

# 【不定滑窗】-寻找符合要求的最长子串

## 题目描述与示例

### 题目描述

给定一个字符串 `s`，找出这样一个子串：

1. 该子串中的任意一个字符最多出现 `2` 次；
2. 该子串不包含指定某个字符；

请你找出满足该条件的最长子串的长度。

### 输入

第一行为要求不包含的指定字符，为单个字符，取值范围 `[0-9a-zA-Z]`

第二行为字符串 `s`，每个字符范围 `[0-9a-zA-Z]`，长度范围 `[1,10000]`

### 输出

一个整数，满足条件的最长子串的长度；

如果不存在满足条件的子串，则返回 `0`

### 示例一

#### 输入

```
1 D
2 ABC123
```

#### 输出

```
1 6
```

## 示例二

### 输入

```
1 D
2 ABACD1231
```

### 输出

```
1 4
```

## 解题思路

子串被两个条件限制：

1. 同一字符数目不能超过 2
2. 不能包含指定字符

如果只考虑第一个限制条件，那么这就是一道典型的滑动窗口题目。该子问题用我们的**滑窗三问三答**来解题。

## 滑窗三问

**Q1：**对于每一个右指针 `right` 所指的元素 `ch`，做什么操作？

**Q2：**什么时候要令左指针 `left` 右移？`left` 对应的元素做什么操作？`while` 中的**循环不变量**是什么？

**Q3：**什么时候进行 `ans` 的更新？

## 滑窗三答

**A1：**将 `ch` 加入哈希表 `hash_table` 中进行频数统计中

**A2:** 右指针 `right` 所指的元素 `ch` 在哈希表中出现的个数超过了 `2`，说明当前滑窗对应的子串不符合题意，需要将 `left` 所指的元素 `s[left]` 在哈希表中的频数 `-1`，同时令 `left` 右移，直到 `ch` 在哈希集合中的频数为 `2`。

**A3:** `left` 右移结束后，更新 `ans`。

对于第二个限制条件，其实也很好处理。有两种处理方式：

1. 在遍历 `s` 的过程中，遇到指定字符 `a` 直接跳过，从 `right+1` 位置开始重新进行滑窗过程，同时哈希表清空
2. 先根据指定字符 `a` 对 `s` 进行切割，得到单词列表 `lst`，对于列表中的每一个单词 `word` 分别进行滑窗过程，对所有滑窗过程得到结果 `temp` 取最大值。

## 代码

### 解法一

(对第二个限制条件的第一种处理方式)

### Python

```
1 # 题目：2024E-寻找符合要求的最长子串
2 # 分值：200
3 # 作者：许老师-闭着眼睛学数理化
4 # 算法：不定滑窗
5 # 代码看不懂的地方，请直接在群上提问
6
7 import collections
8
9 a = input()
10 s = input()
11
12 left, ans = 0, 0
13 hash_table = collections.Counter() # 哈希表记录窗口中各个字符出现频数
14 for right, ch in enumerate(s):
15     if ch != a: # ch不为应该排除的数：进行滑窗
16         hash_table[ch] += 1 # A1
17         while (hash_table[ch] == 3): # A2
18             hash_table[s[left]] -= 1
19             left += 1
20         ans = max(ans, right-left+1) # A3
```

```

21     else:                                     # ch为应该排除的数：滑窗清空，从right+1开始
        下一个滑窗
22         left = right+1
23         hash_table = collections.Counter()
24
25 print(ans)

```

## Java

```

1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         String a = scanner.nextLine();
7         String s = scanner.nextLine();
8
9         int left = 0;
10        int ans = 0;
11        int[] hashTable = new int[128]; // Array to record the frequency of
        characters in the window
12
13        for (int right = 0; right < s.length(); right++) {
14            char ch = s.charAt(right);
15
16            if (ch != a.charAt(0)) { // If ch is not the character to be
        excluded, slide the window
17                hashTable[ch]++; // A1
18                while (hashTable[ch] == 3) { // A2
19                    char leftChar = s.charAt(left);
20                    hashTable[leftChar]--;
21                    left++;
22                }
23                ans = Math.max(ans, right - left + 1); // A3
24            } else { // If ch is the character to be excluded, clear the
        window and start a new one from right+1
25                left = right + 1;
26                hashTable = new int[128];
27            }
28        }
29
30        System.out.println(ans);
31    }
32 }
33

```

## C++

```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4
5 using namespace std;
6
7 int main() {
8     string a;
9     getline(cin, a);
10
11     string s;
12     getline(cin, s);
13
14     int left = 0;
15     int ans = 0;
16     int hashTable[128] = {0}; // Array to record the frequency of characters
    in the window
17
18     for (int right = 0; right < s.length(); right++) {
19         char ch = s[right];
20
21         if (ch != a[0]) { // If ch is not the character to be excluded, slide
    the window
22             hashTable[ch]++; // A1
23             while (hashTable[ch] == 3) { // A2
24                 char leftChar = s[left];
25                 hashTable[leftChar]--;
26                 left++;
27             }
28             ans = max(ans, right - left + 1); // A3
29         } else { // If ch is the character to be excluded, clear the window
    and start a new one from right+1
30             left = right + 1;
31             fill(hashTable, hashTable + 128, 0);
32         }
33     }
34
35     cout << ans << endl;
36
37     return 0;
38 }
39
```

## 解法二

(对第二个限制条件的第二种处理方式)

## Python

```
1 # 题目：2024E-寻找符合要求的最长子串
2 # 分值：200
3 # 作者：许老师-闭着眼睛学数理化
4 # 算法：不定滑窗
5 # 代码看不懂的地方，请直接在群上提问
6
7 import collections
8
9 a = input()
10 s = input()
11
12 ans = 0
13 lst = s.split(a)                # 根据指定字符a对s进行切割，得到单词列表lst
14 for word in lst:                # 对于列表中的每一个单词word分别进行滑窗过程
15     left, temp = 0, 0
16     hash_table = collections.Counter() # 哈希表记录窗口中各个字符出现频数
17     # 对word进行滑窗，word中最大值为temp
18     for right, ch in enumerate(word):
19         hash_table[ch] += 1        # A1
20         while (hash_table[ch] == 3): # A2
21             hash_table[word[left]] -= 1
22             left += 1
23         temp = max(temp, right-left+1) # A3
24     ans = max(ans, temp)          # 每次滑窗过程结束后的结果temp，都要拿来更新
25                                 ans
26 print(ans)
```

## Java

```
1 import java.util.Scanner;
2
3 public class Main {
```

```

4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         String a = scanner.nextLine();
7         String s = scanner.nextLine();
8
9         int ans = 0;
10        String[] words = s.split(a); // Split the string 's' based on 'a'
11
12        for (String word : words) {
13            int left = 0;
14            int temp = 0;
15            int[] hashTable = new int[128]; // An array to record the
frequency of characters in the window
16
17            for (int right = 0; right < word.length(); right++) {
18                char ch = word.charAt(right);
19
20                if (ch != a.charAt(0)) { // If 'ch' is not the character to be
excluded, slide the window
21                    hashTable[ch]++; // A1
22                    while (hashTable[ch] == 3) { // A2
23                        char leftChar = word.charAt(left);
24                        hashTable[leftChar]--;
25                        left++;
26                    }
27                    temp = Math.max(temp, right - left + 1); // A3
28                } else { // If 'ch' is the character to be excluded, clear the
window and start a new one from right+1
29                    left = right + 1;
30                    hashTable = new int[128];
31                }
32            }
33
34            ans = Math.max(ans, temp);
35        }
36
37        System.out.println(ans);
38    }
39 }
40

```

## C++

```

1 #include <iostream>
2 #include <string>

```

```

3 #include <vector>
4 #include <algorithm>
5 #include <unordered_map>
6
7 using namespace std;
8
9 int main() {
10     string a;
11     getline(cin, a);
12
13     string s;
14     getline(cin, s);
15
16     int ans = 0;
17     vector<string> words;
18     size_t pos = 0;
19     while ((pos = s.find(a)) != string::npos) {
20         words.push_back(s.substr(0, pos));
21         s.erase(0, pos + a.length());
22     }
23     words.push_back(s);
24
25     for (const string& word : words) {
26         int left = 0;
27         int temp = 0;
28         unordered_map<char, int> hashTable;
29
30         for (int right = 0; right < word.length(); right++) {
31             char ch = word[right];
32             hashTable[ch]++; // A1
33             while (hashTable[ch] == 3) { // A2
34                 char leftChar = word[left];
35                 hashTable[leftChar]--;
36                 left++;
37             }
38             temp = max(temp, right - left + 1); // A3
39         }
40         ans = max(ans, temp);
41     }
42
43     cout << ans << endl;
44
45     return 0;
46 }
47

```



## 时空复杂度

时间复杂度： $O(N)$ 。仅需一次遍历整个字符串  $s$ 。

空间复杂度： $O(N)$ 。哈希表所占用空间。