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

class Program {
    // O(nm) time | O(nm) space
    func longestCommonSubsequence(firstString: String, secondString: String) -> [String] {
        var lcs = [[(String, Int, Int, Int)]]()

        for _ in stride(from: 0, to: firstString.count + 1, by: 1) {
            var row = [(String, Int, Int, Int)]()

            for _ in stride(from: 0, to: secondString.count + 1, by: 1) {
                let tuple = ("", 0, 0, 0)
                row.append(tuple)
            }

            lcs.append(row)
        }

        for i in stride(from: 1, to: firstString.count + 1, by: 1) {
            for j in stride(from: 1, to: secondString.count + 1, by: 1) {
                let firstIndex = firstString.index(firstString.startIndex, offsetBy: i - 1)
                let secondIndex = secondString.index(secondString.startIndex, offsetBy: j - 1)

                if firstString[firstIndex] == secondString[secondIndex] {
                    let char = String(firstString[firstIndex])
                    lcs[i][j] = (char, lcs[i - 1][j - 1].1 + 1, i - 1, j - 1)
                } else {
                    if lcs[i - 1][j].1 > lcs[i][j - 1].1 {
                        lcs[i][j] = ("", lcs[i - 1][j].1, i - 1, j)
                    } else {
                        lcs[i][j] = ("", lcs[i][j - 1].1, i, j - 1)
                    }
                }
            }
        }

        return buildSequence(lcs: lcs)
    }

    func buildSequence(lcs: [[(String, Int, Int, Int)]]) -> [String] {
        var sequence = [String]()

        var i = lcs.count - 1
        var j = lcs[0].count - 1

        while i != 0, j != 0 {
            let currentEntry = lcs[i][j]

            if currentEntry.0 != "" {
                sequence.insert(currentEntry.0, at: 0)
            }

            i = currentEntry.2
            j = currentEntry.3
        }

        return sequence
    }
}

```