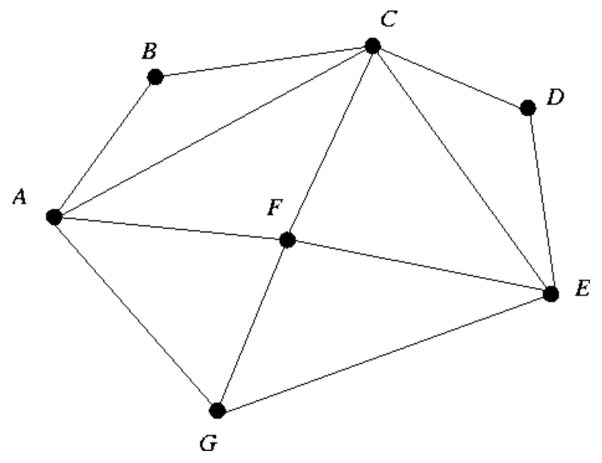
22C:16 Practice Problem Set 10

Morning Section: Complete before Tuesday, 4-22-2014

Evening Section: Complete before Monday, 4-21-2014

These practice problems are based on the program that plays the word ladders game. This is called `playLaddersGame2.py` and is posted on the course website.



Consider the network of "words" shown above. Suppose that we call the function `searchWordNetwork` on this word network with source "`A`" and target "`D`".

1. Show the contents of the `reached` dictionary and the `processed` dictionary at the beginning of each iteration of the while-loop in `searchWordNetwork`. Assume that each time we pull an element out of `reached` using `popitem()`, *we get the element that is alphabetically largest.*

2. Following up on Problem 1, show the contents of the `processed` dictionary, when it is returned from `searchWordNetwork`.

3. Solve Problem 1 again, but now assume that (i) the list of neighbors of each node is in alphabetical order and (ii) each time we pull an element out of `reached` using `popitem()`, *we get the element that was inserted earliest into `reached`.* The implication of assumption (i) is that the `for`-loops in the function that walk through neighbors will do so in alphabetical order.

4. Following up on Problem 3, show the contents of the `processed` dictionary, when it is returned from `searchWordNetwork`.

5. The program `playLaddersGame2.py` takes a pair of 5-letter words and generates a path in the word network between these (provided at least one such path exists). Now I want you to write a program `makeLaddersGame.py` that, when executed, outputs two 5-letter words along with a message of the type "I can make a word ladder 8 words long, can you beat that?" Thus, executing `makeLaddersGame.py` produces a puzzle that the user can then solve along with an estimate of how many words it took the program to connect the source and target words.

   For the program to be useful the words need to be generated at random, so that each time the user calls the program a possibly different pair of source-target words are generated. Also, the program should not generate words that are unreachable from each other. Thus the algorithm you implement should repeatedly (and randomly) choose word-pairs until it finds two words that are reachable from each other.

My final note is that to implement `makeLaddersGame.py` you should use all of the functions we implemented in `playLaddersGame2.py` and just write a new main program.
