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

class Program {
    // O(nlogn) time | O(n) space
    func longestIncreasingSubsequence(_ array: [Int]) -> [Int] {
        var length = 0
        var indices: [Int?] = Array(repeating: nil, count: array.count + 1)
        var sequences: [Int?] = Array(repeating: nil, count: array.count)

        for i in 0 ..< array.count {
            var startIndex = 1
            var endIndex = length
            let number = array[i]
            let newLength = binarySearch(&startIndex, &endIndex, indices, array, number)

            indices[newLength] = i
            sequences[i] = indices[newLength - 1]
            length = max(length, newLength)
        }

        return buildSequence(array, sequences, &indices[length])
    }

    func binarySearch(_ startIndex: inout Int, _ endIndex: inout Int, _ indices: [Int?], _ array: [Int], _ number: Int) -> Int {
        if startIndex > endIndex {
            return startIndex
        }

        var middleIndex = Double(startIndex + endIndex) / 2
        middleIndex = middleIndex.rounded(.down)
        let intMiddle = Int(middleIndex)

        if let index = indices[intMiddle] {
            let numberToCompare = array[index]

            if numberToCompare < number {
                startIndex = intMiddle + 1
            } else {
                endIndex = intMiddle - 1
            }
        }

        return binarySearch(&startIndex, &endIndex, indices, array, number)
    }

    func buildSequence(_ array: [Int], _ sequences: [Int?], _ currentIndex: inout Int?) -> [Int] {
        var sequence = [Int]()

        while currentIndex != nil {
            sequence.insert(array[currentIndex!], at: 0)
            currentIndex = sequences[currentIndex!]
        }

        return sequence
    }
}

```