Kotah Chapman
Assignment Five
Spell Checking with Binary Search Trees
8 November 2015

Abstract

The problem to be solved was the searching and comparisons of words between two text documents, with one document being a dictionary, “random_dictionary,”with correctly spelled words and another, “oliver,” with potential incorrectly spelled words, for number of potential correct and incorrect words in the second document along with the number of comparisons it took to find either and their averages. The solution used here was to store each word alphabetically from the dictionary in to one of twenty six binary search trees that were order alphabetically from a to z. Then words from the second document were taken one by one and searched through the appropriate binary search tree based on their first letter. The searching compared each word to a node of the binary search tree until it either was found, which increased a counter for words found, or it was not found, which increased a counter for words not found. The final numbers for this program were as follows: number of words found 937492, number of words not found 54648, number of comparisons done for those found 15094634, number of comparisons done for those not found 798838, average comparisons for those found 16, and average comparisons for those not found 14. When comparing these numbers to those of the programs of “Spell Checking with Recursive Binary Search” and “Spell Checking with Linked Lists,” it can be observed that the binary search trees and recursive binary search have low numbers for average number of comparisons for both found and not found compared to the large number for linked lists. It can also be observed that this has less instructions required for not finding words versus the other two which had the opposite, less instructions for finding words. This is because of linked lists have to go through the entirety of the specific letter linked list to find if a word is not there. The binary search tree only has to go through specific sub trees of the specific letter to find if it is not there.

Outputs

```
run:
Average Comparisons Found = 16 Comparisons for words found = 15094634 Number of
words found = 937492
Average Comparisons Not Found = 14 Comparisons for words not found = 798838 Number
of words found = 54648
```

BUILD SUCCESSFUL (total time: 10 seconds)