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

package main

// O(n^2) time | O(n) space
func LongestIncreasingSubsequence(array []int) []int {
  sequences := make([]int, len(array))
  lengths := make([]int, len(array))
  for i := range array {
    sequences[i] = -1
    lengths[i] = 1
  }
  for i := range array {
    currentNum := array[i]
    for j := 0; j < i; j++ {
      otherNum := array[j]
      if otherNum < currentNum && lengths[j]+1 >= lengths[i] {
        lengths[i] = lengths[j] + 1
        sequences[i] = j
      }
    }
  }
  maxLengthIndex := 0
  for i := range array {
    if lengths[i] > lengths[maxLengthIndex] {
      maxLengthIndex = i
    }
  }
  return buildSequence(array, sequences, maxLengthIndex)
}

func buildSequence(array, sequences []int, index int) []int {
  out := []int{}
  for index != -1 {
    out = append(out, array[index])
    index = sequences[index]
  }
  reverse(out)
  return out
}

func reverse(numbers []int) {
  for i, j := 0, len(numbers)-1; i < j; i, j = i+1, j-1 {
    numbers[i], numbers[j] = numbers[j], numbers[i]
  }
}

```