Jonathan Steininger
Programming Assignment 5
Assignment5
4/11/2017

PROBLEM STATEMENT

The problem was to create a rudimentary spell-checking program and to find the average number of comparisons it took to find words in the dictionary and the average number of comparisons it performed when it could not find words in the dictionary.

METHODOLOGY

This spell-checking program was implemented by first creating a searchable data structure, in this case a Binary Search Tree, that could store the words from a simple dictionary and then to read the text to be spell-checked, extracting only the words and not the numbers or other special characters, and then comparing those extracted words with the words in the dictionary. Whenever a word is either found or not found, values are stored for the total number of times a word is found and the total number of comparisons it took to find those words, or the total number of times a word is not found and the total number of comparisons it took to not find those words. These numbers can then be used to determine the aforementioned averages. The Binary Search Tree was chosen because it has a usable and predictable order to it, which allows it to be searched much more efficiently than a linked list because the BST search method disregards half of the tree at each step, which drastically reduces the necessary search space.

OBSERVATIONS

The results of this program are rather interesting. Because of the searchable order of the tree the process for not finding a word in the dictionary for a word from the Oliver text does not mean checking against every single word in the dictionary that begins with the same letter. This is much more efficient than examining all of the elements. Also, because the words in each list are not in alphabetical order, but instead random, the average number of comparisons it takes to find a word in the list should be roughly half because the words may be found equally likely near the beginning of the list as at the end of it. The random ordering of the dictionary helps to ensure well-balanced BSTs. Nevertheless, this program is not perfect since it cannot properly deal with words containing apostrophes, specifically contractions. At first, it took words such as: well-respected, king's, can't, and didn't and then turned them into the words: well, respected, king, s, can, t, didn, and t. This is effective for determining that the words that make up “well-respected” are spelled correctly and that “king” is spelled correctly when it would otherwise mark “king's” as being incorrect, but it artificially increases the number of incorrect words by listing “s” as a word, and it also means that “didn't” is counted as incorrect twice, even though “didn't” is spelled correctly, because neither “didn” nor “t” are in the dictionary. After hard-coding exceptions for 29 of the more common contractions and also refraining from searching for the “s” at the end of a possessive, the number of words found increased by 276 (.029%) and the number of words not found decreased by 5336 (9.0%).

Results without dealing with the apostrophe issue

Words Found:	940258
Words Not Found:	59283
Comparisons Found:	15368952
Comparisons Not Found:	573063
Average Comparisons for Words Found:	16.35
Average Comparisons for Words Not Found:	9.67

Results with dealing with several aspects of the apostrophe issue

Words Found:	940534

Words Not Found:	53927
Comparisons Found:	15371603
Comparisons Not Found:	531947
Average Comparisons for Words Found:	16.34
Average Comparisons for Words Not Found:	9.86