Cody Main

Programming Assignment 4

Spell-Checker Problem

October 25, 2016

Analysis:

In this particular programming assignment, the objective is to create a Spell-Checker Program out of Linked Lists that will read in a dictionary file, read in a large text file, compare whether each word in the large text file matches a word in the dictionary and display counters concerning the results. These counters count the total words found, the total words not found, the total comparisons in each node taken to reach the total words found, and the total comparisons in each node taken in to reach the total words not found.

In order to successfully complete this assignment, a Linked List of 26 nodes representing each of the letters of the alphabet must be created. These nodes will store all the words of the dictionary according to the first letter of each word in alphabetical order. Then comparisons of each of the words of the large file will be read in. If a word in the file matches a word in the dictionary, the counters for the total words found and the total comparisons taken to reach total words found will be incremented. Otherwise, the other two counters that count the opposite situation will increment. However the averages for these two comparison counters are actually displayed. In order to achieve the averages, we take the comparisons taken to reach the total words found and divide it by the total words found. We do the same pattern for the opposite counters.

Design:

The design of the Linked List is displayed below…

[0] -----[aardvark |] ---- [anteater |] ----- [apple |]…..
[1] -----[baby |] ---- [bugel |] ------……
[2] -----[cabbage |] ---- ……
[3] --------……..
[4] -------…….
[5]
|
|
|
|
|
|
|
|
[24] -----…..
[25] ----- [zed |] ----- [zygote |] -----…..

Every node in this Linked List contains a set of nodes that contain all the letters of a dictionary in alphabetical order according to the very first letter of each word. This list of nodes is not sorted and is grouped only according to the first letter of each word.

Once the large file is read in, it is parsed line by line and split by the whitespace, essentially making each line an array of strings. Then it is checked to see if each letter in each string is a letter. If it is a letter, it is appended using StringBuilder. Otherwise, it is not appended.

Then the string is checked to see if it can be found in the dictionary according to its very first character. If it is not found in the section pertaining to its first character, then the string is not a word. The counters increment depending on the situation.

Observations:

After implementing all of the coding for this assignment, there are two observances that can be noted in the output…

1) The words found do not include the words with spelling errors. This means the program has exhaustively checked and compared all the words in the dictionary with the word being checked according to its first letter, and the all the ones that have matches are counted as words found. This means that the program has performed accurately.

2) The both the average comparisons taken to reach the words found is lower than the words found. This is because we are only displaying the average comparisons. In reality the total comparisons taken to reach the total words found is of much greater value because it happened more often. The same goes for the total comparisons taken to reach the total words not found. However, because we are taking the averages, these are far lower.

Output:

The output has been displayed according to the assignment specifications using the two files that were given for each run…

```
run:


run:


==============================================

Counter Testing...

==============================================
```

Number of Words Found: 914045.0

Number of Words Not Found: 64546.0

Average Comparisons Found: 3554.062306560399

Average Comparisons Not Found: 7426.771496297214

==========

BUILD SUCCESSFUL (total time: 22 seconds)