Version: ~~5-4-2020: 12PM~~ 5-5-2020 9PM

CS 2223 D20 Term. Homework 4

Homework Instructions

- This homework is to be completed individually. If you have any questions as to what constitutes improper behavior, review the examples as I have posted online http://web.cs.wpi.edu/~heineman/html/teaching_/cs2223/d20/#policies.
- Due Date for this assignment is **6PM Monday May 11th** which is the day before the end of the term.
- Submit your assignments electronically using the canvas site for CS2223. Login to canvas.wpi.edu and locate HW4. You must submit a single ZIP file that contains all of your code as well as the written answers to the assignment.
- All of your Java classes must be defined in a packager USERID where USERID is your CCC user id.
- Submission information is found at the end of this document.

Primary Instructions

Note that Homework 4 is due on Monday May 11th.

For Question 1, you already have seen all of the topics you need to solve it.

For Question 2, that material will be covered on Day 26.

Version: ~~5-4-2020: 12PM~~ 5-5-2020 9PM

Q1. Word Ladders: Breadth First Search Exercise (90 pts)

A word ladder is a word game invented by Lewis Carroll. A word ladder puzzle begins with two words of the same length. To solve the puzzle, you must find a sequence of other words to link the two, in which two adjacent words in successive steps differ by one letter.

For example, given the words COLD and WARM, the following is a sample word ladder:

COLD → CORD → CARD → WARD → WARM

Note how each successive word differs by exactly one letter from the prior word. While the sequence of words can be quite long, the goal of this question is to come up with the shortest path given a dictionary of valid English words (such as found in **words.english.txt** which you can find in the repository).

Here are the assumptions you can make:

1. All words in the word ladder are just four letters long. Use an AVL<String> tree to store all these words.
2. You are to use a table `SeparateChainingHashST<String,Integer>` *table* to store the mapping from a four-letter word to an integer, representing that word's vertex in the undirected graph.
3. You are to use a table `SeparateChainingHashST<Integer,String>` *reverse* to store the reverse mapping from the integer vertex id to the four-letter word.

Copy `algs.hw4.WordLadder` into your `USERID.hw4` package and modify it to solve this problem. Here are some hints:

1. Use an `AVL<String> tree` (as implemented in `algs.hw4.AVL`) to store all four-letter English words. You can load up the "`words.english.txt`" file and add all four-letter words to this AVL tree. Because it is an AVL tree, it will self-balance as words are inserted in ascending order.
   a. As each word is added to the AVL tree, add an entry into *table* and *reverse*.
2. Next, create a graph, G, with the same number of vertices as there are words in *table*.
3. Now for the final trick, add an undirected edge (u, v) to this graph G if two words (W_u and W_v) differ by just a single letter.

A final hint: how to process all pairs of words in the AVL tree? Consider the following:

```
String min = avl.min();
for (String w1: avl.keys()) {
  for (String w2: avl.keys(min, w1)) {   // will do one more check than necessary...
    …
  }
}
```

This will allow you to process all pairs of words (W_u and W_v). Note that you need to write logic to determine if two Strings differ by just a single letter.

Version: ~~5-4-2020: 12PM~~ 5-5-2020 9PM

Q2. Finding the Longest Word Ladder Using This Dictionary (10 pts)

Given the available English words, can you find the longest Word Ladder that exists in this collection?

Property NonRepeating: You must avoid the obvious ways to have a longest word ladder, that is, repeating a pair of words over and over (like card → cord → card → cord). Thus each of the words in the word ladder must be distinct.

Property MinimalDistance: Second, given the starting and ending words – S and E – in your candidate longest word ladder, there must be no shorter word ladder that exists between these two terminal words. For example, the word ladder "card → cord → word → ward" consists of four words, but there exists a shorter word ladder from card (which is S) and ward (which is E), namely "card → ward."

Here is my one-sentence definition. "The Longest Word Ladder is formed from two words S and E such that (a) there is no shorter word ladder between S and E; and (b) no other **NonRepeating** and **MinimalDistance** word ladder exists between two different words, A and B, which is longer."

It turns out that there is one involving 17 English words (which means there are sixteen transformations):

```
abri,absi,assi,asse,arse,erse,else,elle,ells,alls,alps,amps,imps,impy,immy,is
my,isms
```

Can you find it programmatically?

Copy `algs.hw4.LongestWordLadder` into your `USERID.hw4` package and modify it to solve this problem. You will still load up an AVL tree containing the keys (as in Question 1). Now you should consider using multiple requests to `DijkstraUndirectedSP` on the graph from different source vertices.

Note: as an alternative, you could consider constructing a directed graph and use the FloydWarshall implementation (`edu.princeton.cs.algs4.FloydWarshall`) to compute the two words that require the most steps in a word ladder.

Finally, this question is only required to print out the TWO words that are at the end of the word ladder. Once these two words are printed out (together with the number of steps) you only need to run the WordLadder solution you developed in Q1 to show that you have found this word ladder.

Version: ~~5-4-2020: 12PM~~ 5-5-2020 9PM

Q3. BONUS question: Word Pyramid [1 pt]

A word pyramid is a word game. A word pyramid puzzle begins with a word of N letters. To solve the puzzle, you must find a sequence of other words, of decreasing size in length, until you reach a word with just a single letter. To produce the next word in the sequence, remove a letter and form a new word from the remaining letters

For example, given the starting word, RELAPSE, the following is a sample word pyramid:

RELAPSE → PEARLS → SPEAR → PEAS → SEA → AS → A

Note how each successive word differs by exactly one letter from the prior word (and is an anagram of the remaining letters). Each of the subsequent words must be a valid word (such as found in **`words.english.txt`** which you can find in the repository).

Here are the assumptions you can make:

1. The length of the initial word will be seven characters or less.
2. You are to use a table `SeparateChainingHashST<String, Integer>` *`table`* to store the mapping from a word to an integer, representing that word's vertex in the undirected graph.
3. You are to use a table `SeparateChainingHashST<String, Integer>` *`reverse`* to store the reverse mapping from the integer vertex id to its associated word.

This is a bonus question, so you are on your own. Feel free to create a new class based on the Q1 Word Ladder class. Here is my sample output from a run:

```
Enter word to start from (all in lower case):
relapse
relapse
serape
spear
spar
spa
pa
a
```