ICS 46 Lecture B Fall 2019 Project 4: Lewis Carroll Distance

Due **November 27, 2019, 11:59 PM**
Note that this is well later than the syllabus originally stated.

Introduction

In 1879, Lewis Carroll proposed the following puzzle to the readers of *Vanity Fair*: transform one English word into another by going through a series of intermediate English words, where each word in the sequence differs from the next by only one substitution. To transform *head* to *tail*, one can use four intermediates: head → heal → teal → tell → tall → tail. We refer to the smallest number of substitutions necessary to transform one word to another as the *Lewis Carroll* distance between the two words.

Getting Started

Before you begin work on this project, there are a couple of chores you'll need to complete on your ICS 46 VM to get it set up to proceed.

Refreshing your ICS 46 VM environment
Even if you previously downloaded your ICS 46 VM, you will probably need to refresh its environment before proceeding with this project. Log into your VM and issue the command ics46 version to see what version of the ICS 46 environment you currently have stored on your VM. Note, in particular, the timestamp; if you see a version with a timestamp older than the one listed below, you'll want to refresh your environment by running the command ics46 refresh to download the latest one before you proceed with this project.

If you're unable to get outgoing network access to work on the ICS 46 VM — something that afflicts a handful of students each quarter — then the ics46 refresh command won't work, but an alternative approach is to download the latest environment from the link below, then to upload the file on to your ICS 46 VM using SCP. (See the Project #0 write-up for more details on using SCP.) Once the file is on your VM, you can run the command **ics46b refresh_local NAME_OF_ENVIRONMENT_FILE**, replacing **NAME_OF_ENVIRONMENT_FILE** with the name of the file you uploaded; note that you'd need to be in the same directory where the file is when you run the command.

The file is linked from the "public" ICS 46 page; click this link and enjoy the amazing web design skill that put it together: https://www.ics.uci.edu/~mikes/ics46/

Creating your project directory on your ICS 46 VM

A project template has been created specifically for this project, containing a similar structure to the basic template you saw in Project #0.

Decide on a name for your project directory, then issue the command **ics46b start YOUR_CHOSEN_PROJECT_NAME project4** to create your new project directory using the project3 template. (For example, if you wanted to call your project directory proj3, you would issue the command `ics46 start proj4 project4` to create it.) Now you're ready to proceed!

Choosing a project partner

You *have the **option*** to work with a second person for this assignment. If you do so, I expect you to work via pair programming. That is, you may **not** split the assignment, such as by having one person implement the Wordset while the other person implements the function that does the substitutions, and the two are stitched together later. I reserve the right to ask one or both project partners about the implementation and adjust the score accordingly.

Similarly, any academic dishonesty arising from a group will be treated as an offense by both partners.

Information about registering partnerships and submitting partnered assignments will be released mid-week. Keep an eye on Piazza if you are going this route. At time of this document's publication, your professor isn't 100% sure about one item.

Reviewing related material

I encourage you to read your textbook; in the second edition, section 9.2 covers Hash Tables, and 13.3.5 discussed Breadth-first search.

Requirements

- Fill in WordSet.cpp
    - Your implementation must be done via a hash table, as described in lecture.
    - Your collision resolution policy must be quadratic probing.
    - You must use a dynamically-allocated C++ array, not a std::vector.
    - Your implementation must fit the interface given
    - Your implementation **does not need to be templated** -- nor should it be, for the purposes of this assignment.
        - In fact, doing so will cause an issue for some of our provided tests.
    - You do need to implement the destructor.
    - Do not hard code your table for the uses we'll have in the next section.
    - You must also fill in the desired hash function: treating words as numeric types.
        - Be careful about when you take the modulus within the hash function.
        - Be careful about when you take the modulus outside the hash function.

- Write function `std::string convert(std::string s1, std::string s2, const WordSet & words)` in `proj4.cpp`
    - This function will return the conversion between s1 and s2, accordingly to the lowest *Lewis Carroll* distance. Separate consecutive words with [space]-->[space]. If there are two or more equally least Lewis Carroll distance ways to convert between the two words, you may return any of them.
    - It is recommended that you compute the distance in the following fashion:

```
Start an initially empty queue that holds strings.
Place s1 into it.
While that queue is not empty
    Consider the first word in the queue.
    Remove that word from the queue.
    If that word is s2, produce the path and return.
    Otherwise, for every possible string that is one sub from this one
        If that's a word we haven't examined before,
            Make a note of which word led us directly here.
            Put this new word into the queue.
If this finishes, there wasn't a path in the first place. Do whatever you
want here; I do not plan to give test cases where a path does not exist.
```

    - A good thing to do the first time you see a word in the previous part is to place it into a map<string, string>, where the key is the word you just saw and the value is the word *that you saw immediately before it*. This will allow you to later produce the path: you know the last word, and you know the prior word for each word in the path except the first.Furthermore, if the key isn't in that map, this tells you that you haven't seen it before.

    - Your implementation does not have to be the most efficient thing ever, but it cannot be “too slow.” In general, any test case that takes over a minute on the grader’s computer may be deemed a wrong answer, even if it will later return a correct one.

You **may not** use parts of the C++ standard template library in this assignment in implementing your Wordset. You may use std::queue, std::map, std::stack in proj4.cpp, but you *may not* use them in place of your WordSet: you must use that in the appropriate places.

Deliverables

After using the gather script in your project directory to gather up your C++ source and header files into a single **project4.tar.gz** file (as you did in previous assignments), submit that file (and only that file) to Checkmate. Refer back to Project #0 if you need instructions on how to do that.

Before you gather, be sure your program compiles using `./build` and that, after a build, you can both `./run` and `./run gtest`.

You will submit your project via Checkmate. Keep in mind that that you're responsible for submitting the version of the project that you want graded. We won't regrade a project simply because you submitted the wrong version accidentally. (It's not a bad idea to look at the contents of your tarball before submitting it; see Project #0 for instructions on how to do that.)

Can I submit after the deadline?
Yes, it is possible, subject to the late work policy for this course, which is described in the section titled Late work in the course reference and syllabus.

Grading
Your grade for this project will be two-thirds correctness: I will run some number of test cases using Google test. Different parts of this program are worth differing amounts; if you get a working hash table, but do not solve the Lewis Carroll puzzle, you can still get a large number of points. It is suggested you make several Wordset-specific test cases to ensure it is working.

The other one-third is style. ICS 46 isn’t strictly about C++, but good programming style is important. Your important variable names should have meaningful names, your code should be appropriately commented, your WordSet should be a hash table and not hard-coded to how you will be using it later in this assignment, and so on.