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

<Xiaoyong Liang> <XL5432>

<Yuankai Yue> <yy7347>

Fall 2016

Test plan summary:

Our goal of testing is to cover most of our code in the project. We used Junit to write unit tests for virtually every method we wrote. By making some of our private functions public, we were able to test them thoroughly before using them in our code. Not only did we test modules, we also write unit test to cover the top level functions. In addition, we also manually tested our code for integration. Overall, according to IntelliJ, we have a unit test coverage of 95%.

1.

1. Test name: testCountDiff
2. What feature does the test cover: Checks for correct number of different characters between two five-character String.
3. Set up for the test: None
4. Expected output for a good module: Different # of char in multiple strings.
5. The pass/fail criterion for the test: All asserts passed

2.

1. Test name: testBFSandDFSNoDuplicates
2. What feature does the test cover: Checks no duplicates produced by BFS and DFS search
3. Set up for the test: Create dictionary
4. Expected output for a good module: When put the result in a HashSet, the size of the HashSet is the same as the ArrayList, implying no duplication in ladder.
5. The pass/fail criterion for the test: All passed

3.

1. Test name: testDFS
2. What feature does the test cover: Checks the ladder produced by DFS is correct
3. Set up for the test: Create dictionary
4. Expected output for a good module: The arrayList contains start and end word, and size greater than 2 if start is not equal to end and start and end is not one character away and we know that there exists a ladder.
5. The pass/fail criterion for the test: All passed

4.

1. Test name: testBFS
2. What feature does the test cover: Checks the ladder produced by BFS is correct
3. Set up for the test: Create dictionary
4. Expected output for a good module: The arrayList contains start and end word, and size greater than 2 if start is not equal to end and start and end is not one character away and we know that there exists a ladder.
5. The pass/fail criterion for the test: All passed

5.

1. Test name: testBFSandDFS
2. What feature does the test cover: Checks if BFS finds a valid ladder given that DFS has found a valid ladder, and vice versa. Also checks BFS always finds a ladder that has less or equal words than that found by DFS, if there exists any ladder between two words.
3. Set up for the test: Create dictionary
4. Expected output for a good module: When BFS finds a ladder, DFS should also find one, and size from BFS if less than or equal to that from DFS. If BFS can’t find a ladder, DFS should not find one.
5. The pass/fail criterion for the test: All passed

**Unit Test Code**

@Before

public void setup(){

Main.initialize();

}

@Test

public void testParse() {

String input = "hello world";

Scanner scan = new Scanner(input);

ArrayList<String> expected = new ArrayList<>();

expected.add("hello");

expected.add("world");

assertEquals(expected, Main.parse(scan));

ArrayList<String> words = Main.parse(new Scanner("start money"));

assertTrue(words.get(0).equals("start")&&words.get(1).equals("money"));

words = Main.parse(new Scanner("start money"));

assertTrue(words.get(0).equals("start")&&words.get(1).equals("money"));

words = Main.parse(new Scanner("START MONEY"));

assertTrue(words.get(0).equals("start")&&words.get(1).equals("money"));

}

@Test(timeout = 30000)

public void testCountDiff() {

Assert.assertEquals(2, Main.getDifference("apple", "apppp"));

Assert.assertEquals(5, Main.getDifference("11111", "apppp"));

Assert.assertEquals(1, Main.getDifference("count", "cbunt"));

Assert.assertEquals(4, Main.getDifference("apple", "sspss"));

Assert.assertEquals(5, Main.getDifference("apple", "sssss"));

Assert.assertEquals(3, Main.getDifference("apple", "apsss"));

Assert.assertEquals(0, Main.getDifference("apple", "apple"));

}

@Test

public void testSortNeighbors() {

Assert.assertArrayEquals(new String[]{"appli", "apppp", "ppppp"}, Main.sortNeighbors("apple", new ArrayList<>(Arrays.asList(new String[]{"ppppp", "appli", "apppp"}))).toArray());

Assert.assertArrayEquals(new String[]{"appli", "apppp", "ppppp"}, Main.sortNeighbors("apple", new ArrayList<>(Arrays.asList(new String[]{"appli", "apppp", "ppppp"}))).toArray());

Assert.assertArrayEquals(new String[]{"appli", "apppp", "apiii", "azzzz", "zzzzz"}, Main.sortNeighbors("apple", new ArrayList<>(Arrays.asList(new String[]{"zzzzz", "appli", "apppp", "apiii", "azzzz"}))).toArray());

}

@Test

public void testBFSAndDFS(){

Random randomGenerator = new Random();

String[] dictString =dict.toArray(new String[dict.size()]);

for(int i=0; i<10; i++){

int i1 = randomGenerator.nextInt(dictString.length);

int i2 = randomGenerator.nextInt(dictString.length);

String word1 = dictString[i1].toLowerCase();

String word2 = dictString[i2].toLowerCase();

ArrayList<String> bfs = Main.getWordLadderBFS(word1,word2);

ArrayList<String> dfs = Main.getWordLadderDFS(word1,word2);

Assert.assertTrue((bfs.size()==0 && dfs.size()==0) || (bfs.size()!=0 && dfs.size()!=0));

Assert.assertTrue((bfs.size()==0 && dfs.size()==0) || (bfs.size() <= dfs.size()));

}

}

@Test

public void testBFS(){

assertTrue(Main.getWordLadderBFS("start", "start").size()==0);

assertTrue(Main.getWordLadderBFS("bazoo", "habit").size()==0);

ArrayList<String> stoneAtone = Main.getWordLadderBFS("stone", "atone");

Assert.assertEquals(2, stoneAtone.size());

assertTrue(Main.ladderFound);

assertTrue(stoneAtone.contains("stone"));

assertTrue(stoneAtone.contains("atone"));

ArrayList<String> startMoney = Main.getWordLadderBFS("smart", "money");

assertTrue(startMoney.contains("smart"));

assertTrue(startMoney.size()>2);

assertTrue(startMoney.contains("money"));

ArrayList<String> stoneMoney = Main.getWordLadderBFS("stone", "money");

assertTrue(stoneMoney.contains("stone"));

assertTrue(stoneMoney.size()>2);

assertTrue(stoneMoney.contains("money"));

}

@Test

public void testDFS(){

assertTrue(Main.getWordLadderDFS("start", "start").size()==0);

assertTrue(Main.getWordLadderDFS("bazoo", "habit").size()==0);

ArrayList<String> stoneAtone = Main.getWordLadderDFS("stone", "atone");

Assert.assertEquals(2, stoneAtone.size());

assertTrue(Main.ladderFound);

assertTrue(stoneAtone.contains("stone"));

assertTrue(stoneAtone.contains("atone"));

ArrayList<String> startMoney = Main.getWordLadderBFS("smart", "money");

assertTrue(startMoney.contains("smart"));

assertTrue(startMoney.size()>2);

assertTrue(startMoney.contains("money"));

ArrayList<String> stoneMoney = Main.getWordLadderBFS("stone", "money");

assertTrue(stoneMoney.contains("stone"));

assertTrue(stoneMoney.size()>2);

assertTrue(stoneMoney.contains("money"));

}

@Test(timeout = 30000)

public void testBFSNoDuplicates1() {

ArrayList<String> list = Main.getWordLadderBFS("start", "start");

Assert.assertEquals(list.size(), new HashSet(list).size());

}

@Test(timeout = 30000)

public void testBFSNoDuplicates2() {

ArrayList<String> list = Main.getWordLadderBFS("bazoo", "habit");

Assert.assertEquals(list.size(), new HashSet(list).size());

}

@Test(timeout = 30000)

public void testBFSNoDuplicates3() {

ArrayList<String> list = Main.getWordLadderBFS("smart", "money");

Assert.assertEquals(list.size(), new HashSet(list).size());

}

@Test(timeout = 30000)

public void testBFSNoDuplicates4() {

ArrayList<String> list = Main.getWordLadderBFS("stone", "money");

Assert.assertEquals(list.size(), new HashSet(list).size());

}

@Test(timeout = 30000)

public void testDFSNoDuplicates1() {

ArrayList<String> list = Main.getWordLadderDFS("start", "start");

Assert.assertEquals(list.size(), new HashSet(list).size());

}

@Test(timeout = 30000)

public void testDFSNoDuplicates2() {

ArrayList<String> list = Main.getWordLadderDFS("bazoo", "habit");

Assert.assertEquals(list.size(), new HashSet(list).size());

}

@Test(timeout = 30000)

public void testDFSNoDuplicates3() {

ArrayList<String> list = Main.getWordLadderDFS("smart", "money");

Assert.assertEquals(list.size(), new HashSet(list).size());

}

@Test(timeout = 30000)

public void testDFSNoDuplicates4() {

ArrayList<String> list = Main.getWordLadderDFS("stone", "money");

Assert.assertEquals(list.size(), new HashSet(list).size());

}