Monte Roth

PA5

11/8/15

Extended Abstract

For this assignment, a Binary Search Tree we created is used to create a functional spell checker. The files given are a dictionary to check spelling with and a book to check the spelling of. This program is mainly two parts aside from the creation of our linked list.

The program starts by reading through the randomized dictionary, it then sorts the dictionary within two steps. Step one: separate the words according to the first character within the word. In order for this to work we create our BST of 26 objects to hold each character of the alphabet. Step two: While the program reads through the dictionary, new words are added to the correct index then traverse through the BST until reaching the end and finding a place they will be assigned to. This is finished once the end of the dictionary is reached.

The second part of the program start by reading our oliver.txt file one word at a time. It implements a counter to track the number of words and comparisons that are either found or not found. It starts by looking at the index in which the word would be contained in then traverses through the tree trying to find whether or not the word is contained already. This program also implements a time method which is used to track the time the program takes to run, print our data, and calculate the data. My results from this program are within distance of results given in class.

A Binary Tree is a data structure used universally and has many applications. This can include, but not limited to, Binary Search Trees (when data is constantly entering or leaving), Heaps (used when implementing efficient priority-queues), and Syntax Trees (constructed by compliers and calculators to parse expressions.) This program, using a BST to implement a spell checker, is much more efficient than our previous assignments 2 and 4. The time taken for a linkedlist to solve our problem (misspelled words) is greater than when using a BST. The reason for so much more efficient is the time difference between the two. The average time complexity of a BST is O(log(n)) while the time complexity of a LinkedList is O(n).