Solution 1   Solution 2   Solution 3   Solution 4

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

import java.util.*;

class Program {
  // O(nm*min(n, m)) time | O((min(n, m))^2) space
  public static List<Character> longestCommonSubsequence(String str1, String str2) {
    String small = str1.length() < str2.length() ? str1 : str2;
    String big = str1.length() >= str2.length() ? str1 : str2;
    List<List<Character>> evenLcs = new ArrayList<List<Character>>();
    List<List<Character>> oddLcs = new ArrayList<List<Character>>();
    for (int i = 0; i < small.length() + 1; i++) {
      evenLcs.add(new ArrayList<Character>());
    }
    for (int i = 0; i < small.length() + 1; i++) {
      oddLcs.add(new ArrayList<Character>());
    }
    for (int i = 1; i < big.length() + 1; i++) {
      List<List<Character>> currentLcs;
      List<List<Character>> previousLcs;
      if (i % 2 == 1) {
        currentLcs = oddLcs;
        previousLcs = evenLcs;
      } else {
        currentLcs = evenLcs;
        previousLcs = oddLcs;
      }
      for (int j = 1; j < small.length() + 1; j++) {
        if (big.charAt(i - 1) == small.charAt(j - 1)) {
          List<Character> copy = new ArrayList<Character>(previousLcs.get(j - 1));
          currentLcs.set(j, copy);
          currentLcs.get(j).add(big.charAt(i - 1));
        } else {
          if (previousLcs.get(j).size() > currentLcs.get(j - 1).size()) {
            currentLcs.set(j, previousLcs.get(j));
          } else {
            currentLcs.set(j, currentLcs.get(j - 1));
          }
        }
      }
    }
    return big.length() % 2 == 0 ? evenLcs.get(small.length()) : oddLcs.get(small.length());
  }
}
```