Jacob Ragains

Assignment 5: Binary Search Tree Dictionary

4/15/2017

C202

This program was nearly identical to Assignment4, we were given source code for a binary search tree, a txt file for words in a dictionary, and a txt file for an ebook. In this code, we were to take an array of binary search trees, fill it, and compare it to the ebook given. Then we were to keep track of all of the words found, words not found, and searches per each of the former conditions. Similar to the previous assignment, the easiest part was filling the array of binary search trees. They were filled according to the first character of each word in the dictionary. This made the temporary dictionary that would be used to compare to the text in the ebook given.

When it came to reading the ebook, it was quite simple, go through each word one at a time, set it to lower case, and split the word from key characters (for example: periods, commas, question marks, etc.) and put them back together as one word. Then we used the search feature of the binary search tree to see if the word was found there. This way of searching was much more efficient than the linked list. On average, it took about 15 searches per each word found compared to the 3500 in the linked list, this allowed for a much faster program. Even if there were one billion words in each dictionary, it would still only take, on average, 30 searches for each word ($2^{30} \approx 1{,}000{,}000{,}000$). Unless you have a very small amount of words in each linked list, this method of searching is going to be much more efficient.