CS 340 Project III

Hash Tables

Jordan Kramer

Algorithms and Data Structures 340

John Matta

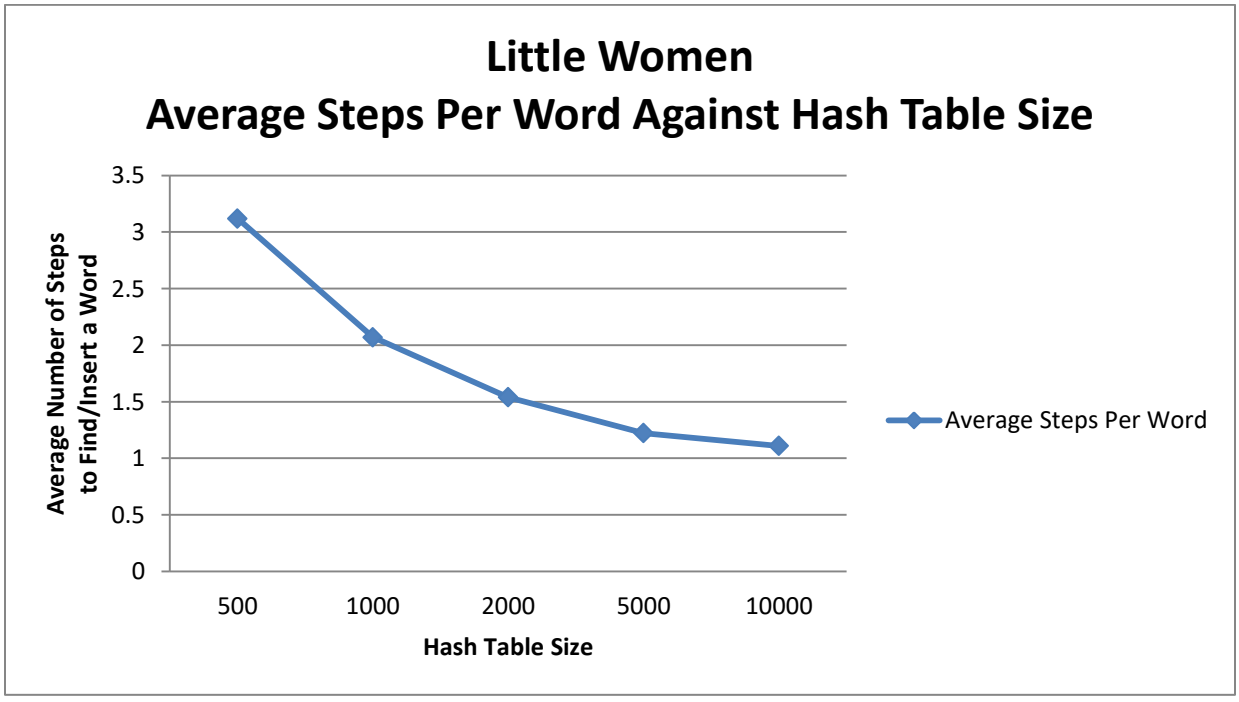
October 16th, 2018

Hash Tables are a very effective tool to use, even in today's society. In my project using hash tables, I worked with various sizes. The size of the hash tables has a direct effect on the amount of steps taken to reach a certain word in a table. Through all of my data, I found that as you have a bigger hash table, the number of steps taken to find or insert a word decreases. Looking at the *LittleWomen* file we used in our project which contains 186,609 words, the best possible amount of steps to access all of the words would be 186,609 steps. This holds true when thinking that the best possible situation would be visiting each word only one time. That means that each word would have its own specific spot in the hash table. In my project, it took anywhere between 200,000-600,000 steps. This data is expected because when given a short book like *LittleWomen,* you can't expect words to not hash to the same value. Also, As you hash table's size decreases, it is going to increase the chance of a collision which would increase the number of steps as well.

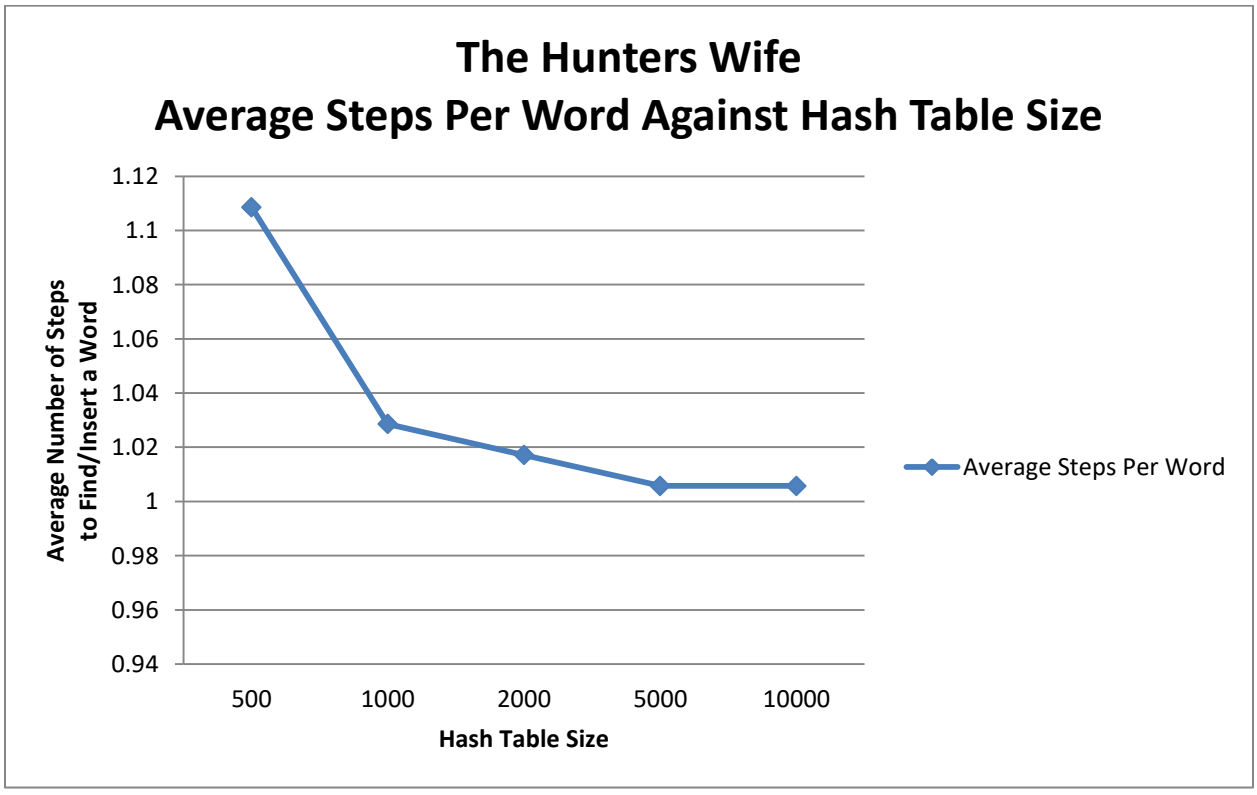
When looking at the average number of steps, my data shows an extremely better result than what was expected. When looking at the hash table of size 500, we would expect each slot to have 22 words. To get that, you have to take word count / 500 which equals about 22. The average number of steps would be 11 because 11 is half of 22. Since 11 is the average number of steps on average, we can compare this to my data which found that on average for a hash table of size 500, the average number of steps was 3.120081029. I was shocked when I found this out because that is very fast. If you use the LinkedList's add() method, this does change the expected behavior of the algorithm. If you add to the end, it will take longer to search for that word and retrieve it. Also, there is a way to determine which words were first in their list. You would need to make a loop and get(i) assuming I is your int that you are incrementing.

For my personal experiment, I wanted to test and see the difference between two different stories. One was a much larger size, and one being a much smaller file. In the larger file called *LittleWomen*, the top ten most common words were: and, the, to, a, of, her, I, it, in, she. In the smaller file called *TheHuntersWife*, the top ten most common words were: the, of, a, he, and, long, it, blue, some, as. Comparing the two lists, about 6 of the ten words were the same. I included he as being the same as her because each book focused primarily on a different gender. I was quite intrigued finding out that 60 percent of the most common words overlapped. It is something that today you really do not think that much about, but people actually use quite similar vocabularies.

I decided to take the average number of steps per word for both of the two different stories. Below is the first plot that graphs the average number of steps per word for each different size hash table.



As I mentioned before, I was very surprised when I saw this data. I expected the average number of steps to be vastly larger than what my data shows above. The expected number of steps for a hash table of 500 should be 11 but above it can be clearly seen as about 3.1. Also I saw that as the hash table gets larger, the number of steps decrease and almost seems to start bottoming out around one step per word. We determined earlier that one step per word would be the best case situation so this actually does make sense. To double check my hypothesis here, I went ahead and also calculated the average number of steps per word for the shorter story and displayed my results below.



Again we see that the average number of steps begins to bottom out or approach one. I believe it is valid to conclude that if you continue to increase your hash table size, you will never go below one and you will continue to approach one.

In conclusion, if you use larger hash tables, they become extremely effective in modern society. When using a larger hash table, you limit the number of collisions and it will only require a few steps to get to the value you are looking for. Looking back at my data, this is proven because as the hash table size grew, the average number of steps per word also decreased. In real life, this is especially useful in a searching mechanism. Say you made a reservation in a hotel your full name would be stored in their system. You would walk up to the reception desk and ask to check-in. As they search for your name, they can quickly locate you just off the first few letters because of a hash table. This speeds up the process of you getting into your room which in turn will make you happier. Imagine all of the names were stored in a system where the search was a linear search. People with the name Adam Adams might be really happy, but otherwise anyone in the back of the alphabet would be quite upset to have to sit and wait until their name was found. The moral of this story is that hash tables are very useful in today's society and they even make some people happier because they can get into their hotel room and relax sooner after a long flight.