**EE422C Project 3 (Word Ladder) Team Plan**

Haley Alexander ha5722

Kausthub Poondi kp26753

Spring 2017

**Test plan summary What was your goal, methodology, and conclusion? 1 paragraph.**

**Example: Did you use JUNIT? Did you test modules or the whole thing all at once? What are as did you feel you covered in your final test suite? What did you not cover?**

The goal is to successfully create word ladders using BFS and DFS via pair programming. We also wanted to learn what it was like to work in a group to develop, code, and then test this project. The method was to talk about algorithms and basic tests before coding, revising, or running. We used pen and paper to visualize the implementation of the algorithm. It was easy until recursion happened, but work and test went quickly overall. We tested each module separately with Smart Money. Then we tested whole thing with different cases via debugger and terminal commands to see if different methods would return radically different results such as DNE and a n rung ladder.

Test Case Break Down:

Applies to Both BFS and DFS: 1, 2, 3, 4

Applies to BFS only: 5, 6, 7, 8

Applies to DFS only: 9, 10, 11, 12

1.

a) **Test name:** Dhole Twixt Both

b) **What feature does the test cover – 1-2 phrases or sentences:**

It covers a set of words with no word ladder

c) **Set up for the test – initialization:**

Exception Handler through debugger

d) **Expected output for a good module:**

“There does not exist a word ladder between dhole and twixt

e) **The pass/fail criterion for the test:**

It System.out.println is the same as above and finds no ladder

f) **Any comments, if any:**

Should be very quick for BFS

2.

a) **Test name:** Quit Both

b) **What feature does the test cover – 1-2 phrases or sentences:**

Tests that nothing gets printed or ran; should terminate the program

c) **Set up for the test – initialization:**

Set user input as “/quit”

d) **Expected output for a good module:**

Should not print. Should exit or terminate or go nowhere. No more input allowed

e) **The pass/fail criterion for the test:**

A blank terminal or lack of ability to enter words after “quit” is given to the program

f) **Any comments, if any:**

None

3.

a) **Test name:** Twice Both

b) **What feature does the test cover – 1-2 phrases or sentences:**

Tests output for when start and end are the same word

c) **Set up for the test – initialization:**

Set user input as “Rocks Rocks”

d) **Expected output for a good module:**

A 2 item Array List and the following output:

“A - rung ladder exist between rocks and rocks”

e) **The pass/fail criterion for the test:**

“A 1-rung ladder exist between rocks and rocks”

f) **Any comments, if any:**

Directions don’t mention if it should be 0 or 1. We are guessing it is 1 because the end word is always counted.

4.

a) **Test name:** XXXXX Both

b) **What feature does the test cover – 1-2 phrases or sentences:**

Tests to see if not real words will break the program

c) **Set up for the test – initialization:**

Set user input as “XXXXX yyyyyy”

d) **Expected output for a good module:**

[XXXXX, YYYYY]

e) **The pass/fail criterion for the test:**

“No word ladder can be found between xxxxx and yyyyyy”

f) **Any comments, if any:**

DFS and BFS should be equally fast

5.

a) **Test name:** Smart Money BFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Tests output to see if we follow instruction

c) **Set up for the test – initialization:**

Set user input as “Smart Money”

d) **Expected output for a good module:**

An n item array list

e) **The pass/fail criterion for the test:**

“A n-rung ladder exist between smart and money”

f) **Any comments, if any:**

N/A

6.

a) **Test name:** Small and quit BFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Tests the program will still quit when /quit appears anywhere in input

c) **Set up for the test – initialization:**

Set user input as “small \quit”

d) **Expected output for a good module:**

No output should be present

e) **The pass/fail criterion for the test:**

A null arraylist

f) **Any comments, if any:**

None

7.

a) **Test name:** One Word BFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Sees if program crashes if one word is inputted

c) **Set up for the test – initialization:**

Set user input as “Dhole”

d) **Expected output for a good module:**

Exception

e) **The pass/fail criterion for the test:**

Exception

f) **Any comments, if any:**

Program can assume that 2 words will be inputted, but it should not run at all.

8.

a) **Test name:** One Letter BFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Sees if program can find words that only differ by one letter

c) **Set up for the test – initialization:**

Set user input as “Bones Boned”

d) **Expected output for a good module:**

[BONES BONED]

e) **The pass/fail criterion for the test:**

There exist a 2-rung ladder between bones and boned

f) **Any comments, if any:**

Testing because a path that doesn’t exist and an input off by letter is same size.

9.

a) **Test name:** One Letter DFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Sees if program can find words that only differ by one letter

c) **Set up for the test – initialization:**

Set user input as “Bones Boned”

d) **Expected output for a good module:**

An n array ladder

e) **The pass/fail criterion for the test:**

There exist a n-rung ladder between bones and boned

f) **Any comments, if any:**

Testing because we limited backtracking, so we want to make sure program doesn’t miss something obvious

10.

a) **Test name:** Small and quit DFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Sees if program can find words that only differ by one letter

c) **Set up for the test – initialization:**

Set user input as “Bones Boned”

d) **Expected output for a good module:**

An n long array

e) **The pass/fail criterion for the test:**

There exist a n-rung ladder between bones and boned

f) **Any comments, if any:**

Testing separately to make sure recursion calls don’t don’t keep going

11.

a) **Test name:** Smart and Money DFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Sees if program can find a ladder for a ladder we know exists

c) **Set up for the test – initialization:**

Set user input as “Smart Money”

d) **Expected output for a good module:**

An n long array

e) **The pass/fail criterion for the test:**

There exist a n-rung ladder between bones and boned

f) **Any comments, if any:**

Testing separately to make sure recursion calls will return a valid n array

12.

a) **Test name:** Small DFS

b) **What feature does the test cover – 1-2 phrases or sentences:**

Makes sure the program doesn’t produce an output or stackoverflow for inputs that are not 5 letters long.

c) **Set up for the test – initialization:**

Set user input as “Small Size”

d) **Expected output for a good module:**

Nothing

e) **The pass/fail criterion for the test:**

No stack overflow

f) **Any comments, if any:**

Testing this because we don’t want to stack overflow present in the grader even if though we may assume that inputs are valid