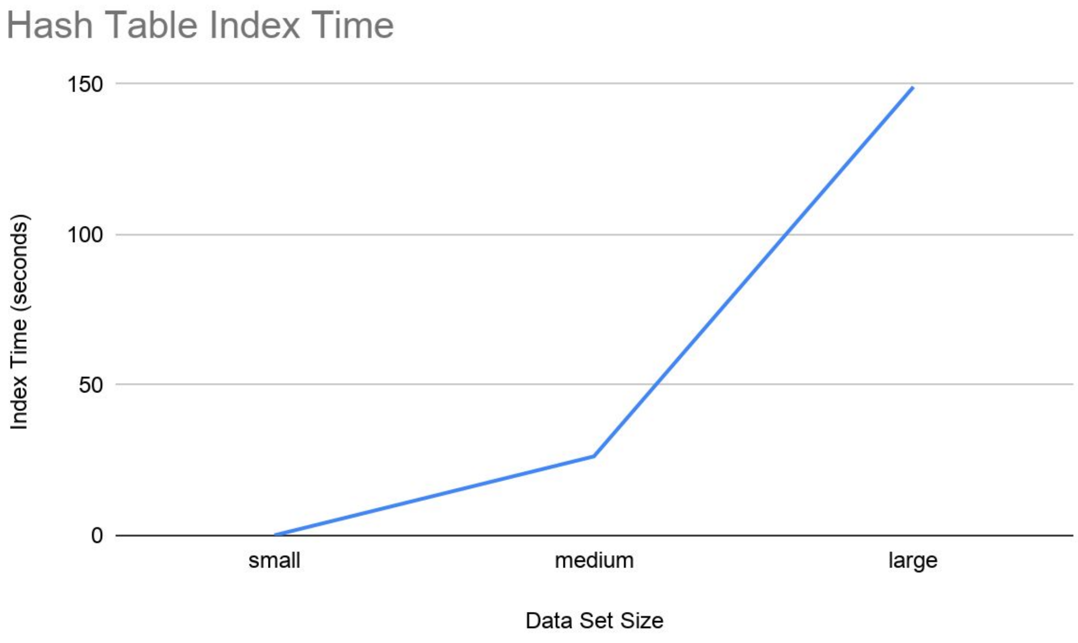
Time To Load Index For Each Data Structure Based on Data Set Size (In Seconds)

AVL Tree			Hash Table		
Small (50)	Medium (500)	Large (1000)	Small (50)	Medium (10000)	Large (31815)
2.45472	59.1101	20.1103	0.124664	27.0641	163.351
2.47125	46.4499	18.0188	0.102083	24.9682	140.798
1.53507	45.9457	17.4811	0.100676	27.1242	146.781
1.53499	47.4921	18.1813	0.087848	25.5431	153.673
1.57637	43.6665	16.3524	0.089118	26.8557	161.352
1.57603	43.7113	17.5995	0.088899	27.0897	137.452
1.59086	48.9417	16.3079	0.089295	26.9897	142.375
1.52019	49.3595	16.4591	0.102887	27.1019	143.323
1.52431	49.2693	17.0583	0.090032	25.4564	147.281
1.61578	47.8922	16.8225	0.101011	24.9678	154.732

Two-Sample T Testing

AVL Tree			Hash Table		
Small (50)	Medium (500)	Large (1000)	Small (50)	Medium (10000)	Large (31815)
1.73996	48.1838	17.4391	0.097651	26.3161	149.1118


AVL Tree Index Time
Index Time (seconds)
50
40
30
20
10
0
small
medium
large
Data Set Size

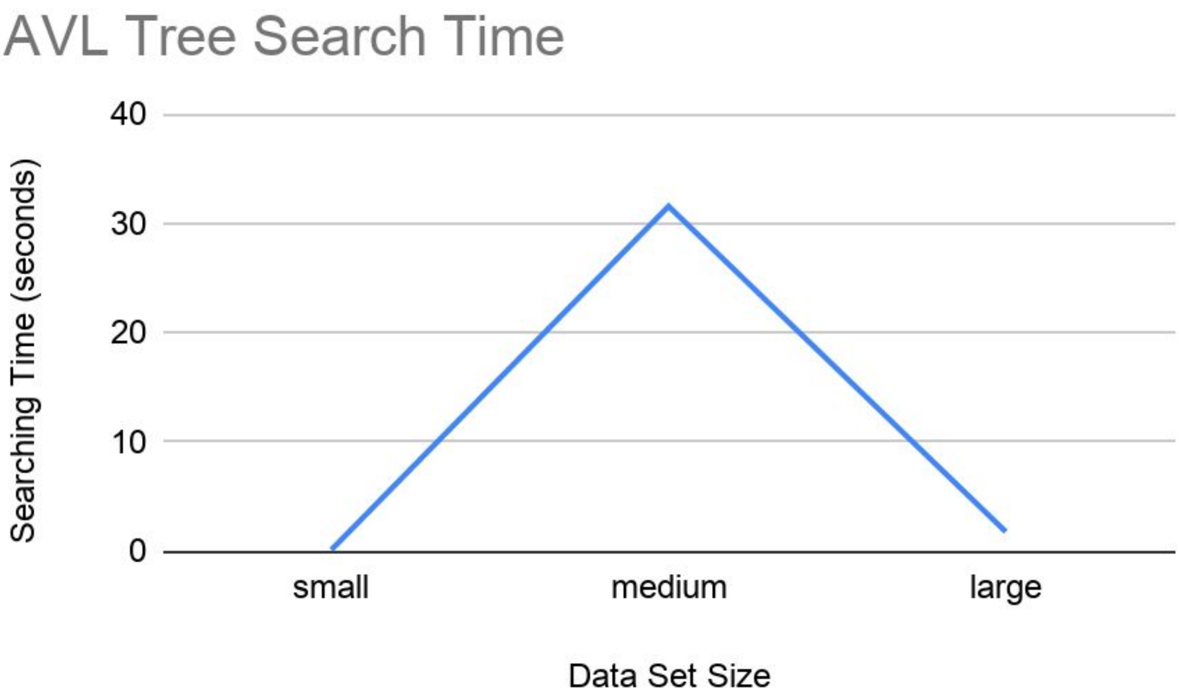

Hash Table Index Time
Index Time (seconds)
150
100
50
0
small
medium
large
Data Set Size

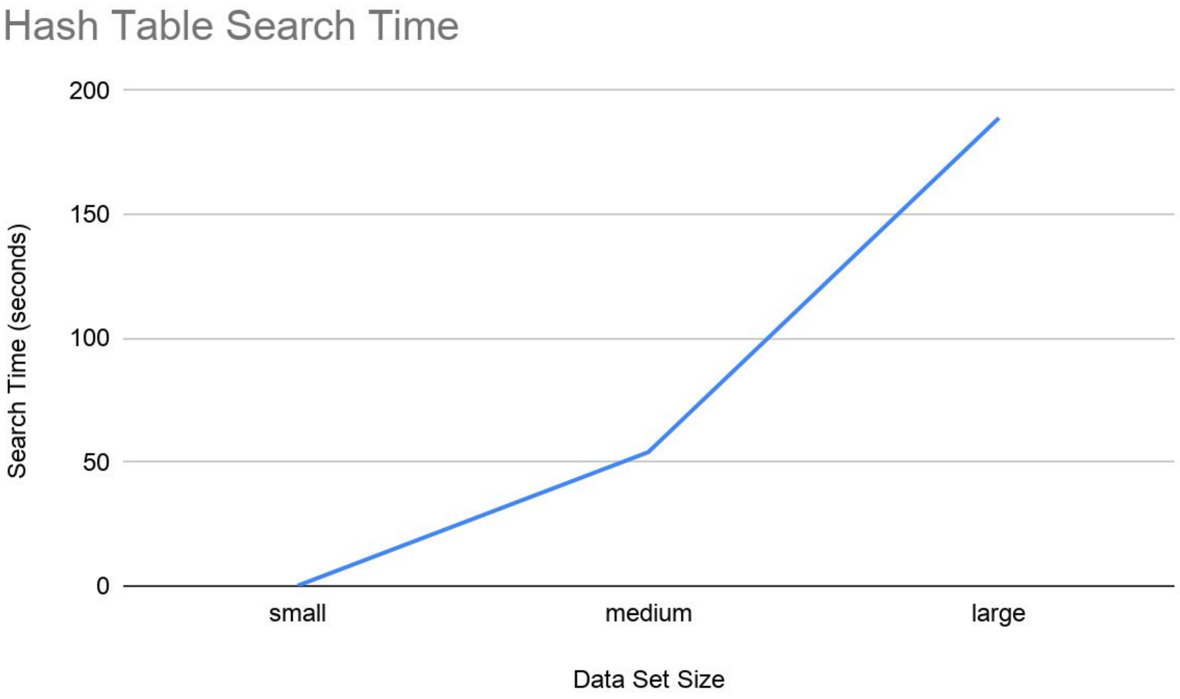
Time To Search In Each Data Structure Based on Data Set Size (In Seconds)

AVL Tree			Hash Table		
Small (50)	Medium (500)	Large (1000)	Small (50)	Medium (10000)	Large (31815)
0.09462	30.9382	1.7405	0.11432	54.8234	187.8532
0.18266	29.9306	1.7721	0.13608	53.6932	190.5232
0.12327	31.3843	1.7287	0.18797	54.4721	188.5435
0.15214	30.8612	1.9173	0.19872	54.8111	188.6345
0.10796	31.5667	1.7330	0.13729	54.5623	187.6435
0.17549	31.2345	1.8206	0.12393	52.6753	189.6356
0.13503	31.3268	1.7800	0.14861	54.7685	191.4353
0.14895	32.1035	1.7357	0.11214	53.2453	189.7865
0.15229	34.2197	1.8071	0.13932	54.6847	188.2345
0.10685	32.0841	1.8362	0.14702	52.7462	187.4324

Two-Sample T Testing

AVL Tree			Hash Table		
Small (50)	Medium (500)	Large (1000)	Small (50)	Medium (10000)	Large (31815)
0.13793	31.56496	1.78712	0.14454	54.04821	188.97222


AVL Tree Search Time
Searching Time (seconds)
40
30
20
10
0
small
medium
large
Data Set Size


Hash Table Search Time
Search Time (seconds)
200
150
100
50
0
small
medium
large
Data Set Size