Solution 1

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

package main

import (
  "sort"
)

type Chain struct {
  NextString     string
  MaxChainLength int
}

// 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
func LonguestStringChain(strings []string) []string {
  // 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.
  stringChains := map[string]*Chain{}
  for _, str := range strings {
    stringChains[str] = &Chain{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.
  sort.Slice(strings, func(i, j int) bool {
    return len(strings[i]) < len(strings[j])
  })
  sortedStrings := strings

  for _, str := range sortedStrings {
    findLongestStringChain(str, stringChains)
  }
  return buildLongestStringChain(strings, stringChains)
}

func findLongestStringChain(str string, stringChains map[string]*Chain) {
  // Try removing every letter of the current string to see if the
  // remaining strings form a string chain.
  for i := range str {
    smallerString := getSmallerString(str, i)
    if _, found := stringChains[smallerString]; !found {
      continue
    }
    tryUpdateLongestStringChain(str, smallerString, stringChains)
  }
}

func getSmallerString(str string, index int) string {
  return str[:index] + str[index+1:]
}

func tryUpdateLongestStringChain(currentString, smallerString string, stringChains map[string]*Chain) {
  smallerStringChainLength := stringChains[smallerString].MaxChainLength
  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
  }
}

func buildLongestStringChain(strings []string, stringChains map[string]*Chain) []string {
  // Find the string that starts the longest string chain.
  maxChainLength := 0
  chainStartingString := ""
  for _, str := range strings {
    if stringChains[str].MaxChainLength > maxChainLength {
      maxChainLength = stringChains[str].MaxChainLength
      chainStartingString = str
    }
  }

  // Starting at the string found above, build the longest string chain.
  ourLongestStringChain := []string{}
  currentString := chainStartingString
  for currentString != "" {
    ourLongestStringChain = append(ourLongestStringChain, currentString)
    currentString = stringChains[currentString].NextString
  }
  if len(ourLongestStringChain) == 1 {
    return []string{}
  }
  return ourLongestStringChain
}

```