Sarah Whelan
slw96
12/4/2014
Testing for P4

The Main Testing Graph:

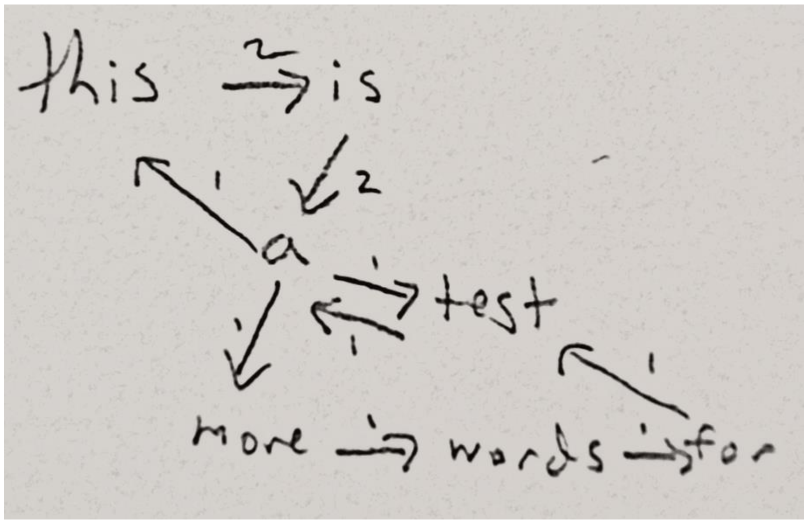

Figure 1: A graph made from a text file containing: "this is a test a this is a more words for test."

On the testing of Node and Edge classes (WordNode and WordPair):
I tested the numNodes, numEdges, inDegree, outDegree, prevWords, and nextWords of the WordGraph class in doing so I feel the node and edge classes were tested sufficiently because each of those calls the corresponding getter methods of the nodes and edges. Additionally the generatePhrase method works as expected providing further evidentiary support of the validity of the node and edge classes.

On the Word Sequence Cost Method:
After updating to the newer version of calculating word sequence cost the counts of the words are being minimized. I did run into the case where the second part (the sum) would often add zero because there were many sums that came to be the log of 1 or 0. This worked out though because it equalized the nodes with the same outEdgeCost or nodes that are equally likely.

$$\mathrm{L}(w_1,\ldots,w_n) = \log\frac{N}{C(w_1)} + \sum_{i=1}^{n-1}\log\frac{C(w_i)}{C(w_i,w_{i+1})},$$

On the Generate Phrase Method:
Takes startWord, endWord, a limit of the depth of the Dijsktra's search, and r a number of allowed repetitions per word in path. Generate phrase checks if both words are in the graph and if either is missing returns and empty string because there will be no path (see test 33). Run Dijsktra's algorithm on the graph with the starting node. The algorithm stores the most recent node, the path cost, the length of the path, and the number of repetitions of each node and searches for the lowest cost path to get to the end word but stops searching a specific path if the limit of path length has been reached. It allows

repetitions of words however this allowance will never change the word string that the method returns because adding more words and repeating them will never be a lower cost than a phrase with each word used only once. This could be rectified if the limit N was a minimum number of words and thus the path would sometimes have to use duplicates to reach that minimum. However, here N is a maximum and repetition of words won't be effective. The phrase method without repetition is demonstrated in tests 28 through 33. The phrase method with repetitions is demonstrated in tests 34 through 39.

Reasoning for each test in main method:
run WordGraph
1. When attempting to open a file that does not exist the program should tell the user the file doesn't exist and take no further action.
File not found. Please try again.

For the following series of test a file containing: "this is a test a this is a more words for test" will be used contained in test.txt
Test Num Nodes
2. The number of nodes created from this file should be seven.
7
Test Num Edges
3. The number of edges created from this file should be nine.
9
Test Word Counts
4. The word count for a word appearing once should be one (using more)
1
5. The word count for a word appearing twice should be two (using this)
2
6. The word count for a word not appearing in the source text should be negative one (using not)
-1
Test In Degrees
7. The indegree for "more" should be: 1 //checks a normal word
1
8. The indegree for "this" should be: 1 //checks the beginning word
1
9. The indegree for "not" should be: -1 //checks for a word not in the graph
-1
10. The indegree for "test" should be: 2 //checks a word with more than one
2
Test Out Degrees
11. The outdegree for "more" should be: 1 //checks a normal word
1
12. The outdegree for "this" should be: 1 //checks the first word
1
13. The outdegree for "not" should be: -1 //checks a word not in the graph
-1
14. The outdegree for "a" should be: 3 //checks a word with more than one
3
Test Previous Words
15. The prevWords for "more" should be: "a" //checks a normal word

a
16. The prevWords for "this" should be: "a" //checks a word with duplicate previous words
a
17. The prevWords for "not" should be: null //checks a word not in the graph
null
18. The prevWords for "a" should be: "is, test" //checks a word with duplicates and more than one
is test
19. The prevWords for "test" should be: "a, for" //checks a word with more than one
a for

Test Next Words

20. The nextWords for "more" should be: "words" //checks a normal word
words
21. The nextWords for "this" should be: "is" //checks a word with duplicates
is
22. The nextWords for "not" should be: null //checks a word not in the graph
null
23. The nextWords for "a" should be: "test, this, more" //checks a word with more than one
test this more
24. The nextWords for "test" should be: "a" //checks a word with just one
a

Test Word Sequence Cost

/*uses the total number of words divided by the count of the word: the next two cases should be the same because each additional word has an edge weight with the previous word of one leading to log(1) or adding 0 to the path cost*/

25. When using "this", "is", "a" as a test for word sequence cost the program should return: ~1.79.
1.791759469228055
26. When using "this" as a test for word sequence cost the program should return: ~1.79.
1.791759469228055
//here we can see actually higher costs
26. When using "is", "a", "more", "words", "for", "test" as a test for word sequence cost the program should return: ~2.89.
2.8903717578961645
27. Testing when a word is not in the graph is not applicable because the specifications said to assume the words are in the graph.

Tests of Generate Phrase with starting word, ending word, and limit parameters

28. Generate Phrase with "this", "is" and 5 should return: "this is". //two words directly connected with a limit higher than 2
this is
29. Generate Phrase with "this", "is" and 1 should return: "". //two words directly connected testing limit

30. Generate Phrase with "this", "for" and 7 should return: "this is a more words for". //case of words being indirectly connected
this is a more words for
31. Generate Phrase with "this", "this" and 5 should return: "this". //the same word path should always be one

this
32. Generate Phrase with "test", "for" and 5 should return: "test a more words for". //test a path that has the same number of words as the limit
test a more words for
32. Generate Phrase with "test", "for" and 4 should return: "".//test an existing path with a limit one less than path length

33. Generate Phrase with "test", "not" and 4 should return: "". //test a word not in the graph

Tests of Generate Phrase with starting word, ending word, limit and repetitions parameters.

/*here we can see the addition of repetitive words does not change any of the shortest paths same cases as above*/

34. Generate Phrase with "this", "is", 5, and 3 should return: "this is".
this is
35. Generate Phrase with "this", "is", 1, and 5 should return: "".

36. Generate Phrase with "this", "for" 7, and 6 should return: "this is a more words for".
this is a more words for
37. Generate Phrase with "this", "this" 5, and 12 should return: "this".
this
38. Generate Phrase with "test", "for" 5, and 2 should return: "test a more words for".
test a more words for
39. Generate Phrase with "test", "for" 4, and 3 should return: "".

40. Generate Phrase with "test", "not" 4, and 1 should return: "".

Testing an Empty File
41. Testing making a graph of an empty file. Should have num nodes of 0.
0
Testing a one word file
42. Testing making a graph of a file with one word. Should have num nodes of 1.
1
43. Testing making a graph of a file with one word. Should have indegree of 0.
0
44. Testing making a graph of a file with one word. Should have outdegree of 0.
0