Nick Manolov

Programming Assignment Five

"Spell Check with Binary Search Trees"

04/11/2017

Nick Manolov

C202

Dr. Hettiarachchi

April 11, 2017

Spell Checker

The assignment was exactly like the previous one except we had to use a different data structure. The main idea of the project was to load a dictionary into an array of binary search trees by filtering the first character of each word by it's ASCII value. This was done by subtracting 97(ASCII value of the letter A). Using this method an alphabetical array of binary search trees was built.

The main idea of the project was to use the above mentioned library to compare the words from another file and see which words exist in the dictionary. If they did exist, that means the word is spelled correctly and the appropriate variable that kept count was updated. The tricky part of this process was filtering out the right information. The text file"oliver.txt" was full of numbers and special characters. To get around this we used a regular expression which ignored everything but words.

The average of comparisons went way down with this approach also. The binary search tree data structure is a lot better because the time complexity is O(logn) compared to LinkedList where it was O(N) because it had to go through each word in the appropriate List.

In conclusion, we filtered a huge data file to find certain information that we used to compare against a dictionary and give back results such as how many words were found, not found, how many comparisons were made, and the average of comparisons.

```
run:
There were 914054 found words in this file
Comparisons made for words found: 14950610
There were 64537 words in this file, that were not found.
Comparisons made for words not found: 741543
The average of comparisons of words found: 16
The average of comparisons of words not found:  11
BUILD SUCCESSFUL (total time: 1 second)
```