Assignment 5 – Efficient Search in Complex Search Spaces

Assigned: Thursday March 21, 2013
Due: Tuesday April 2, 2013 11:59 PM
Assignment Type: Team

Problem Description

When I was a sophomore in college, I found myself in what would turn out to be the single 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 connect things together 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 game is to transform a start word into an end word of the same length by a sequence of one-letter changes at a time, where each change produces a legal word. We'll call this sequence of word transformations a Dodgson sequence.

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

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

A shorter, and in fact the shortest, Dodson sequence from "clash" to "clown" would be:

clash, class, claws, clows, clown

You must write a Java program that generates the **shortest** Dodgson sequence for given start and end word combinations. Your program must take its input from a text file named `diller.txt`, which contains an unspecified number of start and end word pairs, one pair per line. Your program must output a Dodgson sequence with minimum length for each line of input. To qualify as a word, a character string must appear in some specified lexicon. For this assignment, you must use be able to use any of the following three word lists: `CSW12.txt` (the word list used in international Scrabble tournaments), `OWL.txt` (the word list used in North American Scrabble tournaments), or `words.txt` (the word list supplied in UNIX distributions).

Your solution must involve the following files, at a minimum.

1. A5.java – This class drives the execution of your solution. Executing the main method of this class must generate a minimum Dodgson sequence for every word pair listed in diller.txt, and send its output to stdout. The output must be a comma-separated list of strings that begins with the start word and ends with the end word (see the examples above), one sequence per line. NOTE: You must observe this output format since automatic string matching will be used to score the output of your code.

2. Dodgson.java – This class is the Dodgson sequence "generator." It must contain all the logic needed to transform one string into another string (start word, end word) while obeying the Dodgson sequence rules: (1) The start word, end word, and all intermediate words must be of the same length (contain the same number of characters). (2) A valid transformation is one in which a given word is transformed into a new word by changing only one letter. (3) To qualify as a "word" a string must be in a specified lexicon.

   The Dodgson class must have the following methods, at a minimum:
   - A parameterless constructor that initializes the class with a default lexicon file.
   - A constructor that takes a single String parameter that is the name of a lexicon file (e.g., "CSW12.txt").
   - public String getSequence(String start, String end) – This method returns a String that contains a minimum Dodgson sequence from start to end, formatted as described above.

3. Lexicon.java – This interface describes all the services that can be used to access a lexicon. This interface is provided to you and MUST NOT BE CHANGED.

4. Lex.java – This class must implement Lexicon and will provide access to a specified lexicon. You are free to add any method you like beyond the Lexicon methods. You must correctly implement all the Lexicon methods, even if they are not used by the Dodgson class.

5. CSW12.txt, OWL.txt, words.txt – These three files contain the words that compose different lexicons. Any of these three could be used to initialize Lex and Dodgson.

6. diller.txt – This file is the required input file of start- and end-word pairs (one per line, separated by blanks).

You may implement and use other interfaces and classes at your discretion.

Here are some sample minimum Dodgson sequences for a few word pairs.

```
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
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
```

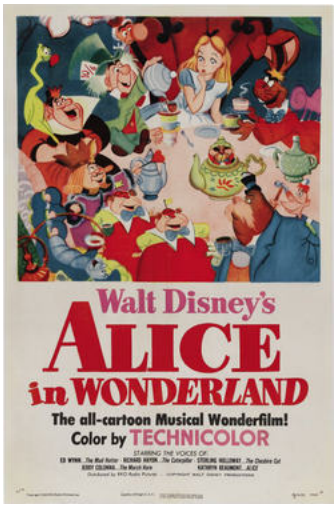
Assignment Submission

To submit your solution, you must turn in a single zip file that contains the following files: A5.java, Dodgson.java, and Lex.java. If any other files are in the zip file, they must be classes/interfaces that you have created as part of your solution. **DO NOT** turn in any of the text files or the Lexicon.java interface. You must upload this single zip file to Canvas no later than the date and time indicated. If your submission does not compile with the assignment test suite, you will receive a score of zero points for the assignment.

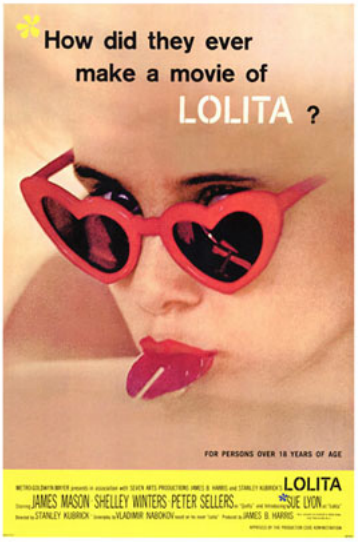
Extra Credit Challenge for the Bored

If you really want to have some fun and challenge your grey cells at the same time, you can opt to implement a second game mode in which you take a single word as input (the start word) and output a Dodgson sequence that transforms it into some *related* word that your program chooses. The relation would have to be something like synonym, antonym, or semantic association (bank – loan). To be considered for the extra points you must correctly implement the required part of the assignment. You can earn up to 15 points *on this assignment* for this game option.

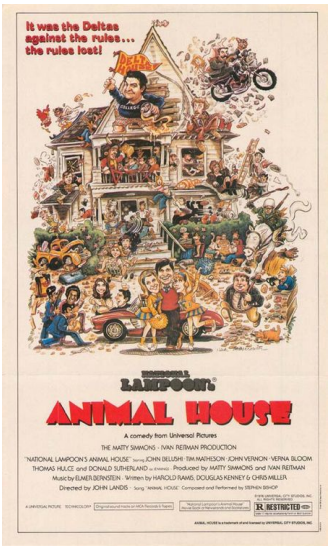
Extra Credit Team Ice Breaker



Alice in Wonderland



Lolita



Animal House

And, just for fun and maybe as a team icebreaker, the assignment carries an additional, first-come, first-served 10-point bonus *on Exam 2* in honor of my old English professor. The first team in the class to send me an email that correctly connects the content of this assignment (Dodgson sequences) to the game "Six Degrees of Kevin Bacon" will earn the points. Only connections that adhere to the following rules will be eligible for the points. (1) Intermediate steps in the connection **must** include the following three films/books: *Alice in Wonderland*, *Lolita*, and *Animal House*. The connection **must** be specified in the spirit of a "Dodgson sequence" – that is, a linear connection just like you have to connect words. This connection is not lexical. It is entirely conceptual, and you must be able to write a narrative description of the linear sequence of connections something Dodgson sequence >> Alice in Wonderland >> Lolita >> Animal House >> Six Degrees of Kevin Bacon. There may be other intermediate steps than those shown. (And just to go ahead an answer the inevitable question: There is no illegal/immoral sexual activity involved in the connections.)