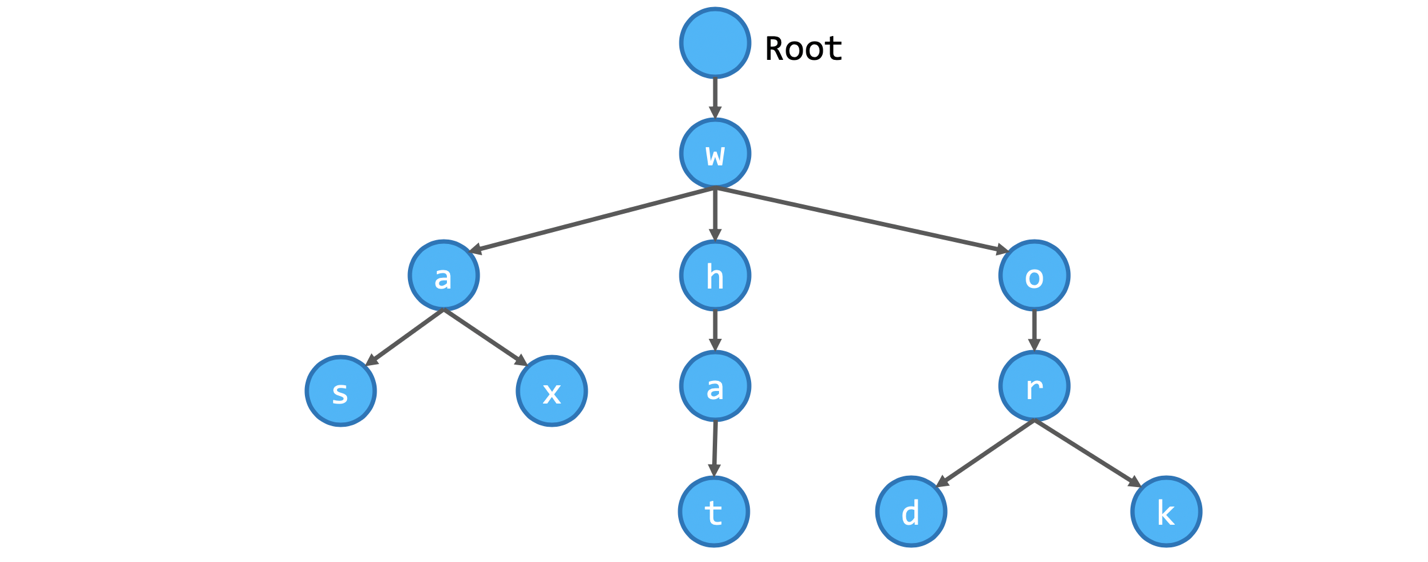
Lab 10 – Spelling Tree

Please Note: You must zip the file this time. Please use your name and student id for the folder name before you zip.

An interesting idea for storing a bunch of words is the concept of a trie (say it like try). In a trie words are stored as a general tree using a particular format:



In the above trie there are words “was”, “wax”, “what”, “word” and “work”. This appears to be a relatively efficient mechanism of storing many words because a lot of words start with the same letters. Searching to see if a word is in a trie is very easy, you start at the root and you loo, at each character and decide which branch go do down. For example, if we search for “waa” it would fail because when we reach the first a the only other options are s and x.

# Part A - Trie Implementation

Download the trie.py file from D2L and complete the functions provided. Once working, you should load in several words and test that you can detect when words are present in the trie and when they are not.

I would recommend that you append an extra letter (such as $ or #) to the end of each word. You will need to know that read and reading are both words... so you store read# and reading#.

# Part B – Spell Checker Using Trie

Complete the code found in the spellchecker.py file. This part must use the Trie from part A.

The spell checker words like this:

* Load the file words\_alpha.txt into the dictionary. The file can be obtained from <https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt> .
* Process the alice\_wonderland.txt file from Lab 2.
* Configure a timer to measure how long it takes to load the dictionary and how long it takes to process the file as separate measurements.

I would recommend using a small dictionary and a small test file before attempting to load the full dictionary as it might be quite slow.

# Part C – Spell Checker Using Python Set

Reimplement the Spell checker from Part B but instead of using the trie, use a set.