Jacob Sriraman

Binary Search problem # 8: Comparison Writeup

After implementing binary search using the three different methods in the prior problems, it was clear that using a tree data structure is a lot more efficient for the spellchecker problem than the array-based non-tree implementation. When comparing search times between the three different methods, BST and AVL had similar times, on the order of hundreds of nanoseconds, while the simple binary search was on the order of tens to hundreds of microseconds (a drastic difference). After studying the different implementations from the videos and readings provided on Moodle, this observation makes sense.

One issue with the basic BST, however, is that it couldn’t take in all 10000 dictionary words on stack memory, and since I wasn’t supposed to convert to heap memory for this assignment, I had to just insert less words, which ended up being around 1000. So, it must be noted that the execution time comparison may not be completely fair for this reason. However, without a stack overflow issue, I am still confident that it is more efficient than the basic Binary Search.

Lastly, regarding ease of implementation, I would say that binary search was easiest without a tree. The BST wasn’t too hard to implement either (despite the stack overflow issue), and the AVL was the most complicated to implement. I would say that the efficiency and organization of the AVL tree is worth the difficult implementation, though.

For sources, I only consulted the links that were provided in class, mostly being from GeeksforGeeks. The AVL tree that I implemented was strongly based on the GeeksForGeeks example, also, but I changed it a little bit and added a few extra functions to give it the functionality that was required.