The Effect of Search Engine Index Structure on Search Time

By Hayden Center and Ethan Potthoff

Introduction

Both the AVL Tree and the Hashtable data structures are used to store the same data types, but, behind the scenes, each container is structured very differently.

In an AVL Tree, data is stored in nodes with each node having a left subnode and a right subnode, creating a tree structure. All of the data element nodes to the left of a particular node have are less than that node's data value, while all the data element nodes to the right of a particular node are greater than that node's data value. Additionally, if the AVL Tree becomes out of balance, or one of the subtrees becomes too large relative to the others, the data structure rotates its nodes until the tree is balanced again. While inserting elements into the AVL Tree and rebalancing takes time, searching for elements in an AVL Tree is the data structure's specialty: If an element to find is lesser or greater than the current node, the search moves left or right respectively and halves the data set to search after each comparison.

In a Hashtable, however, each data element is stored in an array at its specific hash value. To calculate a hash value for a particular data element, the element is put through a hash function, which returns an integer, and then modded by the array size, which insures the elements hash value corresponds with one of the indices of the array. At each array index, the elements with the corresponding hash value are stored in a linked list chain. During element insertion, if a data chain at a certain hash value gets too long, the Hashtable doubles the size of the underlying array and rehashes all the elements to their new hash value in the array. This helps to minimize search time by eliminating the possibility of data elements bunching up at a particular hash value. While rehashing can be a time expensive process, searching the Hashtable is very quick: The element to find is put through the hash function and modded by the table size to get a hash value, and then the short chain at that specific hash value is searched to find the element.

Both the AVL Tree and the Hashtable have an advantage over other data structures when it comes to search time, which leads to our experiment's central question: Which data structure minimizes search time the most?

Hypothesis

Because the Hashtable uses hash values to instantly find the correct index in its' array, the worst case search of a Hashtable on larger data sets is only limited to the maximum chain length specified. The AVL Tree, however, gets larger with the growing data set, so search time increases with its size. For this reason, we believe that on larger data sets, the Hashtable will exhibit a faster search time in our search engine than the AVL Tree.

Procedure

To compare the search times of the AVL Tree against the Hashtable, 63,980 .json files containing text from past Supreme Court case rulings will be parsed and read into both an AVL Tree index and a Hashtable index using a maximum chain length of 20 words. Once all the words are successfully in the data structures, the same words will be searched and search time recorded using the standard template library chrono function. The clock starts immediately after the user enters a search, and the clock stops right as the word is found in the index. The search engine will then display the time and then print the top 15 relevant documents for the search.

Because the words are stored individually in the index only once, the frequency of a particular word in the case rulings shouldn't impact search time. As a result, we randomly chose 12 words from the 50 most frequent word list to use in our searching tests. In the experiment, each of the 12 words will be searched for 3 times with search time recorded, and the average of the 3 search times will be used for comparisons and analysis.

Data

Hashtable Search Times:

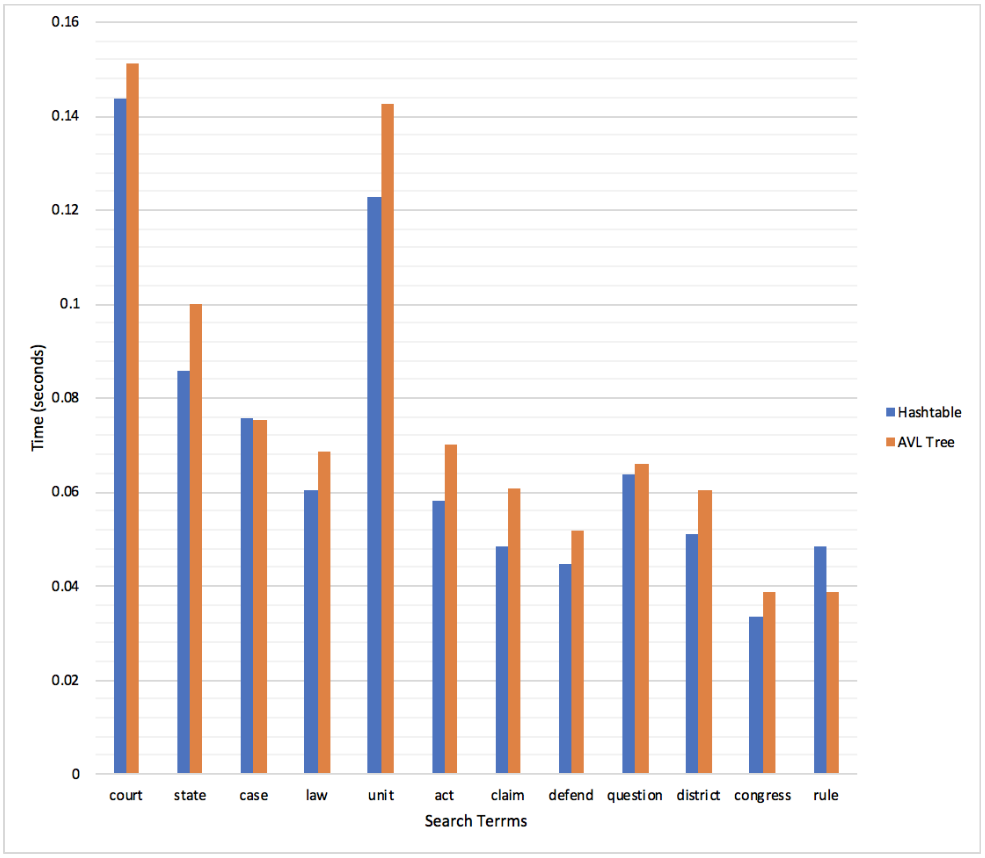
Search Word	Time 1	Time 2	Time 3	Avg. Time
court	0.146631	0.146474	0.137699	0.143601
state	0.086542	0.089555	0.08151	0.085869
case	0.073923	0.068341	0.085265	0.075843
law	0.062603	0.064515	0.054317	0.060478
unit	0.122531	0.121688	0.124592	0.122937
act	0.057799	0.060896	0.055835	0.058176
claim	0.04831	0.046051	0.051376	0.048579
defend	0.043136	0.047919	0.042652	0.044569
question	0.064636	0.060931	0.065873	0.063813
district	0.047664	0.053736	0.051789	0.051063
congress	0.035753	0.029686	0.034767	0.033402
rule	0.049158	0.049803	0.046065	0.048342

Figure 1

AVL Tree Search Times:

Search Word	Time 1	Time 2	Time 3	Avg. Time
court	0.149112	0.15208	0.151983	0.151058
state	0.108672	0.094416	0.096487	0.099858
case	0.078475	0.073352	0.074546	0.075457
law	0.075588	0.063951	0.066031	0.068523
unit	0.145766	0.148477	0.133891	0.142711
act	0.076884	0.0676	0.065306	0.06993
claim	0.061583	0.059557	0.061301	0.060813
defend	0.051494	0.056351	0.048114	0.051986
question	0.066907	0.065736	0.065028	0.065890
district	0.062308	0.058296	0.060478	0.060360
congress	0.047904	0.034707	0.033695	0.038768
rule	0.085358	0.05646	0.061298	0.038768

Figure 2



Analysis

The averages from Figure 1 and Figure 2 above are represented in the chart to the right. The y- axis represents time in seconds, while the x- axis lists the 12 search terms. The blue bars on the chart represent the Hashtable search times, while the orange bars represent the AVL Tree search times.

In 10 of the 12 cases, the AVL Tree exhibited a slightly slower search time than the Hashtable, but in the cases of ‘case’ and ‘rule’, the Hashtable resulted in the a slower time. On average, the hashtable found a result in 0.069722 seconds while the AVL Tree found a result in 0.077010 seconds. This means, based on these 12 search results, that the AVL Tree finds a search only 0.007287 seconds slower than the Hashtable on average.

Conclusion

After performing the experiment and timing the search results for these selected query terms, we reject our hypothesis and conclude that the variance of the search result times between the Hashtable and AVL Tree is too small to conclude the Hashtable is always the superior searching data structure. In our large data set, the Hashtable was often slightly faster because the time to hash a word and search through a maximum of 20 elements is quick, but, with that being said, the AVL Tree searched through a 456,841 word index just 0.007 seconds slower on average. With smaller data sets, we suspect that the AVL Tree would actually be the slightly faster than the Hashtable in searching for elements due to the shown efficiency of a binary search. With the small scale of time difference between the two data structures searching a large data set, it is safe to say that either the AVL Tree or the Hashtable are both great ways to structure data to be quickly searched for later. Either structure will return a single value out of 456,841 data elements within milliseconds.