Solution 1

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

using System.Collections.Generic;

public class Program {

  public class stringChain {
    public string nextstring;
    public int maxChainLength;

    public stringChain(string nextstring, int 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.
    Dictionary<string,
      stringChain> stringChains = new Dictionary<string, stringChain>();
    foreach (string str in strings) {
      stringChains[str] = 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 List<string>(strings);
    sortedstrings.Sort((a, b) => a.Length - b.Length);

    foreach (string str in sortedstrings) {
      findLongeststringChain(str, stringChains);
    }

    return buildLongeststringChain(strings, stringChains);
  }

  public static void findLongeststringChain(string str, Dictionary<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 < str.Length; i++) {
      string smallerstring = getSmallerstring(str, i);
      if (!stringChains.ContainsKey(smallerstring)) continue;
      tryUpdateLongeststringChain(str, smallerstring, stringChains);
    }
  }

  public static string getSmallerstring(string str, int index) {
    return str.Substring(0, index) + str.Substring(index + 1);
  }

  public static void tryUpdateLongeststringChain(
    string currentstring,
    string smallerstring,
    Dictionary<string, stringChain> stringChains
    ) {
    int smallerstringChainLength = stringChains[smallerstring].maxChainLength;
    int currentstringChainLength = stringChains[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[currentstring].maxChainLength = smallerstringChainLength + 1;
      stringChains[currentstring].nextstring = smallerstring;
    }
  }

  public static List<string> buildLongeststringChain(List<string> strings, Dictionary<string,
    stringChain> stringChains) {
    // Find the string that starts the longest string chain.
    int maxChainLength = 0;
    string chainStartingstring = "";
    foreach (string str in strings) {
      if (stringChains[str].maxChainLength > maxChainLength) {
        maxChainLength = stringChains[str].maxChainLength;
        chainStartingstring = str;
      }
    }

    // Starting at the string found above, build the longest string chain.
    List<string> ourLongeststringChain = new List<string>();
    string currentstring = chainStartingstring;
    while (currentstring != "") {
      ourLongeststringChain.Add(currentstring);
      currentstring = stringChains[currentstring].nextstring;
    }

    return ourLongeststringChain.Count ==
          1 ? new List<string>() : ourLongeststringChain;
  }
}

```