Overview

A **word chain** is a puzzle where the challenge is to connect two words by a series of intermediate words, each of which must be pulled from a bag and differ from the previous word by a single letter.

The WordChainSolver project included with this exercise contains the following unimplemented class:

```
class WordChainSolver {
  val words : Set[String] =
    ???

  def chain (from : String, to : String) : Option[Vector[String]] =
    ???
}
```

Where: (1) `words` is a bag of words initialised from the `dictionary.txt` supplied with the project; and (2) the `chain()` should, given the bag of `words`, find the shortest word chain between `from` and `to`, or return None if such a chain does not exist.

You should aim to maximise the efficiency of your algorithm, and ensure that any corner cases are handled gracefully. The chains it produces should be the shortest possible (in the case of multiple candidates of equal length, pick arbitrarily). The following sample chains are relatively easy to evaluate:

- warm, ward, word, wold, cold
- lead, load, goad, gold
- show, shot, soot, boot, boat

Your solution is expected to be syntactically valid Scala 2.11 code that compiles and executes successfully. Take note of the testing framework already set up as part of the WordChainSolver project.

Running:

```
$ sbt run
```

Testing:

```
$ sbt test
```

You may find it faster to enter the SBT REPL (`$ sbt`) first.

Resources

- Scala 2.11 Standard Library: https://www.scala-lang.org/files/archive/api/2.11.11
- Scalatest (specifically: matchers): http://www.scalatest.org/user_guide/using_matchers