EE422C Project 3 (Word Ladder) Test Plan

Paul Cozzi pac2472
Alexander Doria aed2395
Fall 2016

Summary

We based our testing methodology on our understanding of how our code will be tested by the automatic grader. As such, we modularly tested our code using Junit, and tested code as a whole from the console to make sure it ran as specified. All methods were checked for compilation as we coded, and tests were run in JUnit and the debugger to ensure that they work. Our tests are outlined below, and we can be sure from those tests that our BFS and DFS work in most cases, our printTree works for different trees, and the parse method works with words or the /exit command. We did not test every possible combination of words, nor did we test undefined behavior (receiving two of the same word).

1.
a) DFS_Simple
b) What feature does the test cover
    Checks for correct ladder output from "ready" to "reads"
c) Set up for the test – initialization.
    Call initialize()
d) Expected output for a good module.
    Test returns an arrayList containing ready and reads and valid ladder between them as expected for output
e) The pass criterion for the test.
    An exception is not thrown, should output valid word ladder for test to pass. Ladders are verified using LadderChecker method
f) implemented in Junit

2.
a) BFS_Simple
b) What feature does the test cover
    Checks for correct ladder output of ultra short ladder ("ready" to "reads")
c) Set up for the test – initialization.
    Call initialize()
d)Expected output for a good module
    Test returns an arrayList containing only ready and reads is expected for output
e) The pass criterion for the test.
    list is only 2 words long, and an exception is not thrown, should contain correct output for test to pass
f) implemented in Junit

3.
a) BFS_Moderate
b) What feature does the test cover
Checks for a correct moderately sized ladder
c) Set up for the test – initialization.
call initialize()
d) Expected output for a good module.
The smallest ladder (9 rungs) from "Smart" to "money"
e) The pass criterion for the test.
No exception or error is thrown, word ladder is 9 rungs and valid through LadderChecker
f) Implemented in JUnit

4.
a) DFS_Moderate
b) What feature does the test cover
Checks for a correct moderately sized ladder
c) Set up for the test – initialization.
call initialize()
d) Expected output for a good module.
valid word ladder from smart to money
e) The pass criterion for the test.
No exception or error is thrown, word ladder is valid as checked by LadderChecker, contains no duplicates
f) Implemented in JUnit

5.
a) DFS_Exhaustive_nonrepeating
b) What feature does the test cover
Checks that many different word ladders have no repeats
c) Set up for the test – initialization.
call initialize()
d) Expected output for a good module.
Valid word ladders from random pairs of dictionary words when passed to DFS, test is run for a long time
e) The pass criterion for the test.
Each of the checked word ladders must be valid according to LadderChecker and have no repeating words. No exceptions or errors
f) Implemented in JUnit

6.
a) DFS_Exhaustive_noncrashing
b) What feature does the test cover
Checks that DFS does not stack overflow
c) Set up for the test – initialization.
Call initialize()
d) Expected output for a good module.
Runs for a long time when random word pairs are continuously passed to DFS

e) The pass criterion for the test.

Must not throw a stack overflow error

f) Implemented in Junit

7.

a) Ladder_Report

b) What feature does the test cover

Checks that ladders are printed properly when empty and with a ladder of reasonable size

c) Set up for the test – initialization.

build an empty ArrayList<String> and an ArrayList<String> with size 10

d) Expected output for a good module.

Prints nothing for the empty arraylist and prints the second arraylist from start to end

e) The pass criterion for the test.

Visual output is a ladder in the correct order

f) Implemented by replacing main()

8.

a) Command_Input

b) What feature does the test cover

Checks that the exit command work

c) Set up for the test – initialization.

none

d) Expected output for a good module.

Program terminates without error

e) The pass/fail criterion for the test.

Parse function must terminate without returning to caller when /quit is typed in

f) implemented in main with keyboard input in the debugger

9.

a) Command_Correct

b) What feature does the test cover

Checks that parse function works

c) Set up for the test – initialization.

none

d) Expected output for a good module.

An arraylist with the two valid words that were type in, in order

e) The pass/fail criterion for the test.

ArrayList must contain the correct two strings after parse() is called, and no errors or exceptions occur

f) implemented in main in the debugger