LCS.java

```
   package DynamicProgramming;

   import java.util.Arrays;

   public class LCS {
       //Approach-1
       public static int longestCommonSubsequenceApproach1(String text1, String text2) {
2          int result = helper1(text1, text1.length() - 1, text2, text2.length() - 1);
1          return result;
       }
       public static int helper1(String text1, int indexOne, String text2, int indexTwo) {
4          if(indexOne < 0 || indexTwo < 0) return 0;

5          if(text1.charAt(indexOne) == text2.charAt(indexTwo)) return 1 + helper1(text1, indexOne - 1, text

3          return Math.max(helper1(text1, indexOne - 1, text2, indexTwo), helper1(text1, indexOne, text2, in
       }

       //Approach-2
       public static int longestCommonSubsequenceApproach2(String text1, String text2) {
           int n = text1.length();
           int m = text2.length();

           int[][] dp = new int[n][m];
           for(int[] row: dp) {
1              Arrays.fill(row, -1);
           }

2          int result = helper2(text1, n - 1, text2, m - 1, dp);
1          return result;
       }
       public static int helper2(String text1, int i, String text2, int j, int[][]dp) {
4          if(i < 0 || j < 0) return 0;
2          if(dp[i][j] != -1) return dp[i][j];

1          if(text1.charAt(i) == text2.charAt(j)) {
3              dp[i][j] = 1 + helper2(text1, i - 1, text2, j - 1, dp);
1              return dp[i][j];
           }

2          dp[i][j] = Math.max(helper2(text1, i - 1, text2, j, dp), helper2(text1, i, text2, j - 1, dp));
1          return dp[i][j];
       }

       //Approach-3
       public int longestCommonSubsequenceApproach3(String text1, String text2) {
           int n = text1.length();
           int m = text2.length();

           // Shift index ie 0 -> -1, 1 -> 0 and so on n -> n - 1
2          int[][] dp = new int[n + 1][m + 1];

           // Base Case:
2          for(int i = 0; i <= n; i++) {
               dp[i][0] = 0;
           }

2          for(int j = 0; j <= m; j++) {
               dp[0][j] = 0;
           }

2          for(int i = 1; i <= n; i++) {
2              for(int j = 1; j <= m; j++) {
3                  if(text1.charAt(i - 1) == text2.charAt(j - 1)) {
3                      dp[i][j] = 1 + dp[i - 1][j - 1];
                   } else {
2                      dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
                   }
               }
           }
1          return dp[n][m];
       }
   }
```

Mutations

```
8   1. Replaced integer subtraction with addition → KILLED
    2. Replaced integer subtraction with addition → KILLED
9   1. replaced int return with 0 for DynamicProgramming/LCS::longestCommonSubsequenceApproach1 → KILLED
    1. changed conditional boundary → KILLED
12  2. negated conditional → KILLED
    3. changed conditional boundary → KILLED
    4. negated conditional → KILLED
    1. Replaced integer subtraction with addition → KILLED
    2. replaced int return with 0 for DynamicProgramming/LCS::helper1 → KILLED
14  3. Replaced integer subtraction with addition → KILLED
    4. Replaced integer addition with subtraction → KILLED
    5. negated conditional → KILLED
    1. replaced int return with 0 for DynamicProgramming/LCS::helper1 → KILLED
16  2. Replaced integer subtraction with addition → KILLED
    3. Replaced integer subtraction with addition → KILLED
26  1. removed call to java/util/Arrays::fill → KILLED
29  1. Replaced integer subtraction with addition → KILLED
    2. Replaced integer subtraction with addition → KILLED
30  1. replaced int return with 0 for DynamicProgramming/LCS::longestCommonSubsequenceApproach2 → KILLED
    1. negated conditional → KILLED
33  2. changed conditional boundary → KILLED
    3. changed conditional boundary → KILLED
    4. negated conditional → KILLED
34  1. negated conditional → KILLED
    2. replaced int return with 0 for DynamicProgramming/LCS::helper2 → SURVIVED
36  1. negated conditional → KILLED
    1. Replaced integer addition with subtraction → KILLED
37  2. Replaced integer subtraction with addition → KILLED
    3. Replaced integer subtraction with addition → KILLED
38  1. replaced int return with 0 for DynamicProgramming/LCS::helper2 → KILLED
41  1. Replaced integer subtraction with addition → KILLED
    2. Replaced integer subtraction with addition → KILLED
42  1. replaced int return with 0 for DynamicProgramming/LCS::helper2 → KILLED
51  1. Replaced integer addition with subtraction → KILLED
    2. Replaced integer addition with subtraction → KILLED
54  1. changed conditional boundary → SURVIVED
    2. negated conditional → SURVIVED
58  1. changed conditional boundary → SURVIVED
    2. negated conditional → SURVIVED
62  1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
63  1. negated conditional → KILLED
    2. changed conditional boundary → KILLED
    1. Replaced integer subtraction with addition → KILLED
64  2. negated conditional → KILLED
    3. Replaced integer subtraction with addition → KILLED
    1. Replaced integer subtraction with addition → KILLED
65  2. Replaced integer subtraction with addition → KILLED
    3. Replaced integer addition with subtraction → KILLED
67  1. Replaced integer subtraction with addition → KILLED
    2. Replaced integer subtraction with addition → KILLED
71  1. replaced int return with 0 for DynamicProgramming/LCS::longestCommonSubsequenceApproach3 → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- DynamicProgramming.LCSTest.testApproach1(DynamicProgramming.LCSTest) (0 ms)
- DynamicProgramming.LCSTest.testApproach2(DynamicProgramming.LCSTest) (0 ms)
- DynamicProgramming.LCSTest.testApproach3(DynamicProgramming.LCSTest) (0 ms)

Report generated by PIT 1.15.0