

Count Substring of 0 and 1

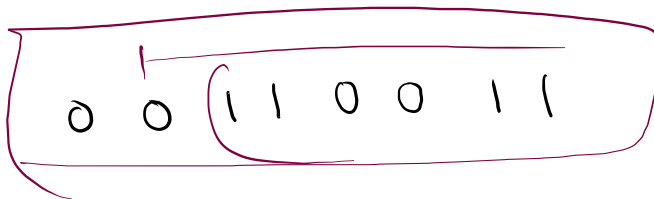
Sample Input 0

00110011

Sample Output 0

6

8 →



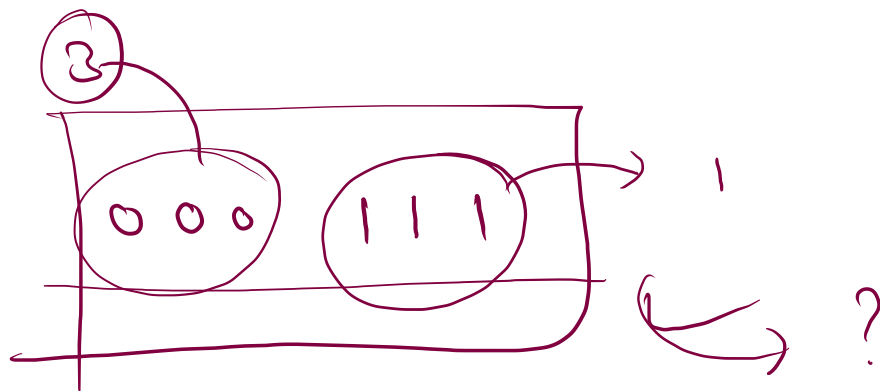
$$\frac{f(9)}{2} = 36$$

non-empty
or
 $\text{len} \geq 1$

0
0 0
0 0 1
0 0 1 1
0 0 1 1 0
0 0 1 1 0 0
0 0 1 1 0 0 1
0 0 1 1 0 0 1 1

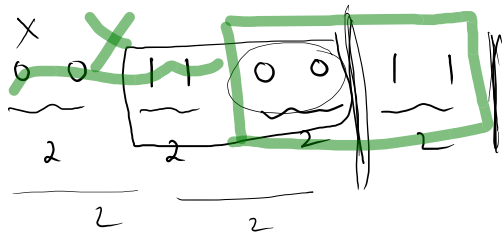
0
0 1
0 1 1
0 1 1 0
0 1 1 0 0
0 1 1 0 0 1
0 1 1 0 0 1 1

?
1 1 0 0
0 0 1 1



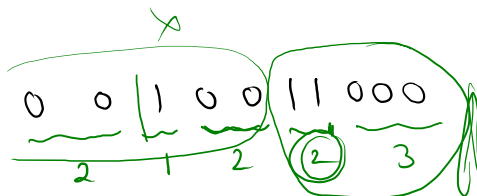
3

$$\begin{matrix} \approx \\ \textcircled{4} \end{matrix} + 2 = \textcircled{6}$$



$$c = \cancel{7} \cancel{4} \underline{6}$$

01
10
0011
01



min



3
=

2
=



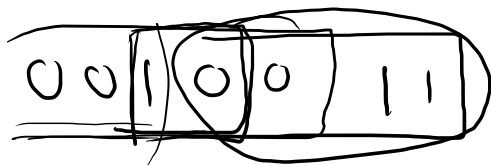
$$\min(3, 2)$$

$$= \textcircled{2}$$

$p = \cancel{0} \cancel{1} \cancel{2}$

0 0 1 0 0 1 1 0 0
~~2~~ ~~2~~ ~~1~~ 1

$c = \cancel{1} \cancel{2} \cancel{1} \cancel{2} \cancel{1} \cancel{2} \cancel{1}$



$s[i] == s[i-1] \rightarrow c++$

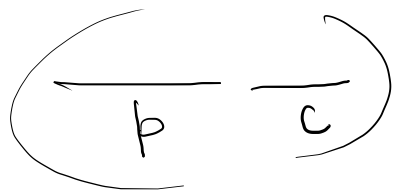
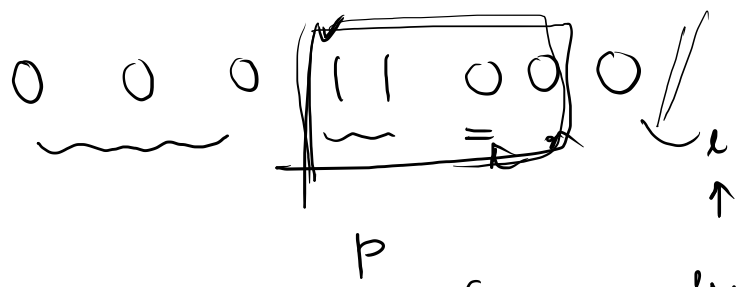
$ans = \cancel{0} \cancel{0} \cancel{1} \cancel{2} \cancel{1}$

$s[i] \neq s[i-1] \rightarrow \underline{\text{evaluate.}}$

$\min(p, c)$
 $(0, 2)$

$p = \emptyset \cancel{1} 2$
 $c = \cancel{1} \underline{2} \cancel{3} \cancel{4} 3$

$ans = \cancel{0} \cancel{0} \cancel{2} \underline{4}$



$(100, 10)$

$\min(p, c) = 2$

$\cancel{0} \cancel{0} \cancel{0} \cancel{0} \cancel{0} \cancel{1} \cancel{1}$
 $5, (2)$

```

public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String s = scn.next();

    int ans = 0;
    int prev = 0;
    int curr = 1;

    for(int i = 1; i < s.length(); i++){
        if(s.charAt(i) == s.charAt(i-1)){
            curr++;
        }
        else{
            //evaluate
            ans += Math.min(prev, curr);
            prev = curr;
            curr = 1;
        }
    }

    ans += Math.min(prev, curr);

    System.out.println(ans);
}

```

ans = ~~0~~ 7 (4)

p = 2

c = 1 2 3

~~~~~~~~~

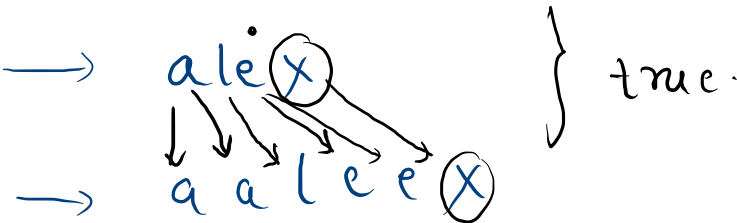
p, c = 2

# 925. Long Pressed Name

Easy 2230 320 Add to List Share

Your friend is typing his name into a keyboard. Sometimes, when typing a character c, the key might get long pressed, and the character will be typed 1 or more times.

You examine the typed characters of the keyboard. Return True if it is possible that it was your friend's name, with some characters (possibly none) being long pressed.



```
name = "alex", typed = "aaleex"
```

a lex (F)

↓ x ↓  
a pex

a lex (T)

|||  
a a l e e x

name a l l e x (F)

typed ↓ ↓ ↓ x  
a l e x

a lex (T)

a l e x x x x



$n \rightarrow$  alex  
 $t \rightarrow$  alex  
 (F)

$n \rightarrow$  moon  
 $t \rightarrow$  mon  
 (T)

$n \rightarrow$  alex (T)  
 $t \rightarrow$  a l l l l l e c x x  
 (F)

$n \rightarrow$  alex (T)  
 $t \rightarrow$  alex  
 (F)

$n \rightarrow$  alex (F)  
 $t \rightarrow$  alex  
 (F)

alex  
alex

ch from 'n' can map with multiple  
 ch from 't'

& oppsite is not possible.

name ,      typed.  
↓                    ↓  
n                    m

alex  
|||  
ale ? ✓

n > m → not ok ,

m > n ✓ ok

i i ✓  
a lex  
3  
a l l l l  
j j j j j x

Handwritten diagram illustrating the recursive step of the Longest Common Subsequence (LCS) algorithm. It shows two strings, "a l e x" and "a l e x x x x". The first string has "a" and "l" underlined. The second string has "a" and "l" underlined, and the last four "x"s grouped together in a circle. Arrows indicate the recursive calls: one from the "a" in the first string to the "a" in the second string, and another from the "l" in the first string to the "l" in the second string. A blue arrow points from the bottom left to the text  $n[i] == x$ , which is written in red.

$$n[i-1] == t[j] \quad \{ j++ \}$$

$n \rightarrow$ 

$$\begin{array}{c}
 i \\
 a \text{ lex} \\
 \swarrow \searrow \\
 a \text{ alx} \\
 j
 \end{array}
 \rightsquigarrow \text{false.}$$

a l l e x i

a l l l l l e x

j

Critical.

No.1

$n \rightarrow a lex^{i \leftarrow i}$

$t \rightarrow a lex xxx f$   
 $j$

ij →

alex<sup>i</sup>  
alex<sup>j</sup>x

Ex = T

```
//j is available and i ended  
while(j < m){  
    if(name.charAt(i-1) == typed.charAt(j)){  
        j++;  
    }  
    else{  
        return false;  
    }  
}
```

else →

alex<sup>i</sup>  
alex<sup>j</sup>xf<sub>i</sub>

Ex = F

imp.

$i$   $\rightarrow$  should end.

$\rightarrow i == n$

$n \rightarrow$   $i$   
alex

$t \rightarrow$  all l l  
false.

$j$  ended  $i$  remaining.

$i == n$

$n \rightarrow$

$n-1$   $i$   
a l e x

$\rightarrow$

a l l l l

i should end.

j

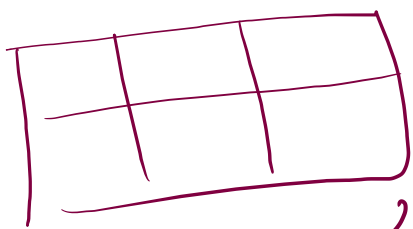
$i == n$   
T

✓  
a l e x  
aman



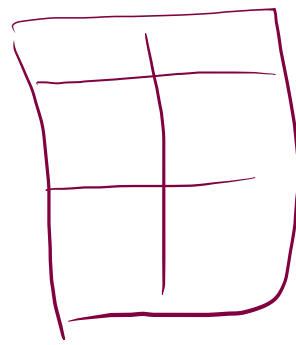
# Matrix Multiplication.

A



$$\underline{2 \times 3}$$

B



$$\underline{3 \times 2}$$

$$C = 2 \times 2$$

A

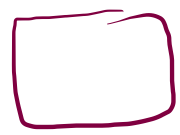
B

$$\underline{2 \times 3}$$

$$\underline{2 \times 3}$$



C



$$2 \times 3 \checkmark$$

$$A = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad m \times n$$

$$B = \begin{bmatrix} 3 & 0 & 2 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix} \quad p \times q$$

$$1 \times 3 + 0 \times 2 + 2 \times 2 = 7$$

$$1 \times 3 + 2 \times 1 + 1 \times 2 = 7$$

$$1 \times 0 + 0 \times 2 + 2 \times 3 = 6$$

$$1 \times 0 + 2 \times 2 + 3 \times 1 = 7$$

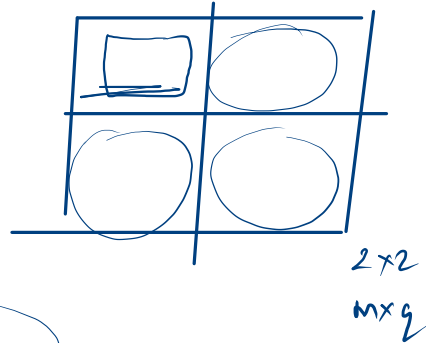
$$C = \begin{bmatrix} 7 & 6 & 7 \end{bmatrix} \quad m \times q$$

$$A[0][0] \times B[0][0] + A[0][1] \times B[1][0] + A[0][2] \times B[2][0]$$

```

for(int i = 0; i < m; i++){
    for(int j = 0; j < q; j++){
        for(int k = 0; k < n; k++){ // k < p
            C[i][j] += A[i][k] * B[k][j];
        }
    }
}

```



$$i=0$$

$$j=0$$

$$k=0 \vee 2$$

$$k < 3$$

$$+ A[0][0] \times B[0][0]$$

$$+ A[0][1] \times B[1][0]$$

$$+ A[0][2] \times B[2][0]$$