【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++

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  // 创建一个字符串变量，用于存储用户输入的字符串

```

```
    string s;
    // 从标准输入中读取一行字符串，并将其存储到变量s中
    getline(cin, s);

    // 初始化变量，用于记录可分割成新的均衡子串的最大个数
    int ans = 0;
    // 初始化变量，用于记录当前位置字符'X'和'Y'的差值
    int count = 0;

    // 遍历字符串的每个字符
    for (int i = 0; i < s.length(); i++) {
        // 判断当前字符是'X'还是'Y'
        if (s[i] == 'X') {
            // 如果是'X'，则将count加1，表示出现了一个'X'
            count++;
        } else {
            // 如果是'Y'，则将count减1，表示出现了一个'Y'
            count--;
        }

        // 在每次更新count后，判断count是否为0
        if (count == 0) {
            // 如果为0，表示当前位置可以作为分割点，将ans加1
            ans++;
        }
    }

    // 输出可分割成新的均衡子串的最大个数
    cout << ans << endl;

    return 0;
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        // 获取用户输入的字符串
        String s = sc.nextLine();

        // 初始化变量，用于记录可分割成新的均衡子串的最大个数
        int ans = 0;
        // 初始化变量，用于记录当前位置字符'X'和'Y'的差值
        int count = 0;

        // 遍历字符串的每个字符
        for (int i = 0; i < s.length(); i++) {
            // 判断当前字符是'X'还是'Y'
            if (s.charAt(i) == 'X') {
                // 如果是'X'，则将count加1，表示出现了一个'X'
                count++;
            } else {
                // 如果是'Y'，则将count减1，表示出现了一个'Y'
                count--;
            }

            // 在每次更新count后，判断count是否为0
            if (count == 0) {
                // 如果为0，表示当前位置可以作为分割点，将ans加1
                ans++;
            }
        }

        // 输出可分割成新的均衡子串的最大个数
        System.out.println(ans);
    }
}
```

javaScript

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
```

```
rl.on('line', (s) => {
  // 初始化变量，用于记录可分割成新的均衡子串的最大个数
  let ans = 0;
  // 初始化变量，用于记录当前位置字符'X'和'Y'的差值
  let count = 0;

  // 遍历字符串的每个字符
  for (let i = 0; i < s.length; i++) {
    // 判断当前字符是'X'还是'Y'
    if (s.charAt(i) === 'X') {
      // 如果是'X'，则将count加1，表示出现了一个'X'
      count++;
    } else {
      // 如果是'Y'，则将count减1，表示出现了一个'Y'
      count--;
    }

    // 在每次更新count后，判断count是否为0
    if (count === 0) {
      // 如果为0，表示当前位置可以作为分割点，将ans加1
      ans++;
    }
  }

  // 输出可分割成新的均衡子串的最大个数
  console.log(ans);

  rl.close();
});
```

Python

```
# 从标准输入中读取用户输入的字符串
s = input("")

# 初始化变量，用于记录可分割成新的均衡子串的最大个数
ans = 0
# 初始化变量，用于记录当前位置字符'X'和'Y'的差值
count = 0
```

```
# 遍历字符串的每个字符
for char in s:
  # 判断当前字符是'X'还是'Y'
  if char == 'X':
    # 如果是'X'，则将count加1，表示出现了一个'X'
    count += 1
  else:
    # 如果是'Y'，则将count减1，表示出现了一个'Y'
    count -= 1

  # 在每次更新count后，判断count是否为0
  if count == 0:
    # 如果为0，表示当前位置可以作为分割点，将ans加1
    ans += 1

# 输出可分割成新的均衡子串的最大个数
print(ans)
```

C语言

```
#include <stdio.h>
#include <string.h>

// 定义一个函数用于计算平衡字符串的数量
int balancedStringSplit(const char *s) {
    int n = strlen(s); // 获取字符串的长度
    int ans = 0; // 初始化结果为0
    // 遍历字符串
    for (int i = 0; i < n; ) {
        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
        // 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
        while (j < n && score != 0) score += s[j++] == 'X' ? 1 : -1;
        i = j; // 将j的值赋给i
        ans++; // 结果加1
    }
    return ans; // 返回结果
}

int main() {
```

```
    char s[100002]; // 定义一个字符串变量
    gets(s); // 从标准输入读取一行字符串

    int result = balancedStringSplit(s); // 调用函数处理字符串，并将结果赋值给result
    printf("%d\n", result); // 输出结果

    return 0;
}
```

完整用例

用例1

```
XXYYXY
```

用例2

```
XYXYXYXY
```

用例3

```
XXYXYYX
```

用例4

```
XXYXYYXXY
```

用例5

```
XYYXXY
```

用例6

```
XY
```

用例7

```
XXYY
```

用例8

```
XXXYYY
```

用例9

```
XYXYXY
```

用例10

```
YXXYXYY
```

文章目录

用例8
用例9
用例10