WordLadder.java

```
   package String;
   import java.util.HashSet;
   import java.util.LinkedList;
   import java.util.List;
   import java.util.Queue;

   /*
       **Problem Statement:**
       A transformation sequence from word beginWord to word endWord using a dictionary wordList is a
      sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

       Every adjacent pair of words differs by a single letter.
       Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
       sk == endWord
       Given two words, beginWord and endWord, and a dictionary wordList, return the number of words in
      the shortest transformation sequence from beginWord to endWord, or 0 if no such sequence exists.

       **Example 1:**
       Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]
       Output: 5
       Explanation: One shortest transformation sequence is "hit" -> "hot" -> "dot" -> "dog" -> cog",
      which is 5 words long.

       **Example 2:**
       Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]
       Output: 0
       Explanation: The endWord "cog" is not in wordList, therefore there is no valid transformation
      sequence.

       **Constraints:**
       1 <= beginWord.length <= 10
       endWord.length == beginWord.length
       1 <= wordList.length <= 5000
       wordList[i].length == beginWord.length
       beginWord, endWord, and wordList[i] consist of lowercase English letters.
       beginWord != endWord
       All the words in wordList are unique.
    */

   public class WordLadder {
       public static int ladderLength(String beginWord, String endWord, List<String> wordList) {
           HashSet<String> set = new HashSet(wordList);

1          if (!set.contains(endWord)) {
               return 0;
           }

           Queue<String> queue = new LinkedList();
           queue.offer(beginWord);
           int level = 1;

1          while (!queue.isEmpty()) {
               int size = queue.size();
2              for (int i = 0; i < size; i++) {
                   String curr = queue.poll();
                   char[] words_chars = curr.toCharArray();
2                  for (int j = 0; j < words_chars.length; j++) {
                       char original_chars = words_chars[j];
3                      for (char c = 'a'; c <= 'z'; c++) {
1                          if (words_chars[j] == c) {
                               continue;
                           }
                           words_chars[j] = c;
                           String new_word = String.valueOf(words_chars);
1                          if (new_word.equals(endWord)) {
2                              return level + 1;
                           }
1                          if (set.contains(new_word)) {
                               set.remove(new_word);
                               queue.offer(new_word);
                           }
72                        }
73                        words_chars[j] = original_chars;
74                    }
75                }
76 1              level++;
77            }
78            return 0;
79        }
80    }
```

Mutations

```
44  1. negated conditional → KILLED
52  1. negated conditional → KILLED
54  1. negated conditional → TIMED_OUT
    2. changed conditional boundary → KILLED
57  1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
59  1. Replaced integer addition with subtraction → KILLED
    2. negated conditional → KILLED
    3. changed conditional boundary → SURVIVED
60  1. negated conditional → KILLED
65  1. negated conditional → KILLED
66  1. Replaced integer addition with subtraction → KILLED
    2. replaced int return with 0 for String/WordLadder::ladderLength → KILLED
68  1. negated conditional → KILLED
76  1. Changed increment from 1 to -1 → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- String.WordLadderTest.testWordLadder(String.WordLadderTest) (0 ms)

Report generated by PIT 1.15.0