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

package main

// O(nlogn) time | O(n) space
func LongestIncreasingSubsequence(array []int) []int {
  sequences := make([]int, len(array))
  indices := make([]int, len(array)+1)
  for i := range array {
    sequences[i] = -1
    indices[i] = -1
  }
  length := 0
  for i, num := range array {
    newLength := binarySearch(1, length, indices, array, num)
    sequences[i] = indices[newLength-1]
    indices[newLength] = i
    length = max(length, newLength)
  }
  return buildSequence(array, sequences, indices[length])
}

func binarySearch(startIndex, endIndex int, indices, array []int, num int) int {
  if startIndex > endIndex {
    return startIndex
  }
  middleIndex := (startIndex + endIndex) / 2
  if array[indices[middleIndex]] < num {
    startIndex = middleIndex + 1
  } else {
    endIndex = middleIndex - 1
  }
  return binarySearch(startIndex, endIndex, indices, array, num)
}

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 max(arg int, rest ...int) int {
  for _, num := range rest {
    if num > arg {
      arg = num
    }
  }
  return arg
}

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]
  }
}

```