Our Solution(s)   Run Code

Solution 1   Solution 2   Solution 3   Solution 4

```go
1  // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3  package main
4
5  type entry struct {
6    letter byte
7    length int
8    previ  int
9    prevj  int
10 }
11
12 // O(nm) time | O(nm) space
13 func LongestCommonSubsequence(str1 string, str2 string) string {
14   lcs := make([][]entry, len(str2)+1)
15   for i := range lcs {
16     lcs[i] = make([]entry, len(str1)+1)
17     for j := range lcs[i] {
18       lcs[i][j].letter = 0
19       lcs[i][j].length = 0
20       lcs[i][j].previ = -1
21       lcs[i][j].prevj = -1
22     }
23   }
24
25   for i := 1; i < len(str2)+1; i++ {
26     for j := 1; j < len(str1)+1; j++ {
27       if str2[i-1] == str1[j-1] {
28         lcs[i][j] = entry{str2[i-1], lcs[i-1][j-1].length + 1, i - 1,
29       } else {
30         if lcs[i-1][j].length > lcs[i][j-1].length {
31           lcs[i][j] = entry{0, lcs[i-1][j].length, i - 1, j}
32         } else {
33           lcs[i][j] = entry{0, lcs[i][j-1].length, i, j - 1}
```

Your Solutions   Run Code

Solution 1   Solution 2   Solution 3

```go
1  package main
2
3  // O(nm*min(n, m)) time | O(nm*min(n, m)) space
4  func LongestCommonSubsequence(s1 string, s2 string) string {
5    // Write your code here.
6    return ""
7  }
8
```

Our Tests

Custom Output   Submit Code

Run or submit code when you're ready.