CSCI 531: Artificial Intelligence

Program #2: Word Chains

Due Wednesday, September 26, 2012 at Midnight

Word Chains

For this proramming assignment, you will use uninformed search methods Breadth-First Search (BFS) and Depth-First Search (DFS) to create word chains. A word chain is a sequence of valid words that differ from adjacent words on the chain by a single character. Given a start word and final word, the goal is to find a sequence, or chain, of words that link the first word to the final word. To reduce the computational overhead, we will work with four-letter words. For example, given the words tick and flea, we might generate the chain:

- tick
- sick
- sock
- soak
- soap
- slap
- slaw
- slew
- flew
- flea

This is nine steps (not counting the starting word). Answers are not necessarily unique. The chain tick, pick, peck, peak, peat, feat, flat, flaw flew, flea is also a valid chain of length 9.

Valid chains can vary in length. For example, in the first chain, we may have had tick, sick, lick, lock, sock ... which would give a chain of length 11.

This assignment has two parts. Part 1 is to use BFS to generate word chains. These chains should be the shortest possible with respect to the dictionary used. The second part, **for the graduate students only** is to also generate word chains with DFS.

Inputs

There are three inputs:

- A small list of four-letter words should be used as the dictionary. The words are listed, one per line, in alphabetical order, all lowercase.
- The user should be prompted for the *start* word.
  You may assume it will be proper length (4 characters) and lowercase.
- The user should be prompted for the *final* word.
  You may assume it will be proper length (4 characters) and lowercase.

Processing

Read in the contents of the dictionary file. You can store the strings in an array and binary search for words as necessary, or use, e.g., a Java Vector and rely on the `contains` method. Use the basic `Graph-Search` algorithm (Figure 3.9 in 3rd edition of the text). Of course, it will have to be adapted.

You will need a search tree node. The book lists four components (State, Parent, Action and Path-Cost). In this case, you will only need the "state", which is just the current word, the depth/path-cost, and a reference to the parent node in the search tree.

For efficiency purposes some form of Hashtable should be used to avoid "repeated states" (repeated words).

In this case, "expanding" a node can be just go through each position of the current string, try each character 'a' to 'z' in the position. If the word is in the dictionary and has not already been used (check hash), then create a tree node, attach it to the search tree and put a reference to the node on the "open list" (according to the search strategy).

For BFS, you will need to implement the open list as a queue. A simple solution is to create a queue that just contains references to the nodes in your search tree. There is no need to actually duplicate information in the nodes.

For DFS, it is okay to use a stack to manage the open list.

Outputs

For BFS, output the length of the optimal chain (depth of the goal node), a word chain of that length, *in order from start word to final word.* Also, count and output the number of entries added to the queue, and the number of extractions from the queue.

Graduate students only: For DFS, output a word chain (probably won't be optimal length), *in order from start word to final word.* Indicate the length of the chain, the number of entries added to the stack, and the number of extractions from the stack.

Miscellaneous

- You may write your program in Java or C/C++.
- Take advantage of the features of the language you use.
- Each student should work alone and is bound by the honor code.
- Read the dictionary in only once, but allow the user to enter a series of start and finish words, maybe ending in the some terminal start word, e.g. EXIT.
- Some sample inputs and *possible* results for BFS. You should get the same solution length, but the queue additions and extractions may differ. The word chains are not shown here.
    - Start: test Finish: test
      Solution length 0; 1 added; 1 removed.
    - Start: test Finish: case
      Solution length 4; 1039 added; 550 removed.
    - Start: foot Finish: ball
      Solution length 4; 1359 added; 813 removed.
    - Start: head Finish: tail
      Solution length 5; 1348 added; 724 removed.
    - Start: pink Finish: blue
      Solution length 8; 1835 added; 1570 removed.
    - Start: icon Finish: etch
      Solution length 11; 1891 added; 1871 removed.
    - Start: icky Finish: okra
      No solution; 2 added; 2 removed.
    - Start: exam Finish: quiz
      No solution; 1908 added; 1908 removed.
- Submit your source code (only) to dwilkins@cs.olemiss.edu.

Send comments or suggestions to dwilkins@cs.olemiss.edu

Last modified: Tue Sep 11 20:08:19 CDT 2012