Solution 1

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

// 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
function longestStringChain(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.
  const stringChains = {};
  for (const string of strings) {
    stringChains[string] = {nextString: '', maxChainLength: 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.
  const sortedStrings = strings.sort((a, b) => a.length - b.length);
  for (const string of sortedStrings) {
    findLongestStringChain(string, stringChains);
  }

  return buildLongestStringChain(strings, stringChains);
}

function findLongestStringChain(string, stringChains) {
  // Try removing every letter of the current string to see if the
  // remaining strings form a string chain.
  for (let i = 0; i < string.length; i++) {
    const smallerString = getSmallerString(string, i);
    if (!(smallerString in stringChains)) continue;
    tryUpdateLongestStringChain(string, smallerString, stringChains);
  }
}

function getSmallerString(string, index) {
  return string.slice(0, index) + string.slice(index + 1);
}

function tryUpdateLongestStringChain(currentString, smallerString, stringChains) {
  const smallerStringChainLength = stringChains[smallerString].maxChainLength;
  const 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;
  }
}

function buildLongestStringChain(strings, stringChains) {
  // Find the string that starts the longest string chain.
  let maxChainLength = 0;
  let chainStartingString = '';
  for (const string of strings) {
    if (stringChains[string].maxChainLength > maxChainLength) {
      maxChainLength = stringChains[string].maxChainLength;
      chainStartingString = string;
    }
  }

  // Starting at the string found above, build the longest string chain.
  const ourLongestStringChain = [];
  let currentString = chainStartingString;
  while (currentString !== '') {
    ourLongestStringChain.push(currentString);
    currentString = stringChains[currentString].nextString;
  }

  return ourLongestStringChain.length === 1 ? [] : ourLongestStringChain;
}

exports.longestStringChain = longestStringChain;

```