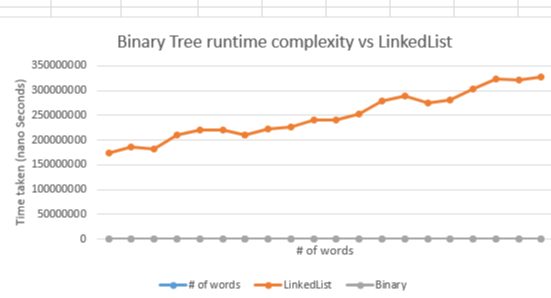
Binary Search Trees – Tries

**Purpose**

This lab was designed to teach you how to use an efficient data structure to solve a real-world problem and [benchmarking.](https://en.wikipedia.org/wiki/Benchmark_%28computing%29)

**Description (part 1)**

Finish DictionaryLL and DictionaryBST and then test with DictionaryLLTester and DictionaryBSTTester. Both tester classes should output true for all cases. Then run the DictionaryBenchmarking class(this will take a minute) to compare optimization between a linked list and a binary search tree. You should find part 1 really easy. For simplicity the classes will only store lower case words so be sure to convert all words to lower case before adding/checking. Copy and paste the data into a google spreadsheet or excel to get a graph as depicted below.



**Description (part 2)**

Finish the AutoCompleteDictionaryTrie class. You may check the accuracy against the AutoCompleteDictionaryTrieTester that ouputs all true values then the TrieGrader class should output(to a file) a text file that matches below. If done correctly you should see “All tests passed! Congratulations!”. Otherwise check the output file against the text below to find your mistake.

//TESTING ADDING WORDS (addWord, insert)//

\*\* Test #1: Adding first word to dictionary...

addWord returned true.

\*\* Test #2: Adding two more words and testing size...

Size is 3.

\*\* Test #3: Adding more words to dictionary trie (testing size after insertions)...

Dict size is 24.

\*\* Test #4: Adding duplicate word...

Adding duplicate word returned false.

\*\* Test #5: Checking size after try to add duplicate word...

Dict size is 24.

//TESTING FOR WORDS IN/OUT OF DICTIONARY (isWord)//

\*\* Test #6: Checking empty string...

Empty string in dictionary: false.

\*\* Test #7: Checking for word in dictionary...

'doggoes' in dictionary: true.

\*\* Test #8: Testing word only missing last letter...

'downhil' in dictionary: false.

\*\* Test #9: Testing word with one extra letter...

'downhille' in dictionary: false.

\*\* Test #10: Testing for more words in dictionary...

'test' in dictionary: true. 'testcases' in dictionary: true. 'testone' in dictionary: true.

\*\* Test #11: Testing word with capital letters...

'TeSt' in dictionary: true.

//TESTING AUTO COMPLETE FUNCTIONALITY (predictCompletions)//

\*\* Test #12: 3 completions requested...

Autocomplete returned the following: dog, doge, dogg,

\*\* Test #13: Testing size of list...

predictCompletions returned 3 elements.

\*\* Test #14: 6 completions requested, 0 expected...

predictCompletions found 0 words.

\*\* Test #15: 10 completions requested, 6 expected...

predictCompletions found 6 elements.

\*\* Test #16: Testing for correctness of 6 words...

Words returned by predictCompletions: dogg, doggo, doggie, doggos, doggies, doggoes,

\*\* Test #17: 7 completions requested (test for size)...

predictCompletions returned 7 elements.

\*\* Test #18: Testing if list is sorted from shortest to longest...

Check above output.

\*\* Test #19: Testing if list contains correct shorter words...

Check above output.

\*\* Test #20: Testing for remaining words...

Out of 'testone', 'testine', 'testell', and 'testing', 2 words were found.