# CSCI 3333 Homework AC1:
# Implement an Autocompleter using an AVL-Tree

## 1 Introduction

A common feature of smartphone typing is *autocomplete*: after typing the beginning of a word, the user is presented with a list of possible completed words they intended to type. For instance, after typing "an", the user is presented with "and", "ant", "anagram", "anterior", etc. Your assignment is to implement autocomplete. Specifically, you will implement `Autocompleter`, a class that maintains a dictionary of words and computes the top-3 most-likely completions of any input word quickly.

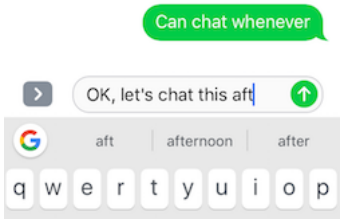


Figure 1: Autocomplete suggests "afternoon" and "after" as completions of "aft".

The dictionary is created by inserting pairs of dictionary words and their frequencies, where higher frequency means that the word is more common (e.g. "quit" has frequency 1032 and "quixotic" has frequency 15, indicating that "quit" is a more common word). The dictionary of words and their frequencies should be stored in an AVL binary search tree (see Figure 2).

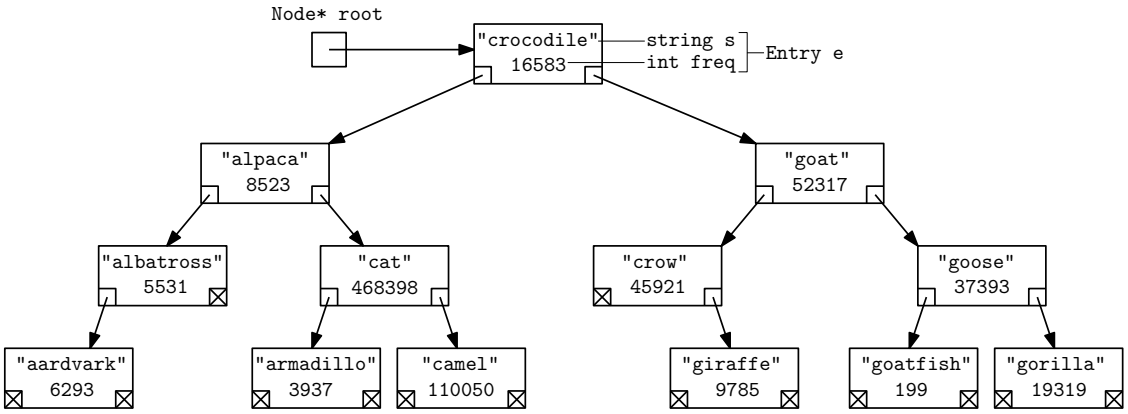


Figure 2: An AVL tree containing a collection of inserted words and their frequencies.

Notice that the set of words that start with a given string are always consecutive in sorted order. Said another way, they're all the words in a specific range (see Figure 3).

To finish this assignment you will implement the Autocompleter class by inserting a dictionary of words into an AVL-tree data structure.

## 2 Instructions

The following files have been given to you:

1. A C++ header file `autocompleter.h` declaring the Autocompleter class.

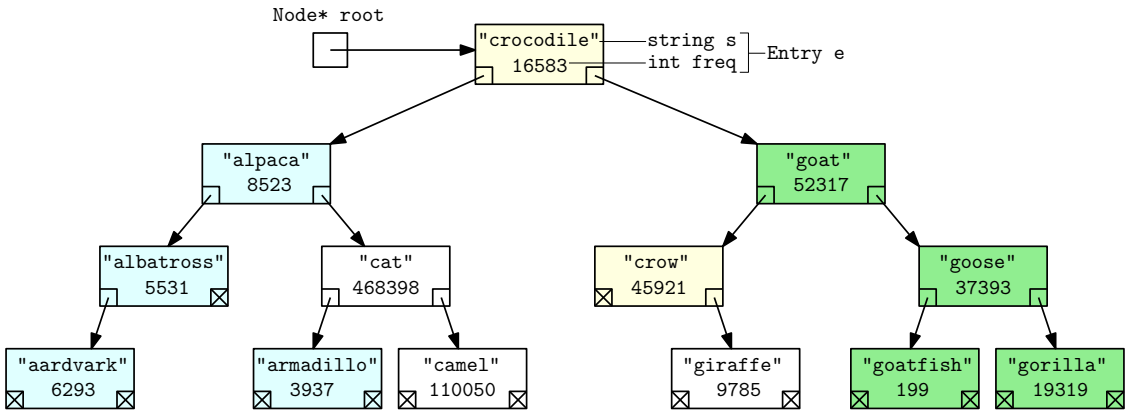


Figure 3: The words starting with `"a"` (blue), `"cr"` (yellow), and `"go"` (green).

2. A C++ source file `main.cpp` containing a `main` function with tests.
3. A text file `words2.txt` containing 293147 common English words and their frequencies.[1]

Create a new C++ source file named `autocompleter.cpp` that implements the class declared in `autocompleter.h`, so that `autocompleter.cpp` and the provided files compile into a program that runs with no failed tests, and achieves the stated asymptotic run times. Submit the source file `autocompleter.cpp` and `autocompleter.h`. If you did not modify the provided header file, you may submit just the .cpp file.

[1] Source: http://norvig.com/ngrams/count_1w.txt.