Sample Program #5 (Word Sort) Testing Strategy

Since the maximum height of the tree was set at 100, I wrote a test case that made a tree of height 100 exactly. I created a "linked list" tree by reading in 101 words in sorted order (from a dictionary), and then randomly always choosing the first or last word not used to insert. I did this until all words were added once and then queried each word. I followed this up by randomly adding words from the list of 101 and querying them. This is test case wordsort_01.

For a tricky sorting case, I decided to take the first 100,000 words in the dictionary and insert them in random order, once each. Then, I queried each of these words once. For this case, after I made the data, I had to make sure that the tree height didn't exceed 100. If it did, the plan was to rerun the program until it didn't! Luckily, on my first try, none of the heights were even greater than 30, so I stuck with using this as the second test case, wordsort_02.

To test the maximum memory usage, I did the same thing as test case 2, but instead generated 100,000 unique strings of length 20, exactly. Using the same strategy as previously, the height of the tree was in the 30s, so the case was valid. This is test case wordsort_03.

For a fun, somewhat minimal case, I did 100 insertions and queries of the same word, "camel".

None of the cases generated thus far queried too many words that weren't in the tree. So the last test case involves choosing random words from the dictionary for queries instead of the ones that were already inserted. This should result in a very high percentage of -1 -1 answers. Also, this one won't keep track of previously chosen words, so words may occur more than once. So that the real tree is queried sometimes, half of the queries are for the word just inserted. The other half are for random words. This also proved to be a really fascinating case in frequencies. Out of 150,000 words to choose from 1 word was chosen 7 times, and nine words were chosen 6 times, and quite a few words had frequencies from 1 to 5, so this also proved to be a nice case for the sorting.