Alexander Clark, Stephen Shoemaker

CSE 2341

Dr. Fontenot

May 6, 2019

Figure it Out Search Engine: Program Write-up

**Intro:**

In this project we designed a search engine from the ground-up comprised of two main data structures: a Hash Table and an AVL (Anders-Velsky and Landis) Tree in order to store an inverse index that can be used parse supreme court documents in order to be searched and return relevant documents based on search queries.

**Hypothesis:**

We hypothesized that our AVL Tree Index would perform faster than our Hash Table Index when performing an operation that required iteration over the whole data set, while the Hash Table will perform faster when searching a single element in the index.

**Methodology:**

After being certain that each index was performing as expected we ran our program three separate times through the AVL Tree Index, and then again through the Hash Table Index. Numerical data pertaining to the exact speeds of each index was taken through the use of the Chrono library. After recording the respective times, we then repeated the same experiment two more times, each time with a larger data set than the last. For testing both iteration and searching we created three corpuses, one with ten documents, one with 100 documents, and one with 1000 documents.

To test searching we created a vector of ten search queries and measured how long it took to access all ten. We did these three times for each dataset and saved the results. To test iteration, we measured how long it took to iterate through the index and generate a vector. We did that three times for each data set.

**Analysis**

For the AVL tree average search speed increased while the average search speed for the Hash table decreased as trials increased. As the datasets grew, the time it took to search in the AVL tree became more regular. The hash table search was faster than the AVL tree at every amount of corpus. However, iteration seems to favor the AVL tree, which was able to perform faster at generating a vector of elements at every dataset then the Hash Table.

**Conclusion**

Our hypothesis was correct in that iterating was faster on an AVL tree then when iterating over a hash table. This most likely occurs because the hash table has empty slots that are left unfilled that still need to be checked when iterated over while the AVL tree is not array based and does not require as much empty space to be reserved. The second part of our hypothesis was also correct, searching on the hash table was faster at all data sets than the AVL tree. This occurs because the Hash table lookup function runs at constant time while the AVL tree lookup runs on O(log(n)).

Based on our data, the choice between AVL trees and Hash tables depends on what operations you are going to perform more on your data set. If your going to be accessing single elements most of the time, the hash table makes more sense because of its faster accessing times for a single element. If instead, iteration over the whole dataset is more important, the AVL tree will be the faster choice.