# CSCI/CMPE 3333 Homework AC1: Autocomplete

Due Tuesday, Feb 19 at 5:00 PM

## 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.

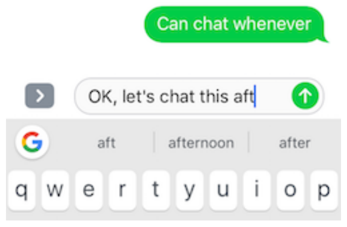

Figure 1: Autocomplete suggests “afternoon” and “after” as completions of “aft”.

The dictionary is given as a file containing a sorted list 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 a balanced binary search tree (see Figure 2).

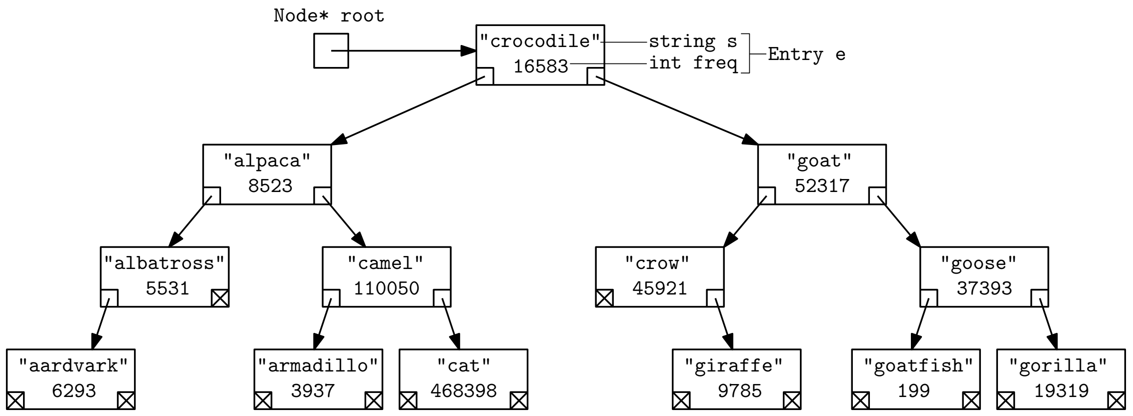

Figure 2: A balanced BST containing the words and their frequencies in the provided file `animals.txt`.

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).

## 2 Instructions

The following files have been given to you:

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

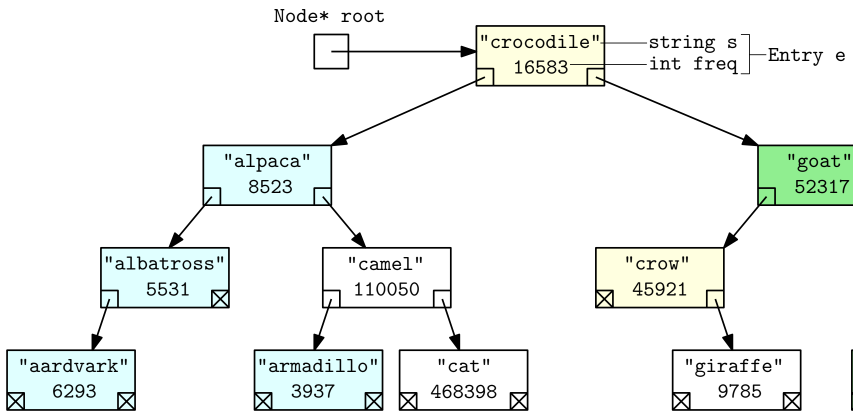

Figure 3: The words starting with `"a"` (blue), `"cr"` (yellow), and `"go"` (green).

2. A C++ source file (`main.cpp`) containing a `main` function with tests.
3. A text file (`animals.txt`) containing 13 animals names and their frequencies in sorted order.[1]
4. A text file (`words.txt`) containing 293147 common English words and their frequencies in sorted order.[2]

Download the files at `https://www.dropbox.com/sh/2inlo5cri4disqn/AACXvZE-vk5OtZAX8ODENfsMa?dl=0`. 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. Submit the source file `autocompleter.cpp`.

## 3 Submission and Grading

Submit the aforementioned source file(s) via Blackboard as attached file(s). In the case of multiple submissions, the last submission before the deadline is graded.

For grading, each submission is compiled with the provided files and run. Submissions that do not run to completion (i.e. fail to print "Assignment complete.") receive no credit. Submissions that take an unreasonable amount of time (e.g. more than a minute or so) to run and do not meet the asymptotic efficiency requirements receive no credit. All other submissions receive full credit.

See the course late work policy for information about receiving partial credit for late submissions.

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