Assignment 1: WikiRacer Pt. 1

Assignment due on Paperless on Sunday, April 24 at 11:59 PM

Introduction and Assignment Goals

It's undeniable: human beings are obsessed with finding patterns. Whether it's in the mysteries of language, the beauties of art, or the depths of strategic games, finding patterns is built into our DNA. In fact, some biologists believe that finding patterns is what sets us apart as a species.

One interesting place to find interesting patterns is Wikipedia. For example, we can play a game called **WikiRacer**, where we try to move from one article to another with the fewest number of clicks. Try a round online before you move on!

In Assignments 1 and 2, we will be building a standard C++ program that plays WikiRacer! Specifically, it will find a path between two given Wikipedia articles in the fewest number of links. Throughout this and next assignment, we'll refer to this path as a "ladder." In the process of completing Assignments 1 and 2, you will get practice working with iterators, algorithms, templates, and special containers like a priority queue. We expect that this assignment will take you less than 30 minutes.

To simplify the implementation and allow for some testing, we will do this in two parts (A1 and A2). For Assignment 1, you will be writing the user-facing side of the program. You will have access to a function `findWikiLadder` that finds ladders between Wikipedia topics for you (you'll implement the core of this interesting search algorithm that finds the ladder in A2). You're in charge with writing parts of a main() function that will take a user-inputted file name, read in the file, find Wiki ladders between the start and end destinations in the file, and print out the ladder. Let's get started!

Aside: did you know that if you click the first link on any Wikipedia page repeatedly, you'll eventually always end up at Philosophy 97% of the time?

Download And Set Up Assignment

If you haven't already, please follow the instructions on this page before proceeding. Please do both the one-time instructions and the instructions for editing each assignment. Since we're working on assignment1, you can follow the instructions exactly. If you have any questions at all, please don't hesitate to send us an email!

Assignment Details

If you've already completed the Assignment Setup and downloaded, and set up Assignment 1, you are all set to start coding! In this assignment, you'll only be editing `main.cpp`. The assignment comes with sample files you can test your programs on in the "res" folder. We'd recommend exploring them, as well as adding your own test cases in random.txt.

1. Your first task is to **implement file reading** for main.cpp. This should take roughly 10 lines of code. We have already provided the code to read in a filename from the user. Specifically, your task is to create a filestream from the filename (check out `std::ifstream`), and then process the file data appropriately for the given program as follows.
    - The input files are formatted as follows: the first line contains the number of input pairs in the file, and each subsequent line consists of one input pair (i.e. two words, a start_page and end_page, separated by a space.) See `input-small.txt` or `input-big.txt` for an example. You can assume files will be formatted correctly.
    - For each input pair, parse out the start_page and end_page into strings. If you're using getline, you will need to do string processing on the return value of std::getline. If you're using std::ifstream, then you can use the > > operator to set the start_page and end_page strings. Next, once you have the two strings that represent the starting page and ending page on each line, then call `findWikiLadder` and append its result to the `outputLadders` vector. To see its function signature (and thus what parameters the `findWikiLadder` function takes), check out `wikiscraper.h`!
2. Your second and final task will be to print out the contents of `outputLadders`, however you may like!
3. To submit, visit paperless.stanford.edu, navigate to the appropriate assignment (assignment 1) under the CS106L class, and submit the suggested files (and ONLY the suggested files, please!).

We have implemented (and hidden) findWikiLadder for you! It will give you real Wiki ladders! Stay tuned for the next assignment, where you implement the core of this function yourself!

Implementation tips:

- To pass this assignment, please give both of the tasks above your best shot. If you're having difficulty with either (or with setting up your environment), please don't hesitate to reach out to us!
- Remember to avoid mixing `cin` and `getline`!
- Depending on how you implement reading in values, you may end up needing to convert a string to an integer. To do so, you can use the `stoi(line)` function, which takes in a `string line` and returns the integer it represents.
- In order to check your code, you can print out each input pair before it gets passed into `findWikiLinks`. The starter code already prints out filenames upon which to test your code.

For this assignment, you don't need to handle the case that the user inputs an invalid filename. (Since you are your own user in this case, make sure that the filenames you type in are valid!)

That's it! We've made this assignment a bit shorter than usual, to accommodate for setting up your environment, the surely hectic return to campus, and to give you capacity to explore the assignment code. We recommend adding your own Wiki ladders to random.txt, as well as checking out `wikiscraper.h` and

> `setup.sh`. This assignment (and class) is one that will reward you for putting more effort into learning from the materials we provide you!. Lastly, you can expect future assignments to take around as long as this one took you.
>
> And finally... Good luck! :)