Donnie Sandlin

Assignment 5

11/8/2015

Problem Statement:

In this problem it is required that a spell check program is made to count average number of comparisons for words found and words using a binary search tree.

Methodology:

This algorithm requires a few steps to run properly and that is reading in the dictionary and oliver file, parsing the file for the oliver file, Creating the variable and recursive variable, and we are making the dictionary into arrays of words beginning with the same letter . We are also creating a couple of other files to help this program correctly.

First begins with creating the global variables to be used throughout the the Assignment3 file. We also create the comparison counters, counter for words, and recursive variable is required. After that the scanners are read in for the oliver and dictionary file, but a string parser is written for the oliver file. Both files are compared using the search method in the BinarySearchTree. We add a ToString method at the end to print off the desired statements needed in the program.. This is the end of the main program that we need to the main part of the program.

Second file on this assignment is BinarySearchTree. This file contains all the methods we need to so we can get the program to function properly. Some of the methods that this file contains that we will, but the only one we will use is the search method. This will be used to compare the dictionary and the oliver file. There are other methods in this file but they will not be used, because they do not help us in this program.

AbstractTree implements Tree so this will take effect throughout the program. This is also a file that is needed to make users able to add things to lists and also even get the size of lists. This is another file that doesn’t seem like much, but is beneficial to make the program work correctly.

Another file that is in this package is a Tree file. All this file does is allows the methods in AbstractTree to work throughout the package. This file does not do much else in this program, but it is still important to have the package.

Last file that is in this package is a client file that runs the program. One thing needed in this program is a new assignment3 object. After we do that we call the method to read the file, read the dictionary, and print the average comparisons. This file is done and doesn’t need anymore work after that.

Observations and Results:

First observation was that is faster than using binary search and it is even way faster than the linked list, which surprised me. This program also has fewer comparisons as well and this is because it finds the list that has the word and checks only that one. Binary search searches all of the words which are why that one takes so long. This one does not do that which is why it is so quick when running it. Using linked lists is a little slower than Binary search trees but that is because it functions almost the same as binary search trees.

Conclusions:

This program runtimes were quick like the Linked List, but it was a lot faster. It was definitely a lot nicer than running the binary search and Linked Lists.

Output:

```
run:

Words found/correct words: 940320.0

Words Not Found/incorrect: 59221.0

total comparison: 1.5295584E7

word incorrect comparison: 568198.0

avg word found: 16.266360387953036

avg word not found: 9.594535722125597

BUILD SUCCESSFUL (total time: 4 seconds)
```