AlgoExpert | Quad Layout | Go | 12px | Sublime | Monokai | 00:00:00

Prompt | Scratchpad | Our Solution(s) | Video Explanation | Run Code

Solution 1

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

package main

import "math"

// O(n^2) time | O(n) space
func MaxSumIncreasingSubsequence(array []int) []interface{} {
  sequences := make([]int, len(array))
  sums := make([]int, len(array))
  for i := range sequences {
    sequences[i] = math.MinInt32
    sums[i] = array[i]
  }
  maxSumIndex := 0
  for i, currentNum := range array {
    for j := 0; j < i; j++ {
      otherNum := array[j]
      if otherNum < currentNum && sums[j]+currentNum >= sums[i] {
        sums[i] = sums[j] + currentNum
        sequences[i] = j
      }
    }
    if sums[i] > sums[maxSumIndex] {
      maxSumIndex = i
    }
  }

  maxSum := sums[maxSumIndex]
  sequence := buildSequence(array, sequences, maxSumIndex)
  return []interface{}{maxSum, sequence}
}

func buildSequence(array []int, sequences []int, index int) []int {
  sequence := []int{}
  for index != math.MinInt32 {
    sequence = append(sequence, array[index])
    index = sequences[index]
  }
  reverse(sequence)
  return sequence
}

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]
  }
}
```