A5: Efficient Search in Complex Search Spaces

Assigned: Tuesday March 26, 2014
Due: Tuesday April 8, 2014, 11:59 P.M.
Type: Individual

Problem Overview

When I was a sophomore in college, I found myself in what would turn out to be the most influential course that I've ever taken. Everything about the course was in some way exceptional, but the professor particularly so. She made what could have been an ordinary English Composition class into an examination of thinking. One of the recurring themes in the class was "making connections"—the process of thought and intellect that allows us to associate things and see relationships among different things. So, in honor and memory of O.A.L., this assignment is all about making connections.

The focus of the assignment is to implement a word connection game that has been played in one variation or another for at least 130 years. The object of the game is to transform a start word into an end word of the same length by a sequence of steps, each of which consists of a one-letter change to the current word that results in another legal word. We'll call this sequence of word transformations a *Dodgson sequence* in honor of the game's inventor, Charles Lutwidge Dodsgon (Lewis Carroll).

For example, given the starting word "clash" and the final word "clown," a legal Dodgson sequence would be:

clash, flash, flask, flack, flock, clock, crock, crook, croon, crown, clown

And a Dodgson sequence from "cat" to "dog" would be:

cat, can, con, cog, dog

The game is usually played so that each player tries to find the *shortest* Dodgson sequence between two words. The shortest sequence would, of course, depend on the *lexicon*, or list of words, being used for the game. Using the SOWPODS word list (see Provided Resources), Dodgson sequences with the fewest number of steps from "clash" to "clown" and from "cat" to "dog" would be:

clash, class, claws, clows, clown

cat, cot, dot, dog

Requirements

You must complete the implementation of a Java class named `Dodgson` that provides several methods relating to Dodgson sequences. Each public method of the `Dodgson` class is discussed below. You are free to create as many private methods as you like. You are also free to import and use any class that you would like.

`Dodgson(InputStream in)`

The constructor for the `Dodgson` class builds the lexicon to be used from the provided `InputStream` object. The caller must associate one of the provided word lists (see Provided Resources) with an `InputStream` and pass that reference as the constructor parameter. The constructor must read all the words from the provided `InputStream` and store them in an appropriate collection, which will be the lexicon for this `Dodgson` object.

`List<String> getMinSequence(String start, String finish)`

This method constructs a shortest-length Dodgson sequence from start to finish. If multiple sequences exist with minimum length, no guarantee is made about which is returned. If no sequence exists from start to finish, an empty List is returned.

`boolean isValidSequence(List<String> seq)`

This method checks the provided list of strings to see if it is a valid Dodgson sequence. A list is a valid Dodgson sequence if and only if each string in the list is a valid word in the lexicon, and each pair of adjacent words has exactly one character in corresponding positions that is different (they have a *Hamming distance* of 1).

`int hammingDistance(String s1, String s2)`

This method returns the *Hamming distance* between the two given strings. The Hamming distance `d` is the number of corresponding characters in `s1` and `s2` that are different. If the two strings do not have the same length, the Hamming distance is defined to be -1. Examples:

s1	s2	d
cat	cat	0
cat	hat	1
cat	dog	3
cat	fish	-1
tiger	eagle	4
love	hate	3

`boolean isWord(String s)`

This method checks to see if the given string is a word in the current lexicon.

`int wordCount()`

This method returns the total number of words in the current lexicon.

Provided Resources

You are provided with the following resources as part of the assignment.

- A source code file, `Dodgson.java`. This is the class that you must implement and submit as your solution to the assignment.
- A jar file, `WordLists.jar`. This jar file contains the following text files that you are allowed to use as lexicons for the Dodgson class: `CSW12.txt` (the word list used in international Scrabble tournaments), `OWL.txt` (the word list used in North American Scrabble tournaments), `sowpods.txt` (SOWPODS—a commonly used combination of the CSW list and the OWL list), `words.txt` (the word list supplied in UNIX distributions), `small.txt` (a relatively small subset of the SOWPODS list), and `names.txt` (a list of first names used in the United States from 1880-2012, provided by the U.S. Social Security Administration[1]).
- A sample client, `A5.java`. This driver class illustrates basic calls to the `Dodgson` methods, and it also demonstrates how to associate a text file contained in `WordList.jar` with an `InputStream` object.

Example Shortest Dodgson Sequences

The following are example shortest Dodgson sequences using the SOWPODS lexicon. The following call could return any of the three lists for (door, lock) shown.

```
getMinSequence(“door”, “lock”)
```

```
cat --> hat
cat,hat

cat --> dog
cat,cot,dot,dog
cat,cot,cog,dog
cat,cag,dag,dog
cat,cag,cog,dog

head --> tail
head,tead,teal,taal,tail
head,tead,teal,teil,tail
head,heid,heil,teil,tail
head,heid,heil,hail,tail
head,heal,teal,taal,tail
head,heal,teal,teil,tail
head,heal,heil,teil,tail
head,heal,heil,hail,tail

door --> lock
door,loor,look,lock
door,dook,look,lock
```

[1] `http://www.ssa.gov/OACT/babynames/limits.html`

```
door,dook,dock,lock

bank --> loan
bank,lank,laik,lain,loin,loan
bank,lank,lark,larn,lorn,loan
bank,lank,lawk,lawn,lown,loan
bank,bonk,book,look,loon,loan
bank,bonk,book,boon,loon,loan
bank,bonk,bork,born,lorn,loan
bank,bonk,bouk,boun,loun,loan
bank,bark,lark,larn,lorn,loan
bank,bark,bork,born,lorn,loan
bank,bark,barn,larn,lorn,loan
bank,bark,barn,born,lorn,loan
bank,bauk,bouk,boun,loun,loan
```

Assignment Submission

You must turn in only the `Dodgson.java` file to Web-CAT no later than the published deadline. **Do NOT submit the provided WordLists.jar file.** Submissions made within the 24 hour period after the published deadline will be assessed a late penalty of 15 points. No submissions will be accepted more than 24 hours after the published deadline.