I：Insert a character

D：Delete a character

R：Replace a character

動態規劃：

horse to rose

horse -> rorse　(replace 'h' with 'r')

rorse -> rose　(delete 'r')

rose -> ros　(delete 'e')

注意：最少 operators 不一定為唯一解

	0 Ø	1 r	2 o	3 s
0 Ø	0	Insert 1	Insert 2	Insert 3
1 h	Delete 1	Replace 1	Insert 2	Insert 2
2 o	Delete 2	Replace 2	2	Insert 3
3 r	Delete 3	Delete 3	Delete 2	Replace 3
4 s	Delete 4	Delete 4	Delete 3	2
5 e	Delete 5	Delete 5	Delete 4	Delete 3

規則：

當 row = col，

1. 遇到相同字元：不做事，繼承左上 operators
2. 遇到左上 operators 最小，進行 Replace，operators 等於左上的 operators+1
3. 遇到左側的 operators 最小，進行 Insert，operators 等於左側的 operators+1
4. 遇到上方的 operators 最小，進行 Delete，operators 等於上側的 operators+1

	0 Ø	1 e	2 a	3 t
0 Ø	0	Insert 1	Insert 2	Insert 3
1 s	Delete 1	Replace 1	Insert 2	Insert 3
2 e	Delete 2	1	Replace 2	Replace 3
3 a	Delete 3	Delete 2	1	Insert 2

Algorithm Edit Distance

Input： (string word1, string word2)

Output： min operators that convert word1 to word2

```
1. operators = int[word1.length() + 1][word2.length() + 1]
2. for(int row=0;  row < word1.length() + 1;  row++){
3.     operators[row][0] = row;
4. }
5. for(int col = 0;  col < word2.length() + 1;  col++){
6.     operators[0][col] = col;
7. }
8. for(int row=1;  row < word1.length() + 1;  row++){
9.     for(int col=1;  col<word2.length() + 1;  col++){
10.        if(word1[row] == word2[col]){
11.            operators[row][col] = operators[row-1][col-1];
12.        }
13.        else{
14.            operators[row][col] = min(operators[row-1][col-1], operators[row-
    1][col], operators[row][col-1)] + 1;
15.        }
16.    }
17.}
18. return operators[word1.length()][world2.length()];
```