

题目描述

在第一人称射击游戏中，玩家通过键盘的A、S、D、W四个按键控制游戏人物分别向左、向后、向右、向前进行移动，从而完成走位。

假设玩家每按动一次键盘，游戏任务会向某个方向移动一步，如果玩家在操作一定次数的键盘并且各个方向的步数相同时，此时游戏任务必定会回到原点，则称此次走位为完美走位。

现给定玩家的走位（例如：ASDA），请通过更换其中一段连续走位的方式使得原走位能够变成一个完美走位。其中待更换的连续走位可以是相同长度的任何走位。

请返回待更换的连续走位的最小可能长度。

如果原走位本身是一个完美走位，则返回0。

输入描述

输入为由键盘字母表示的走位s，例如：ASDA

输出描述

输出为待更换的连续走位的最小可能长度。

用例

输入	WASDAASD
输出	1
说明	将第二个A替换为W，即可得到完美走位

输入	AAAA
输出	3
说明	将其中三个连续的A替换为WSD，即可得到完美走位

题目解析

题目要求，保持W,A,S,D字母个数平衡，即相等，如果不相等，可以从字符串中选取一段连续子串替换，来让字符串平衡。

比如：WWWWAAAASSSS

字符串长度12，W,A,S,D平衡的话，则每个字母个数应该是3个，而现在W,A,S各有4个，也就是说各超了1个。

因此我们应该从字符串中，选取一段包含1个W，1个A，1个S的子串，来替换为D。

WWWWAAAASSSS

WWWWWAAAASSSS

WWWWAAAASSSS

.....

WWWWAAAASSSS

而符合这种要求的子串可能很多，我们需要找出其中最短的，即WAAAAA。

本题其实就是求最小覆盖子串，同[LeetCode - 76 最小覆盖子串\\_伏城之外的博客-CSDN博客](#)

题目解析请看上面链接博客。

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
```

 伏城之外 已关注  2   9   6  专栏目录  已订阅

## JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  rl.on("line", (line) => {
10    console.log(getResult(line));
11  });
12
13  function getResult(str) {
14    // 此时count记录统计W,A,S,D字母的数量
15    const count = {
16      W: 0,
17      A: 0,
18      S: 0,
19      D: 0,
20    };
21
22    for (let c of str) count[c]++;
23
24    // 平衡状态时, W,A,S,D应该都是avg数量
25    const avg = str.length / 4;
26
27    let total = 0; // total用于记录多余字母个数
28
29    let flag = true; // flag表示当前是否为平衡状态, 默认是
30    for (let c in count) {
31      if (count[c] > avg) {
32        flag = false; // 如果有一个字母数量超标, 则平衡打破
33        count[c] -= avg; // 此时count记录每个字母超过avg的数量
34        total += count[c];
35      } else {
36        delete count[c];
37      }
38    }
39
40    if (flag) return 0; // 如果平衡, 则输出0
41
42    let i = 0;
43    let j = 0;
44    let minLen = str.length + 1;
45
46    while (j < str.length) {
47      const jc = str[j];
48
49      if (count[jc]-- > 0) {
50        total--;
51      }
52
53      while (total === 0) {
54        minLen = Math.min(minLen, j - i + 1);
55
56        const ic = str[i];
57        if (count[ic]++ >= 0) {
58          total++;
59        }
60
61        i++;
62      }
63
64      j++;
65    }
66
67    return minLen;
68  }
```

## Java算法源码

```
1  import java.util.HashMap;
2  import java.util.Scanner;
3
4  public class Main {
5    public static void main(String[] args) {
6      Scanner sc = new Scanner(System.in);
7      System.out.println(getResult(sc.next()));
8    }
9
10   public static int getResult(String str) {
11     // count用于记录W,A,S,D字母的数量
12     HashMap<Character, Integer> count = new HashMap<>();
13
14     for (int i = 0; i < str.length(); i++) {
15       Character c = str.charAt(i);
16       count.put(c, count.getOrDefault(c, 0) + 1);
17     }
18
19     // 平衡状态时, W,A,S,D应该都是avg数量
20     let avg = str.length() / 4;
```

## Java算法源码

```
1 import java.util.HashMap;
2 import java.util.Scanner;
3
4 public class Main {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         System.out.println(getResult(sc.next()));
8     }
9
10    public static int getResult(String str) {
11        // count用于记录W,A,S,D字母的数量
12        HashMap<Character, Integer> count = new HashMap<>();
13
14        for (int i = 0; i < str.length(); i++) {
15            Character c = str.charAt(i);
16            count.put(c, count.getOrDefault(c, 0) + 1);
17        }
18
19        // 平衡状态时, W,A,S,D应该都是avg数量
20        int avg = str.length() / 4;
21
22        // total用于记录多余字母个数
23        int total = 0;
24
25        // flag表示当前是否为平衡状态, 默认是
26        boolean flag = true;
27
28        for (Character c : count.keySet()) {
29            if (count.get(c) > avg) {
30                // 如果有一个字母数量超标, 则平衡打破
31                flag = false;
32                // 此时count记录每个字母超过avg的数量
33                count.put(c, count.get(c) - avg);
34                total += count.get(c);
35            } else {
36                count.put(c, 0); // 此时count统计的其实是多余字母, 如果没有超过avg, 则表示没有多余字母
37            }
38        }
39
40        // 如果平衡, 则输出0
41        if (flag) return 0;
42
43        int i = 0;
44        int j = 0;
45        int minLen = str.length() + 1;
46
47        while (j < str.length()) {
48            Character jc = str.charAt(j);
49
50            if (count.get(jc) > 0) {
51                total--;
52            }
53            count.put(jc, count.get(jc) - 1);
54
55            while (total == 0) {
56                minLen = Math.min(minLen, j - i + 1);
57
58                Character ic = str.charAt(i);
59                if (count.get(ic) >= 0) {
60                    total++;
61                }
62                count.put(ic, count.get(ic) + 1);
63
64                i++;
65            }
66            j++;
67        }
68        return minLen;
69    }
70 }
```

## Python算法源码

```
1 # 输入获取
2 s = input()
3
4
5 # 算法入口
6 def getResult(s):
7     # 此时count记录统计W,A,S,D字母的数量
8     count = {
9         "W": 0,
10        "A": 0,
11        "S": 0,
12        "D": 0
13    }
14
15    for c in s:
16        count[c] += 1
17
```

## Python算法源码

```
1 # 输入获取
2 s = input()
3
4
5 # 算法入口
6 def getResult(s):
7     # 此时count记录统计W,A,S,D字母的数量
8     count = {
9         "W": 0,
10        "A": 0,
11        "S": 0,
12        "D": 0
13    }
14
15    for c in s:
16        count[c] += 1
17
18    avg = len(s) / 4 # 平衡状态时, W,A,S,D应该都是avg数量
19    total = 0 # total用于记录多余字母个数
20    flag = True # flag表示当前是否为平衡状态, 默认是
21
22    for c in count.keys():
23        if count[c] > avg:
24            flag = False # 如果有一个字母数量超标, 则平衡打破
25            count[c] -= avg # 此时count记录每个字母超过avg的数量
26            total += count[c]
27        else:
28            count[c] = 0
29
30    if flag:
31        return 0 # 如果平衡, 则输出0
32
33    i = 0
34    j = 0
35    minLen = len(s) - 1
36
37    while j < len(s):
38        jc = s[j]
39
40        if count[jc] > 0:
41            total -= 1
42            count[jc] -= 1
43
44        while total == 0:
45            minLen = min(minLen, j - i + 1)
46
47            ic = s[i]
48            if count[ic] >= 0:
49                total += 1
50                count[ic] += 1
51
52            i += 1
53
54        j += 1
55
56    return minLen
57
58
59 print(getResult(s))
```