5. Edit Distance (Leetcode-72)

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

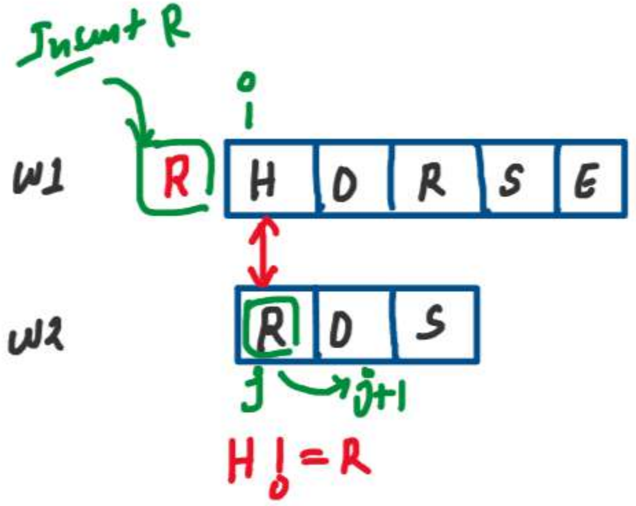
You have the following three operations permitted on a word:
1. Insert a character
2. Delete a character
3. Replace a character

Example

WORD1 = HORSE
WORD2 = ROS

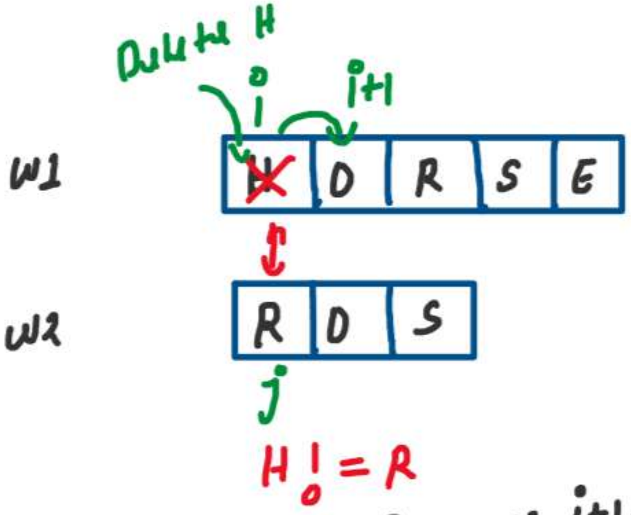
Output 3

Insert



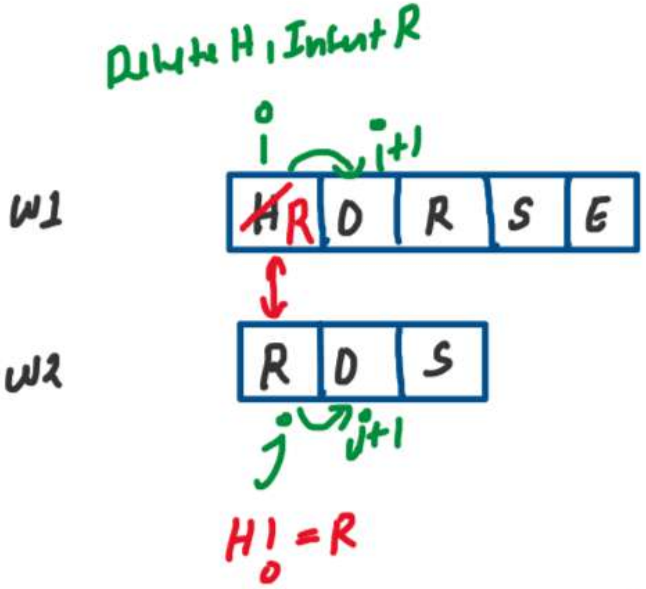
$$\text{insertOpt} = \underset{\underset{\text{1st step}}{\uparrow}}{1} + f(w1,\underset{\underset{\text{R.C.}}{\uparrow}}{\ } w2, i, j+1);$$

Delete



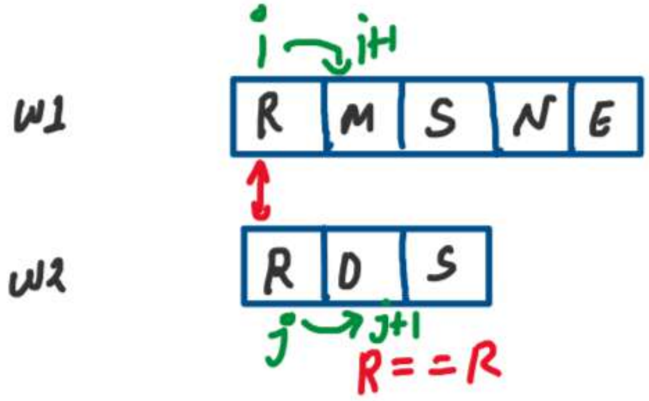
$$\text{deleteOpt} = 1 + f(w1, w2, i+1, j)$$

Replace



replaceOpt = 1 + f(w1, w2, i+1, j+1);

Match case

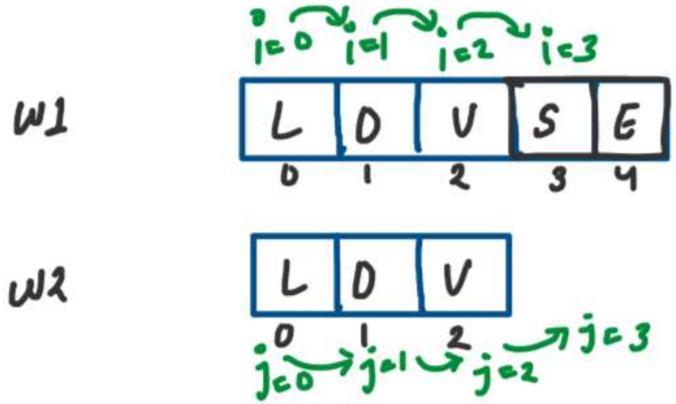


⇒ 0 + f(w1, w2, i+1, j+1)

output

return(min(insert , Replace, Delete))

BASE CASE



size of w1 = 5   size of w2 = 3

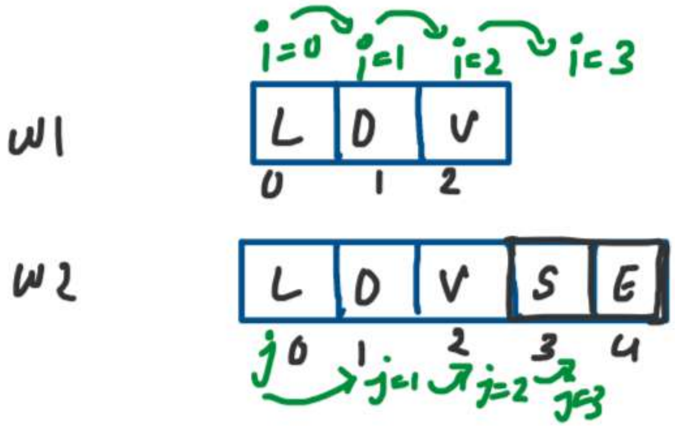
$(j >= 3)$

return $5 - i$

$= 5 - 3$

$= 2$

Total operation
2
for S & E



size of w1 = 3   size of w2 = 5

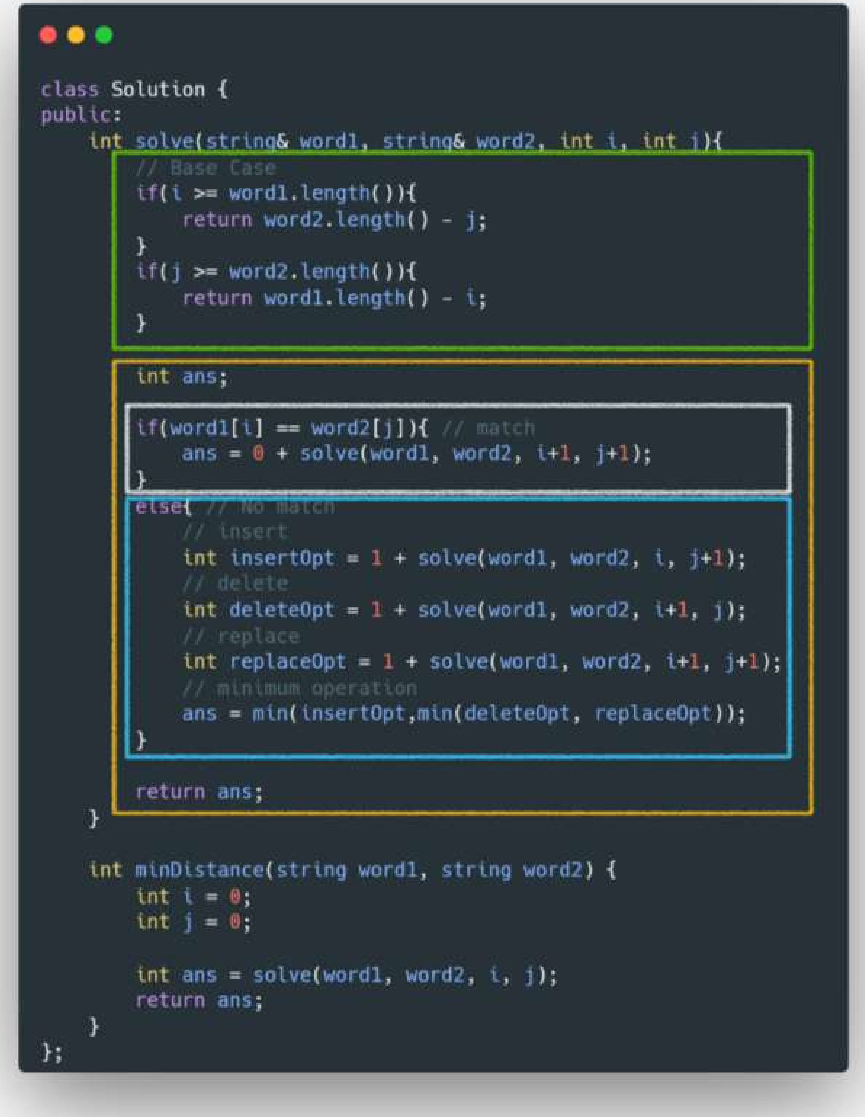
$(i >= 3)$

return $5 - i$

$= 5 - 3$

$= 2$

Total operation
2
for S & E



```
class Solution {
public:
    int solve(string& word1, string& word2, int i, int j){
        // Base Case
        if(i >= word1.length()){
            return word2.length() - j;
        }
        if(j >= word2.length()){
            return word1.length() - i;
        }

        int ans;

        if(word1[i] == word2[j]){ // match
            ans = 0 + solve(word1, word2, i+1, j+1);
        }
        else{ // No match
            // insert
            int insertOpt = 1 + solve(word1, word2, i, j+1);
            // delete
            int deleteOpt = 1 + solve(word1, word2, i+1, j);
            // replace
            int replaceOpt = 1 + solve(word1, word2, i+1, j+1);
            // minimum operation
            ans = min(insertOpt,min(deleteOpt, replaceOpt));
        }

        return ans;
    }

    int minDistance(string word1, string word2) {
        int i = 0;
        int j = 0;

        int ans = solve(word1, word2, i, j);
        return ans;
    }
};
```

T.C. = ?

S.C. = ?

DRY RUN = ?