Mason Mullins
Programming Assignment 4
Dictionary String Comparison Binary Search Trees
November 11, 2016

Abstract

A spell checker is implemented using a random dictionary file (random_dictionary.txt) and each word is read and put into a Binary Search Tree of 26 entries, one for each letter of the alphabet. Another text file is read (oliver.txt) and the first character in each word is read and then traversed to the 26 individual Binary Search Trees - matching the letter in the alphabet. Each word is read from the book and searched in the corresponding tree. A counter is put in place if the word is found or not found. Not found words include misspelled words or words not in the dictionary. There are two total counters, one if for the number of comparisons for words that were found in the dictionary, and the other is for the number of comparisons for words that were not found in the dictionary. At the end of the program, the averages are computed and displayed for the total string comparisons.

The string parser uses File Input Stream and the first character of each word in the File Input Steam is looked at to be placed into the Linked List of the alphabet. Depending on the filename, either the word is placed in the corresponding Linked List entry (if the filename is 'random_dictionary.txt' or 'oliver.txt'). The parser computer the ASCII value of the first character by converting the string at position 0 to an integer subtracted by 97. This will give a number and that is the alphabetical character falling under the 1 through 26 alphabet positions.

This compared to the Linked Lists method of this implementation is much quicker. The time complexity of the trees is O(log(n)) vs O(n) in the linked lists. However, my averages for the words found and words not found are very different. I am getting almost ten times more words found and words not found averages.