Robert Weischedel (u0887905)

Nick Moore (u0847478)

CS 3505

2/16/15

Assignment 4 Testing

Information: Using Robert Weischedel GitHub u0887905

For each of the ten tests that we made we ensure that we check the correct amount of nodes are being added and deleted depending on the various Trie operations we perform. If we added a word using addWord the correct number of nodes were added to the refCount and when we deleted the Trie at the end of the test the refCount value would return to 0. Each of the tests we designed test each of the methods and constructors that were a part of the Trie class.

TestAdd1Word - Checks if adding one word to the Trie works and if the correct number of nodes are generated. We add the word “hiya” to the Trie and it was added correctly along with correctly counting 5 nodes in total. One for the root node, and 4 for the 4 letters in “hiya”.

TestAdd2Words - Does the same thing as the previous test except that it adds two words to the Trie. We add two words that don’t have anything in common so that they won’t share any of the same nodes. We test to see if both words were added, that the correct number of nodes were added (which with the words computer and science) for a total of 16, and then ensuring the Trie and its Nodes have been deleted.

TestAdd2Similiar – Does the same thing as the past two tests except that it adds two words that would share many of the same nodes. This test was to ensure that that the refCount doesn’t over count the needed number of nodes.

TestAdd3Similiar – Does the same thing as the past three tests except that it adds three words that would share many of the same nodes. This test was to ensure that the refCount doesn’t over count the needed number of nodes.

TestTrieCopyConstructor1 – This test checks if the copy constructor works. That if we create one Trie object and try to create a new Trie object by copying the original one into a new one that the values and not the pointers are copied over. This test specifically tries to copy a Trie with only one word in it. And to also ensure that when we delete the two objects no issues occur.

TestTrieCopyConstructor2 – This test is similar to the above test except that we copy a Trie object that contains three values. We also test and ensure that when more words are added to the two different Trie objects they aren’t pointers to each other and the word is only added to the respective Trie object.

TestTriePrefixFinder1 – This test sees that when a Trie object that contains three words that have all the same prefixes and if the user tries to find a word that contains the prefix that is not in the Trie that the method will return the correct list of words beginning with that prefix. Which in this test would be three words.

TestTriePrefixFinder2 – This test performs a similar test like the one above, except that the user tries to search for a word that doesn't exist in the Trie. It ensure that in this case an empty vector is returned.

TestTrieEqualOperator1 – This test uses the = override in order to copy a Trie object into a different Trie object. For this test we copy a simple Trie that consists of three words and then ensure that both Tries contain the same words.

TestTrieEqualOperator2 – This test is the same as above, except that that Tire being copied only contains one value.