EE422C Project 3 (Word Ladder) Test Plan

Akaash Chikarmane (avc536)
Nicholas Sutanto (nds729)
Spring 2017

Test Plan Summary

Our goal for this test plan was to ensure that our word ladder program gives out the correct output for as much possible user inputs as possible for both the Breadth First Search (BFS) and Depth First Search (DFS) implementations. We did this by running 5 test cases using JUnit with each test case testing the whole program at once with a different input. Using these tests, we were able to test if the "/quit" input resulted in quitting the program, and if user inputs ranging from those with only a ladder of one-rung, to those where no word ladder exists. However, these test cases do not test individual methods or modules in the program. Thus, if the program were to fail for a certain input, we would not immediately know which method (e.g. PrintLadder, parse, getwordladderBFS) is causing the error exactly.

1. Test 1
    a. Name: testSmartMoneyDFSBFS_longDict
    b. Checks whether or not the DFS word ladder between "smart" and "money" is the same size as the BFS word ladder
    c. Setup for test: Initialization
    d. Expected output: False
    e. Pass/fail: No stack overflow, successive entries separated by one letter, both BFS and DFS successfully find a word ladder.
    f. Comments: Used JUnit with String literals as getwordladder parameters; used five_letter_words.txt as dictionary.

2. Test 2
    a. Name: testOneRungLadder
    b. Checks if both getwordladder methods work for input words "hones" and "honey", whose shortest word ladder has one rung.
    c. Setup for test: Initialization
    d. Expected ouput: BFS produces one-rung ladder, while DFS does not.
    e. Pass/fail: No stack overflow, both BFS and DFS successfully find a word ladder, BFS produces a one-rung ladder, successive ladder entries separated by one letter.
    f. Comments: Used JUnit with String literals as getwordladder parameters; used five_letter_words.txt as dictionary.

3. Test 3
    a. Name: testQuit
    b. Checks if the program quits when the user inputs "/quit".
    c. Setup for test: Initialization
    d. Expected output: Program quits.

e. Pass/fail: Program quits; does not initiate word ladder searches.
f. Comments: Did not use JUnit; test was ran manually with “/quit” as input.

4. Test 4
    a. Name: testStartEndSwitch
    b. Checks if the BFS word ladder produces word ladders of the same length when the input words are “smart money” and “money smart”.
    c. Setup for test: Initialization
    d. Expected output: BFS word ladder for both input cases are of the same length.
    e. Pass/Fail: no stack overflow, getwordladder methods successfully find a word ladder, word ladder for both input cases are the same length, successive ladder entries are separated by one letter.
    f. Comments: Used JUnit with String literals as parameters to getwordladder; used five_letter_words.txt as dictionary.

5. Test 5
    a. Name: testNoWordLadderExists
    b. Checks if both BFS and DFS program implementations do not print out a word ladder when there exists no word ladder between “cheer” and “wares” using the short_dict.txt as dictionary.
    c. Setup for test: Initialization
    d. Expected ouput: "no word ladder can be found between <start> and <end>"
    e. Pass/Fail: no stack overflow, “no word ladder can be found”; both BFS and DFS do not print out a word ladder.
    f. Comments: Did not use JUnit; replaced five_letter_words.txt in the makedictionary() method to short_dict.txt as dictionary; test was ran manually with “cheer wares” as input.