Program 5: Word Ladders (50 points)

CSC 321 Fall 2017

Due Tuesday, December 5

A "word ladder" is a sequence of words, all of the same length, where each word after the first one differs from its predecessor by exactly one letter. For example, consider `nose`, `hose`, `host`, `hoot`, `foot`, or `fool`, `pool`, `poll`, `pole`, `pale`, `page`, `sage`. Word ladder puzzles were invented by the nineteenth century mathematician and children's book author, Charles Dodgson, better known as Lewis Carroll.

Your program needs to ask first for a file name containing a list of words, and you may assume that all the words have the same length, and are in lexical order. Some lists of words having the same length are available on Moodle. Create a graph using the words as vertices, and join two words with an edge if they differ by exactly one letter, ignoring any differences in case. Report the number of vertices and the number of edges in your graph. Then determine the number of connected components in your graph, the number of vertices in the largest connected component, and the maximal degree of any vertex in the graph, along with a word that achieves this value.

After this, ask if the user wants to determine the distance between two words (command `d`), wants to find information on a particular word (command `i`), or wants to quit (`q`). In the first case, ask for a word, and check if it occurs in the list. If so, report its degree, its neighborhood (i.e., its adjacent words), and its eccentricity within its component. In the second case, ask for the source and destination words, check that they appear in the graph, and then determine a word ladder of shortest possible length connecting the source to the destination, and state the distance as well, or report that no such path exists. Then ask for another command, and keep going until you receive a quit command.

You need to implement a graph class for representing the network of words. The client should maintain the list of words; in the graph, label each vertex with an integer, which the client could use to recover the word quickly. (If you wish you can include in your graph class a const reference to the list of words maintained in the client.) Use adjacency lists in your graph to encode the edges. The methods in the interface should include one for the degree of a vertex, one for determining the neighborhood of a vertex, one for computing the eccentricity of a vertex, one for finding a shortest path between two given words (this should stop as soon as it finds such a path), and some for constructing the graph, like a method for adding an edge. You also need some client code to manage the word ladder application. It should request all the input, read the file of words, create the graph (determining how the words are connected), and check that each word given by the user is in the graph. The word check should be a $O(\log n)$ operation, if there are n words in the graph.

Your code must also employ classes in the C++ Standard Template Library (STL). The `vector` class is very helpful, since you cannot know how many words will be in the input file. Other useful STL classes include `queue`, `set`, `list`, `map`, and `stack`. Don't use ordinary arrays, or your own implementations of lists, queues, etc.

Turn in the interface and implementation files for your graph, `igraph.h` and `igraph.cpp`, and your client file `wordladder.cpp` in Moodle. A Makefile is helpful too if you created that. A sample session appears on the reverse.

Sample Session

```
Enter file name with the words: word5.txt

  Number of vertices: xxx.
  Number of edges: xxx.
  Number of components: xxx.
  Largest component: xxx words.
  Largest degree: xxx at xxx.

Command (d/i/q)? i

Enter source word (5 letters): graph

  Degree of graph is 2.
  Neighborhood of graph:
    grape
    grapy
  Eccentricity of graph: 24.

Command (d/i/q)? d

Enter source word: Texas

Enter destination word: Maine

  The path from Texas to Maine has 13 steps:
    Texas
    Texan
    Teman
    beman
    bemat
    berat
    beray
    berry
    barry
    harry
    hairy
    haire
    haine
    Maine

Command (d/i/q)? d

Enter source word: Texas

Enter destination word: Idaho
There is no path from Texas to Idaho.
```

```
Command (d/i/q)? q

Thank you for playing!
```