75 Days of Code
Day 66
Problem no : Leetcode72.
Problem Title Edit Distance
Problem type : DP

Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2.

You have the following three operations permitted on a word:

Insert a character
Delete a character
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')

Constraints:

0 <= word1.length, word2.length <= 500
word1 and word2 consist of lowercase English letters.

```typescript
function minDistance(word1: string, word2: string): number {
    if ((word1 && !word2) || (!word1 && word2)) {
        return word1 ? word1.length : word2.length;
    }

    let word1Length = word1.length;
    let word2Length = word2.length;
    const dp = new Array(word1Length + 1).fill(0).map(() => new Array(word2Length + 1).fill(0));

    for (let word1Index = 0; word1Index <= word1Length; word1Index++) {
        dp[word1Index][0] = word1Index;
    }
    for (let word2Index = 0; word2Index <= word2Length; word2Index++) {
        dp[0][word2Index] = word2Index;
    }

    for (let row = 1; row <= word1Length; row++) {
        for (let col = 1; col <= word2Length; col++) {
            if (word1[row - 1] === word2[col - 1]) {
                dp[row][col] = dp[row - 1][col - 1];
            } else {
                dp[row][col] = 1 + Math.min(dp[row - 1][col], dp[row][col - 1], dp[row - 1][col - 1]);
            }
        }
    }
    return dp[word1Length][word2Length];
};
```

Accepted

Edit

Runtime

Details

79 ms

Beats 66.91% of users with TypeScript

Memory

49.26 MB

Beats 42.65% of users with TypeScript

More challenges