LongestPalindromicSubsequence.java

```
1   package DynamicProgramming;
2
3   public class LongestPalindromicSubsequence {
4
5   /*https://www.techiedelight.com/longest-palindromic-subsequence-using-dynamic-programming/#:~:text=Longest%20Palindrc
6   General Idea
7                           |       1                                                                   (if i == j)
8   LPS[i..j] = |   LPS[i+1...j-1]+2                                    (if S[i] == S[j])
9                           |       Max(LPS[i+1...j],LPS[i...j-1])      (if S[i] != S[j])
10
11  */
12
13  /*******************
14   * Approach 1: using recursion
15   *
16   * Time Complexity : O(2^(n)
17   *
18   * Space Complexity : O(n)
19   ********************************/
20  public static int longestPalindromeSubseqApproach1(String s) {
21 2        return longestPalindromeSubseqApproach1Helper(0, s.length() - 1, s);
22  }
23
24  public static int longestPalindromeSubseqApproach1Helper(int l, int r, String s) {
25 1        if (l == r)
26 1                return 1;
27 2        if (l > r)
28                  return 0; // happens after "aa"
29 5        return s.charAt(l) == s.charAt(r) ? 2 + longestPalindromeSubseqApproach1Helper(l + 1, r - 1, s)
30 2                        : Math.max(longestPalindromeSubseqApproach1Helper(l + 1, r, s),
31                                     longestPalindromeSubseqApproach1Helper(l, r - 1, s));
32  }
33
34
35  /*******************
36   * Approach 2: Top Down Memoization
37   *
38   * Time Complexity : O(n * n)
39   *
40   * Space Complexity : O(n * n)
41   ********************************/
42  public static int longestPalindromeSubseqApproach2(String s) {
43 2        return longestPalindromeSubseqApproach2Helper(s, 0, s.length() - 1, new Integer[s.length()][s.length()]);
44  }
45
46  public static int longestPalindromeSubseqApproach2Helper(String s, int i, int j, Integer[][] memo) {
47 1        if (memo[i][j] != null) {
48 1                return memo[i][j];
49          }
50 2        if (i > j)
51                  return 0;
52 1        if (i == j)
53 1                return 1;
54
55 1        if (s.charAt(i) == s.charAt(j)) {
56 3                memo[i][j] = longestPalindromeSubseqApproach2Helper(s, i + 1, j - 1, memo) + 2;
57          } else {
58 2                memo[i][j] = Math.max(longestPalindromeSubseqApproach2Helper(s, i + 1, j, memo),
59                                       longestPalindromeSubseqApproach2Helper(s, i, j - 1, memo));
60          }
61 1        return memo[i][j];
62  }
63
64
65  /*******************
66   * Approach 3: Bottom Up Tabulation
67   *
68   * Time Complexity : O(n * n)
69   *
70   * Space Complexity : O(n * n)
71   ********************************/
72  public static int longestPalindromeSubseqApproach3(String s) {
73          int len = s.length();
74          int[][] dp = new int[len][len];
75
76 3        for (int i = len - 1; i >= 0; i--) {
77                  dp[i][i] = 1;
78 3                for (int j = i + 1; j < len; j++) {
79
80 1                        if (s.charAt(i) == s.charAt(j)) {
81 3                                dp[i][j] = dp[i + 1][j - 1] + 2;
82                          } else {
```

```
83 2                            dp[i][j] = Math.max(dp[i + 1][j], dp[i][j - 1]);
84                          }
85                  }
86          }
87 2        return dp[0][len - 1];
88  }
89
90
91  /******************
92   * Approach 4: Bottom Up Tabulation with space optimization
93   *
94   * Time Complexity : O(n * n)
95   *
96   * Space Complexity : O(n)
97   *********************************/
98  // Credit: https://leetcode.com/problems/longest-palindromic-subsequence/discuss/194748/Java-DP-From-O(n2)-to-O(n)-sp
99  /*      Very tricky DIfficult to understand
100         dp[i][j] only depends on dp[i+1][j-1](down-left), dp[i+1][j](down) and dp[i][j-1](left).
101         So if we reduce dp[n][m] to dp[m], that means, for dp[j], its down is itself, its left is dp[j-1].
102         Its down-left is a little tricky.
103         As its down-left dp[i+1][j-1] is now dp[j-1], so we need to preserve it before updating to dp[j-1.
104 */
105 public static int longestPalindromeSubseqApproach4(String s) {
106         int[] dp = new int[s.length()];
107 3       for (int i = s.length() - 1; i >= 0; i--) {
108                 dp[i] = 1;
109                 int pre = 0;
110 3               for (int j = i + 1; j < s.length(); j++) {
111                         int tmp = dp[j];
112 1                       if (s.charAt(i) == s.charAt(j)) {
113 1                               dp[j] = pre + 2;
114                         } else {
115 1                               dp[j] = Math.max(dp[j], dp[j - 1]);
116                         }
117                         pre = tmp;
118                 }
119         }
120
121 2       return dp[s.length() - 1];
122 }
123 }
```

Mutations

21	1. Replaced integer subtraction with addition → KILLED 2. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach1
25	1. negated conditional → KILLED
26	1. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach1H
27	1. negated conditional → KILLED 2. changed conditional boundary → SURVIVED
29	1. Replaced integer addition with subtraction → KILLED 2. negated conditional → KILLED 3. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach1H 4. Replaced integer subtraction with addition → KILLED 5. Replaced integer addition with subtraction → KILLED
30	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer subtraction with addition → KILLED
43	1. Replaced integer subtraction with addition → KILLED 2. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach2
47	1. negated conditional → KILLED
48	1. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach2H
50	1. negated conditional → KILLED 2. changed conditional boundary → SURVIVED
52	1. negated conditional → KILLED
53	1. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach2H
55	1. negated conditional → KILLED
56	1. Replaced integer subtraction with addition → KILLED 2. Replaced integer addition with subtraction → KILLED 3. Replaced integer addition with subtraction → KILLED
58	1. Replaced integer subtraction with addition → KILLED 2. Replaced integer addition with subtraction → KILLED
61	1. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach2H
76	1. changed conditional boundary → KILLED 2. negated conditional → KILLED 3. Replaced integer subtraction with addition → KILLED
78	1. Replaced integer addition with subtraction → KILLED 2. changed conditional boundary → KILLED 3. negated conditional → KILLED
80	1. negated conditional → KILLED
81	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer subtraction with addition → KILLED 3. Replaced integer addition with subtraction → KILLED
83	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer subtraction with addition → KILLED
87	1. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach3 2. Replaced integer subtraction with addition → KILLED
107	1. changed conditional boundary → KILLED 2. negated conditional → KILLED 3. Replaced integer subtraction with addition → KILLED
110	1. changed conditional boundary → KILLED 2. negated conditional → KILLED

	3. Replaced integer addition with subtraction → KILLED
112	1. negated conditional → KILLED
113	1. Replaced integer addition with subtraction → KILLED
115	1. Replaced integer subtraction with addition → KILLED
121	1. replaced int return with 0 for DynamicProgramming/LongestPalindromicSubsequence::longestPalindromeSubseqApproach4 2. Replaced integer subtraction with addition → 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.LongestPalindromicSubsequenceTest.testApproach2(DynamicProgramming.LongestPalindromicSubsequenceTest) (0 ms)
- DynamicProgramming.LongestPalindromicSubsequenceTest.testApproach1(DynamicProgramming.LongestPalindromicSubsequenceTest) (0 ms)
- DynamicProgramming.LongestPalindromicSubsequenceTest.testApproach4(DynamicProgramming.LongestPalindromicSubsequenceTest) (0 ms)
- DynamicProgramming.LongestPalindromicSubsequenceTest.testApproach3(DynamicProgramming.LongestPalindromicSubsequenceTest) (0 ms)

Report generated by PIT 1.15.0