

# 【华为OD机考 统一考试机试C卷】 分割均衡字符串（C++ Java JavaScript Python C语言）

## 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，**C卷真题已基本整理完毕**

抽到原题的概率为2/3到3/3，**也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。**

另外订阅专栏还可以联系笔者开通在线 **OJ** 进行刷题，提高刷题效率。

**真题目录：**华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明

**专栏：**2023华为OD机试( B卷+C卷+D卷) (C++JavaJSPy)

**华为OD面试真题精选：**华为OD面试真题精选

**在线OJ：**点击立即刷题，模拟真实机考环境

## 题目描述

均衡串定义: 字符串只包含两种字符，且两种字符的个数相同。

给定一个均衡字符串，请给出可分割成新的均衡子串的最大个数。

约定字符串中只包含大写的X和Y两种字符。

## 输入描述

均衡串: XXYYXY

字符串的长度[2,100001]。给定的字符串均为均衡串

## 输出描述

可分割为两个子串:

XXYY

XY

备注  
分割后的子串，是原字符串的连续子串。

用例

输入	XXYYXY
输出	2
说明	无

解题思路

原题：<https://leetcode.cn/problems/split-a-string-in-balanced-strings/description/>

这段代码的解题思路如下：

1. 初始化变量 `ans` 为0，用于记录可分割成新的均衡子串的最大个数。
2. 初始化变量 `count` 为0，`count`来记录'X'和'Y'的差值即可。当`count`为0时，表示当前位置可以作为分割点，将`ans`加1。
3. 使用for循环遍历字符串 `s` 的每个字符。
4. 在循环中，判断当前字符是'X'还是'Y':
  - 如果是'X'，则将 `count` 加1，表示出现了一个'X'。
  - 如果是'Y'，则将 `count` 减1，表示出现了一个'Y'。
5. 在每次更新 `count` 后，判断 `count` 是否为0：
  - 如果为0，表示当前位置可以作为分割点，将 `ans` 加1。
6. 循环结束后，输出 `ans`，即可分割成新的均衡子串的最大个数。

C++

```
1 | #include <iostream>
2 | #include <string>
3 | using namespace std;
4 |
5 | int main() {
6 |     // 创建一个字符串变量，用于存储用户输入的字符串
7 | }
```

```

7
8 string s;
9 // 从标准输入中读取一行字符串, 并将其存储到变量s中
10 getline(cin, s);
11
12 // 初始化变量, 用于记录可分割成新的均衡子串的最大个数
13 int ans = 0;
14 // 初始化变量, 用于记录当前位置字符'X'和'Y'的差值
15 int count = 0;
16
17 // 遍历字符串的每个字符
18 for (int i = 0; i < s.length(); i++) {
19     // 判断当前字符是'X'还是'Y'
20     if (s[i] == 'X') {
21         // 如果是'X', 则将count加1, 表示出现了一个'X'
22         count++;
23     } else {
24         // 如果是'Y', 则将count减1, 表示出现了一个'Y'
25         count--;
26     }
27
28     // 在每次更新count后, 判断count是否为0
29     if (count == 0) {
30         // 如果为0, 表示当前位置可以作为分割点, 将ans加1
31         ans++;
32     }
33 }
34
35 // 输出可分割成新的均衡子串的最大个数
36 cout << ans << endl;
37
38 return 0;
39 }

```

## Java

```

1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6     }
7 }

```

```

6
7
8 // 获取用户输入的字符串
9 String s = sc.nextLine();
10
11 // 初始化变量, 用于记录可分割成新的均衡子串的最大个数
12 int ans = 0;
13 // 初始化变量, 用于记录当前位置字符'X'和'Y'的差值
14 int count = 0;
15
16 // 遍历字符串的每个字符
17 for (int i = 0; i < s.length(); i++) {
18     // 判断当前字符是'X'还是'Y'
19     if (s.charAt(i) == 'X') {
20         // 如果是'X', 则将count加1, 表示出现了一个'X'
21         count++;
22     } else {
23         // 如果是'Y', 则将count减1, 表示出现了一个'Y'
24         count--;
25     }
26
27     // 在每次更新count后, 判断count是否为0
28     if (count == 0) {
29         // 如果为0, 表示当前位置可以作为分割点, 将ans加1
30         ans++;
31     }
32 }
33
34 // 输出可分割成新的均衡子串的最大个数
35 System.out.println(ans);
36 }

```

## JavaScript

```

1 const readline = require('readline');
2
3 const rl = readline.createInterface({
4     input: process.stdin,
5     output: process.stdout
6 });

```

```

/
8
9 rl.on('line', (s) => {
10 // 初始化变量, 用于记录可分割成新的均衡子串的最大个数
11 let ans = 0;
12 // 初始化变量, 用于记录当前位置字符'x'和'y'的差值
13 let count = 0;
14
15 // 遍历字符串的每个字符
16 for (let i = 0; i < s.length; i++) {
17 // 判断当前字符是'x'还是'y'
18 if (s.charAt(i) === 'x') {
19 // 如果是'x', 则将count加1, 表示出现了一个'x'
20 count++;
21 } else {
22 // 如果是'y', 则将count减1, 表示出现了一个'y'
23 count--;
24 }
25
26 // 在每次更新count后, 判断count是否为0
27 if (count === 0) {
28 // 如果为0, 表示当前位置可以作为分割点, 将ans加1
29 ans++;
30 }
31 }
32
33 // 输出可分割成新的均衡子串的最大个数
34 console.log(ans);
35
36 rl.close();
37 });

```

## Python

```

1 # 从标准输入中读取用户输入的字符串
2 s = input("")
3
4 # 初始化变量, 用于记录可分割成新的均衡子串的最大个数
5 ans = 0
6 # 初始化变量, 用于记录当前位置字符'x'和'y'的差值
7 count = 0
8

```

```

8
9 # 遍历字符串的每个字符
10 for char in s:
11     # 判断当前字符是'X'还是'Y'
12     if char == 'X':
13         # 如果是'X', 则将count加1, 表示出现了一个'X'
14         count += 1
15     else:
16         # 如果是'Y', 则将count减1, 表示出现了一个'Y'
17         count -= 1
18
19 # 在每次更新count后, 判断count是否为0
20 if count == 0:
21     # 如果为0, 表示当前位置可以作为分割点, 将ans加1
22     ans += 1
23
24 # 输出可分割成新的均衡子串的最大个数
25 print(ans)

```

## C语言

```

1 #include <stdio.h>
2 #include <string.h>
3
4 // 定义一个函数用于计算平衡字符串的数量
5 int balancedStringSplit(const char *s) {
6     int n = strlen(s); // 获取字符串的长度
7     int ans = 0; // 初始化结果为0
8     // 遍历字符串
9     for (int i = 0; i < n; ) {
10         int j = i + 1, score = s[i] == 'X' ? 1 : -1; // 初始化j为i+1, score为s[i]等于'X'时为1, 否则为-1
11         // 当j小于n且score不等于0时, 继续循环, score的值为s[j++]等于'X'时加1, 否则减1
12         while (j < n && score != 0) score += s[j++] == 'X' ? 1 : -1;
13         i = j; // 将j的值赋给i
14         ans++; // 结果加1
15     }
16     return ans; // 返回结果
17 }
18
19 int main() {
20

```

```
20     char s[100002]; // 定义一个字符串变量
21     gets(s); // 从标准输入读取一行字符串
22
23     int result = balancedStringSplit(s); // 调用函数处理字符串, 并将结果赋值给result
24     printf("%d\n", result); // 输出结果
25
26     return 0;
27 }
```

## 完整用例

### 用例1

```
1 | XXXXXY
```

### 用例2

```
1 | XXXYXXYY
```

### 用例3

```
1 | XXXYXXYX
```

### 用例4

```
1 | XXXYXXYYXY
```

### 用例5

```
1 | XYYXXY
```

### 用例6

```
1 | XY
```

### 用例7

1 | XXY

用例8

1 | XXXY

用例9

1 | XYXY

用例10

1 | YXYXY

文章目录

- 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷
  - 题目描述
  - 输入描述
  - 输出描述
  - 用例
  - 解题思路
  - C++
  - Java
  - javaScript
  - Python
  - C语言
  - 完整用例
    - 用例1
    - 用例2
    - 用例3
    - 用例4
    - 用例5
    - 用例6
    - 用例7



用例8  
用例9  
用例10

# 机考真题 华为OD



CSDN @算法大师