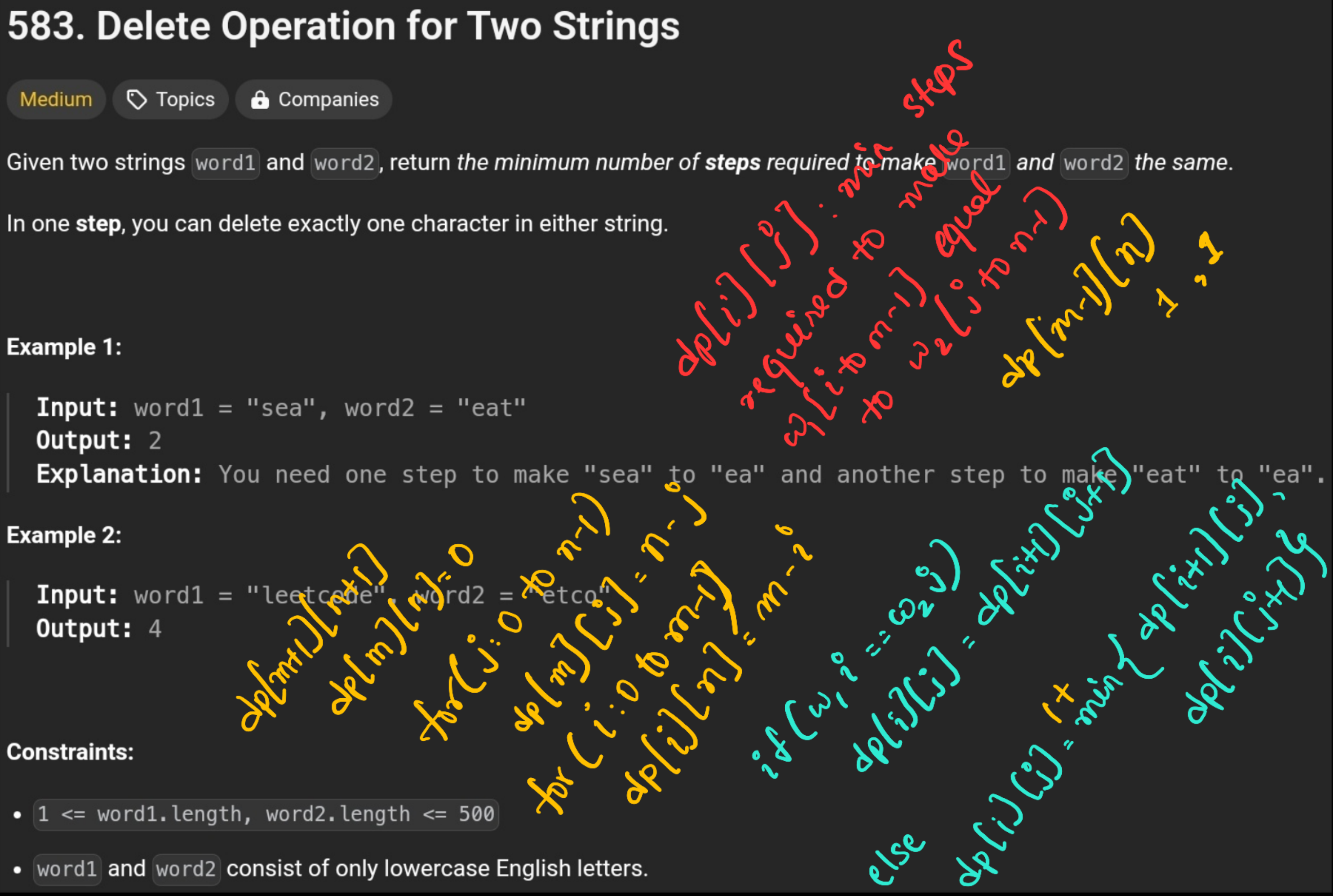


583. Delete Operation for Two Strings

Medium · Topics · Companies

Given two strings `word1` and `word2`, return *the minimum number of* ***steps*** *required to make* `word1` *and* `word2` *the same.*

In one **step**, you can delete exactly one character in either string.

Example 1:

```
Input: word1 = "sea", word2 = "eat"
Output: 2
Explanation: You need one step to make "sea" to "ea" and another step to make "eat" to "ea".
```

Example 2:

```
Input: word1 = "leetcode", word2 = "etco"
Output: 4
```

Constraints:

- `1 <= word1.length, word2.length <= 500`
- `word1` and `word2` consist of only lowercase English letters.

dp[i][j] : min steps required to make w_1[i to m-1] equal to w_2[j to n-1]

dp[m-1][n]

1 , 1

```
dp[m+1][n+1]
dp[m][n] = 0
for(j: 0 to n-1)
    dp[m][j] = n-j
for(i: 0 to m-1)
    dp[i][n] = m-i
```

```
if(w1i == w2j)
    dp[i][j] = dp[i+1][j+1]
else
    dp[i][j] = 1 + min{ dp[i+1][j],
                        dp[i][j+1] }
```

To make the minimum deletion operations, we should not delete the characters that are same in both words.

Get the LCS of word1 and word2

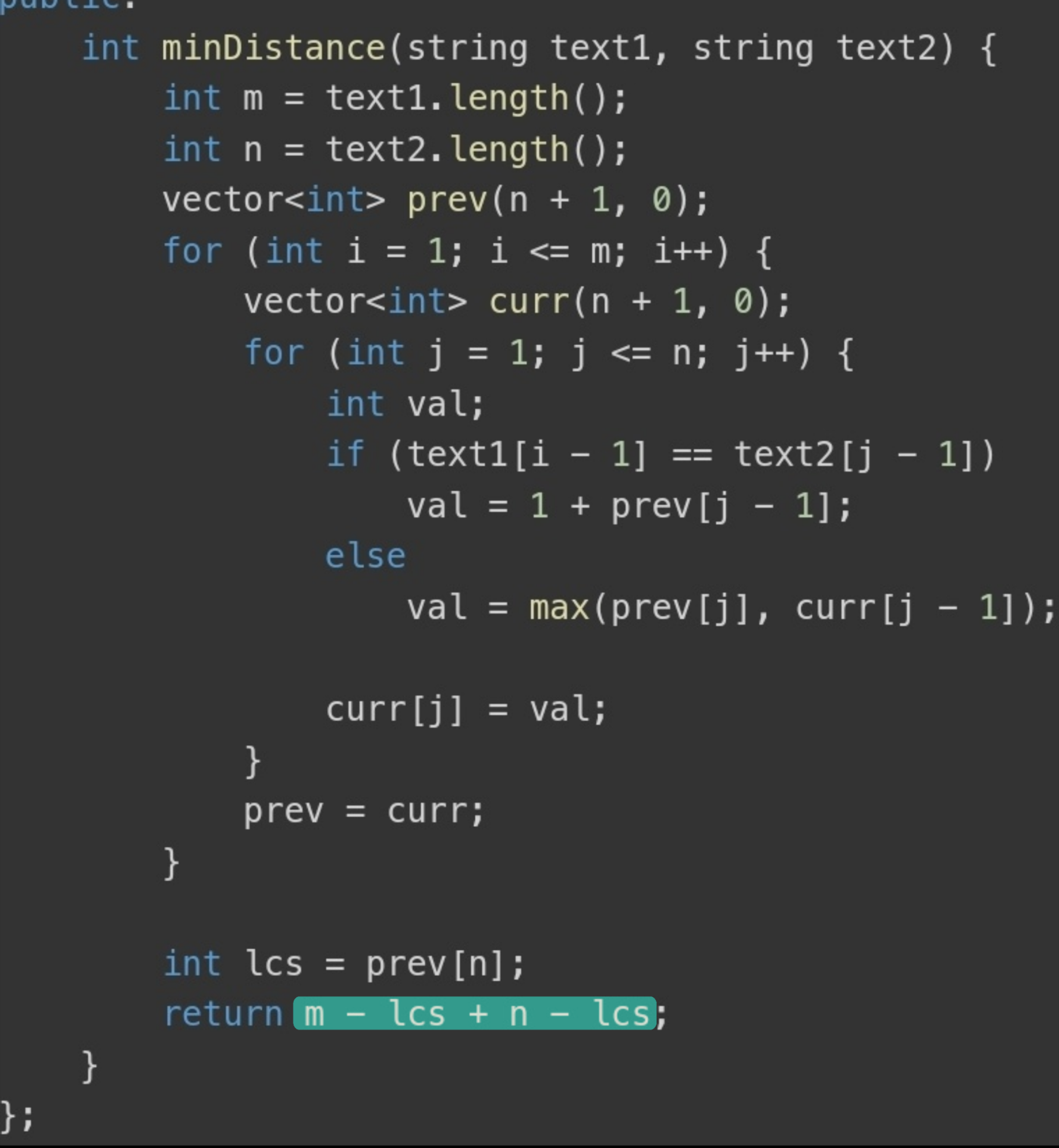
Then

deletions required = m - LCS + n - LCS

i.e. Now the problem got reduced to problem of finding LCS. # 1143

T(n): T(n) of LCS
S(n): S(n) of LCS

```
class Solution {
public:
```



```cpp
public:
    int minDistance(string text1, string text2) {
        int m = text1.length();
        int n = text2.length();
        vector<int> prev(n + 1, 0);
        for (int i = 1; i <= m; i++) {
            vector<int> curr(n + 1, 0);
            for (int j = 1; j <= n; j++) {
                int val;
                if (text1[i - 1] == text2[j - 1])
                    val = 1 + prev[j - 1];
                else
                    val = max(prev[j], curr[j - 1]);

                curr[j] = val;
            }
            prev = curr;
        }

        int lcs = prev[n];
        return m - lcs + n - lcs;
    }
};
```