72. Edit Distance

Hard 4331 59 Add to List Share

Given two words *word1* and *word2*, find the minimum number of operations required to convert *word1* to *word2*.

You have the following 3 operations permitted on a word:

1. Insert a character
2. Delete a character
3. Replace a character

Example 1:

```
Input: word1 = "horse", word2 = "ros"
Output: 3
Explanation:
horse -> rorse (replace 'h' with 'r')
rorse -> rose (remove 'r')
rose -> ros (remove 'e')
```

Example 2:

```
Input: word1 = "intention", word2 = "execution"
Output: 5
Explanation:
intention -> inention (remove 't')
inention -> enention (replace 'i' with 'e')
enention -> exention (replace 'n' with 'x')
exention -> exection (replace 'n' with 'c')
exection -> execution (insert 'u')
```

```java
class Solution {
    public int minDistance(String word1, String word2) {
    if (word1.equals(word2)) {
        return 0;
    }
    if (word1.length() == 0 || word2.length() == 0) {
        return Math.abs(word1.length() - word2.length());
    }
    int[][] dp = new int[word1.length() + 1][word2.length() + 1];
    for (int i = 0; i <= word1.length(); i++) {
        dp[i][0] = i;
    }
    for (int i = 0; i <= word2.length(); i++) {
        dp[0][i] = i;
    }
    for (int i = 1; i <= word1.length(); i++) {
        for (int j = 1; j <= word2.length(); j++) {
            if (word1.charAt(i - 1) == word2.charAt(j - 1)) {
                dp[i][j] = dp[i - 1][j - 1];
            } else {
                dp[i][j] = Math.min(dp[i-1][j-1], Math.min(dp[i-1][j], dp[i][j-1])) + 1;
            }
        }
    }
    return dp[word1.length()][word2.length()];
}
}
```

Your previous code was restored from your local storage. Reset to default