EE422C Project 3 (Word Ladder) Test Plan

Jared Cormier jcc4969
Daniel John dcj597
Spring 2017

Test Plan Summary:
We tested the BFS and DFS modules separately to verify conditions specific to each. We tested the method implemented to help reduce the length of the DFS word ladder separately as well. We used the provided JUNIT test case to test the whole program at once and verify correct outputs and word ladders with no duplicates. In our test cases, we covered same word inputs, inputs with no word ladder, reversed inputs, and general case inputs. We did not cover cases in which the inputs are of different lengths, inputs differ by only one letter, or cases where inputs are not in the provided dictionary as the requirements PDF stated that these cases would not be tested. In conclusion, we were able to test our BFS/DFS module functionality by using select corner test cases. We found that it would be best to implement the DFS function using a helper DFS function that we can call recursively and were able to find a way to shorten the DFS ladder using a helper function that organizes and prioritizes word "neighbors" or adjacent words, based on how close they are to the end word. We feel our plan covers most cases following the requirements of the project.

1. General Case Ladder BFS
    - Checks for correct printing of general case word ladder
    - Set up for the test – initialization:
      None
    - Expected output for a good module:
      Ladder between "world" and "drive". Ladder checked for duplicate words.
    - The pass/fail criterion for the test:
      Ladder has no duplicates and is correct.
    - Comments:
      Test is expected to run in 2 seconds or less.

2. Shortest Ladder BFS
    - Checks for correct printing of one-rung word ladder
    - Set up for the test – initialization:
      None
    - Expected output for a good module:
      One-rung ladder between "jelly" and "hello". Ladder checked for duplicate words.
    - The pass/fail criterion for the test:
      Ladder has no duplicates, is correct, and is only one-rung.
    - Comments:
      Test is expected to run in 2 seconds or less.

3. Case insensitive BFS
    - Checks for correct printing of word ladder with any case input

- Set up for the test – initialization:
  None
- Expected output for a good module:
  Ladder between “StOnE” and “mOnEy”. Ladder checked for duplicate words.
- The pass/fail criterion for the test:
  Ladder has no duplicates and is correct.
- Comments:
  Test is expected to run in 2 seconds or less.

4. No Ladder BFS
   - Checks for correct output verifying no word ladder exists
   - Set up for the test – initialization:
     None
   - Expected output for a good module:
     “no word ladder can be found between ‘wordy’ and ‘xenon’.
   - The pass/fail criterion for the test:
     Ladder does not exist with words included in dictionary
   - Comments:
     Test is expected to run in 2 seconds or less.

5. Same length ladder for reversed input BFS
   - Checks that the same length word ladder is found for two inputs if inputs are reversed.
   - Set up for the test – initialization:
     None
   - Expected output for a good module:
     Both words ladder for inputs “smart money” and “money smart” contain same number of rungs. Ladder checked for duplicates.
   - The pass/fail criterion for the test:
     Ladder has no duplicates and the both ladders have same number of rungs.
   - Comments:
     Shows that BFS always finds shortest path between two points. Test is expected to run in 2 seconds or less.

6. General Case DFS
   - Checks for correct printing of word ladder
   - Set up for the test – initialization:
     Create Hash Set of DFS ladder and compare size of Hash set and DFS ladder Array List - Hash Set automatically removes duplicates. If they are same size, there are no duplicates
   - Expected output for a good module:
     Ladder between “horse” and “locks”. Ladder checked for duplicate words.
   - The pass/fail criterion for the test:

Ladder has no duplicates and is correct.

- Comments:
  Test case passes and makes sure that DFS Module works with no duplicates or "loops" in the ladder. Test is expected to run in 30 seconds or less.

7. Same word input DFS with the word "money"

- Checks for list returned with only two entries (the same word twice) when input is same
- Set up for the test – initialization:
  For this test, we used debugging print lines to print out the size of the DFS ladder and the first two entries – to ensure that there were only 2 entries in the DFS ladder.
  The two inputs were: "money money".
- Expected output for a good module:
  The expected ladder size is 2. Since both inputs are the same word, we should have the two inputs as the only elements in the ladder ArrayList.
- The pass/fail criterion for the test:
  DFS Ladder size must be 2. The first two printed out entries of the DFS ladder should be the word "money" that we inputted.
- Comments:
  Test Case passes and shows that the DFS Module correctly treats cases with the same input repeated – by having only 2 entries in the ladder and having both entries be the given input. Test is expected to run in 30 seconds or less.

8. DFS with different cases in the inputs (lowercase and uppercase)

- Checks to make sure that the DFS module ignores case.
- Set up for the test – initialization:
  Run DFS module with the inputs "smart money".
  Run DFS module with the inputs "SMarT moNEy".
- Expected output for a good module:
  The DFS Ladder should be the same for both input cases. The first input case had a 174-rung ladder with no duplicates. The second input (with the different cases) also had a 174-rung ladder. This shows that the DFS Module is case-insensitive.
- The pass/fail criterion for the test:
  Both inputs should have a 174-rung ladder with no duplicates.
- Comments:
  Test case passes and proves that DFS Module ignores case, as given in the requirements PDF. Test is expected to run in 30 seconds or less.

9. Reduced DFS ladder length

- Tries to reduce length of DFS Ladder by finding adjacent words or neighbors with more letters in common with the “end” word, and then prioritizing them in the neighbor iteration.
- Set up for the test – initialization:
  Try sets of inputs without the reducing method. Try sets of inputs with the reducing function.
- Expected output for a good module:
  We hope that the helper reducing method will decrease the number of rungs on the DFS Ladder and create a more efficient ladder and search process.
- The pass/fail criterion for the test:
  Without reducing method:

  “smart money” = 3917-rung word ladder
  “horse locks” = 4178-rung word ladder
  “hello world” = 2256-rung word ladder
  “hello beard” = 3005-rung word ladder

  With reducing method:

  “smart money” = 174-rung word ladder
  “horse locks” = 67-rung word ladder
  “hello world” = 2191-rung word ladder
  “hello beard” = 754-rung word ladder
- Comments:
  Helper reducing method reduces rung for each test case – often by thousands of rungs. Test is expected to run in 30 seconds or less.

10. DFS with No Ladder between words

- Checks to see how DFS Module reacts to two inputs with no ladder in between them
- Set up for the test – initialization:
  Find two words with no ladder in between them. Check with an online word ladder to double check there are no ladders. We found the input words “zebra” and “adopt” to have no ladders in between them.
- Expected output for a good module:
  The output should say: “no word ladder can be found between zebra and adopt.”
- The pass/fail criterion for the test:
  To pass, the output should say: “no word ladder can be found between zebra and adopt.”
- Comments:
  Test passed – this shows that the DFS Module handled this test case of no word ladder correctly. Test is expected to run in 30 seconds or less.