Solution 1

```java
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

import java.util.*;

class Program {

  public static class stringChain {
    String nextString;
    Integer maxChainLength;

    public stringChain(String nextString, Integer maxChainLength) {
      this.nextString = nextString;
      this.maxChainLength = maxChainLength;
    }
  }

  // O(n * m^2 + nlog(n)) time | O(nm) space - where n is the number of strings
  // and m is the length of the longest string
  public static List<String> longestStringChain(List<String> strings) {
    // For every string, imagine the longest string chain that starts with it.
    // Set up every string to point to the next string in its respective longest
    // string chain. Also keep track of the lengths of these longest string
    // chains.
    Map<String, stringChain> stringChains = new HashMap<String, stringChain>();
    for (String string : strings) {
      stringChains.put(string, new stringChain("", 1));
    }

    // Sort the strings based on their length so that whenever we visit a
    // string (as we iterate through them from left to right), we can
    // already have computed the longest string chains of any smaller strings.
    List<String> sortedStrings = new ArrayList<String>(strings);
    sortedStrings.sort((a, b) -> a.length() - b.length());

    for (String string : sortedStrings) {
      findLongestStringChain(string, stringChains);
    }

    return buildLongestStringChain(strings, stringChains);
  }

  public static void findLongestStringChain(String string, Map<String, stringChain> stringChains) {
    // Try removing every letter of the current string to see if the
    // remaining strings form a string chain.
    for (int i = 0; i < string.length(); i++) {
      String smallerString = getSmallerString(string, i);
      if (!stringChains.containsKey(smallerString)) continue;
      tryUpdateLongestStringChain(string, smallerString, stringChains);
    }
  }

  public static String getSmallerString(String string, int index) {
    return string.substring(0, index) + string.substring(index + 1);
  }

  public static void tryUpdateLongestStringChain(
      String currentString, String smallerString, Map<String, stringChain> stringChains) {
    int smallerStringChainLength = stringChains.get(smallerString).maxChainLength;
    int currentStringChainLength = stringChains.get(currentString).maxChainLength;
    // Update the string chain of the current string only if the smaller string
    // leads to a longer string chain.
    if (smallerStringChainLength + 1 > currentStringChainLength) {
      stringChains.get(currentString).maxChainLength = smallerStringChainLength + 1;
      stringChains.get(currentString).nextString = smallerString;
    }
  }

  public static List<String> buildLongestStringChain(
      List<String> strings, Map<String, stringChain> stringChains) {
    // Find the string that starts the longest string chain.
    int maxChainLength = 0;
    String chainStartingString = "";
    for (String string : strings) {
      if (stringChains.get(string).maxChainLength > maxChainLength) {
        maxChainLength = stringChains.get(string).maxChainLength;
        chainStartingString = string;
      }
    }

    // Starting at the string found above, build the longest string chain.
    List<String> ourLongestStringChain = new ArrayList<String>();
    String currentString = chainStartingString;
    while (currentString != "") {
      ourLongestStringChain.add(currentString);
      currentString = stringChains.get(currentString).nextString;
    }

    return ourLongestStringChain.size() == 1 ? new ArrayList<String>() : ourLongestStringChain;
  }
}

```