Strange Printer

There is a strange printer with the following two special requirements:

1. The printer can only print a sequence of the same character each time.
2. At each turn, the printer can print new characters starting from and ending at any places, and will cover the original existing characters.

Given a string consists of lower English letters only, your job is to count the minimum number of turns the printer needed in order to print it.

Example 1:

```
Input: "aaabbb"
Output: 2
Explanation: Print "aaa" first and then print "bbb".
```

Example 2:

```
Input: "aba"
Output: 2
Explanation: Print "aaa" first and then print "b" from the second place of the string
, which will cover the existing character 'a'.
```

Hint: Length of the given string will not exceed 100.

Solution 1

```java
class Solution {
    public int strangePrinter(String s) {
        int n = s.length();
        if (n == 0) return 0;

        int[][] dp = new int[101][101];
        for (int i = 0; i < n; i++) dp[i][i] = 1;

        for (int i = 1; i < n; i++) {
            for (int j = 0; j < n - i; j++) {
                dp[j][j + i] = i + 1;
                for (int k = j + 1; k <= j + i; k++) {
                    int temp = dp[j][k - 1] + dp[k][j + i];
                    if (s.charAt(k - 1) == s.charAt(j + i)) temp--;
                    dp[j][j + i] = Math.min(dp[j][j + i], temp);
                }
            }
        }
        return dp[0][n - 1];
    }
}
```

written by shawngao original link here

Solution 2

Let `dp(i, j)` be the number of turns needed to print `S[i:j+1]`.

Note that whichever turn creates the final print of S[i], might as well be the first turn, and also there might as well only be one print, since any later prints on interval [i, k] could just be on [i+1, k].

So suppose our first print was on [i, k]. We only need to consider prints where S[i] == S[k], because we could instead take our first turn by printing up to the last printed index where S[k] == S[i] to get the same result.

Then, when trying to complete the printing of interval [i, k] (with S[i] == S[k]), the job will take the same number of turns as painting [i, k-1]. This is because it is always at least as good to print [i, k] first in one turn rather than separately.

Also, we would need to complete [k+1, j]. So in total, our candidate answer is dp(i, k-1) + dp(k+1, j). Of course, when k == i, our candidate is 1 + dp(i+1, j): we paint S[i] in one turn, then paint the rest in dp(i+1, j) turns.

```
def strangePrinter(self, S):
    memo = {}
    def dp(i, j):
        if i > j: return 0
        if (i, j) not in memo:
            ans = dp(i+1, j) + 1
            for k in xrange(i+1, j+1):
                if S[k] == S[i]:
                    ans = min(ans, dp(i, k-1) + dp(k+1, j))
            memo[i, j] = ans
        return memo[i, j]

    return dp(0, len(S) - 1)
```

written by awice original link here

Solution 3

checkout 546. Remove Boxes, it is almost the same question

written by zsheng original link here