

【华为OD机考 统一考试机试C卷】 求满足条件的最长子串的长度（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

给定一个字符串，只包含字母和数字，按要求找出字符串中的最长（连续）子串的长度，字符串本身是其最长的子串，子串要求：

- 1、只包含1个字母(a~z, A~Z)，其余必须是数字；
- 2、字母可以在子串中的任意位置；

如果找不到满足要求的子串，如全是字母或全是数字，则返回-1。

输入描述

字符串(只包含字母和数字)

输出描述

子串的长度

用例

输入	abC124ACb
----	-----------

输出	4
说明	满足条件的最长子串是C124或者124A，长度都是4

输入	a5
输出	2
说明	字符串自身就是满足条件的子串，长度为2

输入	aBB9
输出	2
说明	满足条件的子串为B9，长度为2

输入	abcdef
输出	-1
说明	没有满足要求的子串，返回-1

C++

```
1  #include <iostream>
2  #include <deque>
3  #include <string>
4
5  using namespace std;
6
7  int main() {
8      // 读取输入的字符串
9      string str;
10     cin >> str;
11
12     // 初始化最长子串长度为-1
13     int maxLen = -1;
14     // 初始化一个标志，表示是否找到了包含字母的子串
15
```

```

16 bool hasLetter = false;
17
18 // 初始化双指针l和r，分别表示子串的左右边界
19 int l = 0, r = 0;
20 // 创建一个双端队列用于存储字母的索引
21 deque<int> letterIdx;
22
23 // 遍历字符串
24 while (r < str.length()) {
25     // 获取当前字符
26     char c = str[r];
27
28     // 如果当前字符是字母
29     if (isalpha(c)) {
30         // 设置标志为true，表示找到了包含字母的子串
31         hasLetter = true;
32         // 将字母的索引添加到队列的尾部
33         letterIdx.push_back(r);
34
35         // 如果队列中有多于1个字母的索引
36         if (letterIdx.size() > 1) {
37             // 移除队列头部的字母索引，并将左指针l移动到该索引的下一个位置
38             l = letterIdx.front() + 1;
39             letterIdx.pop_front();
40         }
41
42         // 如果右指针r等于左指针l，跳过当前循环
43         if (r == l) {
44             r++;
45             continue;
46         }
47     }
48
49     // 更新最长子串长度
50     maxLen = max(maxLen, r - l + 1);
51     // 移动右指针
52     r++;
53 }
54
55 // 如果没有找到包含字母的子串，输出-1

```

```

56     if (!hasLetter) {
57         cout << -1 << endl;
58     } else {
59         // 否则输出最长子串长度
60         cout << maxLen << endl;
61     }
62
63     return 0;
64 }

```

java

```

1
2 import java.util.ArrayDeque;
3 import java.util.Deque;
4 import java.util.Scanner;
5
6 public class Main {
7     public static void main(String[] args) {
8         // 创建Scanner对象用于读取输入
9         Scanner sc = new Scanner(System.in);
10        // 读取输入的字符串
11        String str = sc.next();
12        // 初始化最长子串长度为-1
13        int maxLen = -1;
14        // 初始化一个标志, 表示是否找到了包含字母的子串
15        boolean hasLetter = false;
16
17        // 初始化双指针l和r, 分别表示子串的左右边界
18        int l = 0, r = 0;
19        // 创建一个双端队列用于存储字母的索引
20        Deque<Integer> letterIdx = new ArrayDeque<>();
21
22        // 遍历字符串
23        while (r < str.length()) {
24            // 获取当前字符
25            char c = str.charAt(r);
26
27            // 如果当前字符是字母

```

```

48     if (Character.isLetter(c)) {
49         // 设置标志为true, 表示找到了包含字母的子串
50         hasLetter = true;
51         // 将字母的索引添加到队列的尾部
52         letterIdx.addLast(r);
53
54         // 如果队列中有多于1个字母的索引
55         if (letterIdx.size() > 1) {
56             // 移除队列头部的字母索引, 并将左指针l移动到该索引的下一个位置
57             l = letterIdx.removeFirst() + 1;
58         }
59
60         // 如果右指针r等于左指针l, 跳过当前循环
61         if (r == l) {
62             r++;
63             continue;
64         }
65     }
66
67     // 更新最长子串长度
68     maxLen = Math.max(maxLen, r - l + 1);
69     // 移动右指针
70     r++;
71 }
72
73 // 如果没有找到包含字母的子串, 输出-1
74 if (!hasLetter) {
75     System.out.println(-1);
76 } else {
77     // 否则输出最长子串长度
78     System.out.println(maxLen);
79 }
80 }
81 }
82

```

javaScript

```

1  const readline = require('readline').createInterface({
2    input: process.stdin,
3

```

```
3   output: process.stdout
4   });
5
6   // 读取输入的字符串
7   readline.on('line', (str) => {
8       // 初始化最长子串长度为-1
9       let maxLen = -1;
10      // 初始化一个标志, 表示是否找到了包含字母的子串
11      let hasLetter = false;
12
13      // 初始化双指针l和r, 分别表示子串的左右边界
14      let l = 0, r = 0;
15      // 创建一个双端队列用于存储字母的索引
16      let letterIdx = [];
17
18      // 遍历字符串
19      while (r < str.length) {
20          // 获取当前字符
21          let c = str.charAt(r);
22
23          // 如果当前字符是字母
24          if (c.match(/[a-zA-Z]/)) {
25              // 设置标志为true, 表示找到了包含字母的子串
26              hasLetter = true;
27              // 将字母的索引添加到队列的尾部
28              letterIdx.push(r);
29
30              // 如果队列中有多于1个字母的索引
31              if (letterIdx.length > 1) {
32                  // 移除队列头部的字母索引, 并将左指针l移动到该索引的下一个位置
33                  l = letterIdx.shift() + 1;
34              }
35
36              // 如果右指针r等于左指针l, 跳过当前循环
37              if (r === l) {
38                  r++;
39                  continue;
40              }
41          }
42      }
43  }
```

```

44     // 更新最长子串长度
45     maxLen = Math.max(maxLen, r - l + 1);
46     // 移动右指针
47     r++;
48 }
49
50 // 如果没有找到包含字母的子串, 输出-1
51 if (!hasLetter) {
52     console.log(-1);
53 } else {
54     // 否则输出最长子串长度
55     console.log(maxLen);
56 }
57
58 readline.close();
59 });

```

python

```

1  from collections import deque
2
3  # 读取输入的字符串
4  str = input()
5
6  # 初始化最长子串长度为-1
7  maxLen = -1
8  # 初始化一个标志, 表示是否找到了包含字母的子串
9  hasLetter = False
10
11 # 初始化双指针l和r, 分别表示子串的左右边界
12 l, r = 0, 0
13 # 创建一个双端队列用于存储字母的索引
14 letterIdx = deque()
15
16 # 遍历字符串
17 while r < len(str):
18     # 获取当前字符
19     c = str[r]
20
21

```

```

41 # 如果当前字符是字母
42 if c.isalpha():
43     # 设置标志为true, 表示找到了包含字母的子串
44     hasLetter = True
45     # 将字母的索引添加到队列的尾部
46     letterIdx.append(r)
47
48     # 如果队列中有多于1个字母的索引
49     if len(letterIdx) > 1:
50         # 移除队列头部的字母索引, 并将左指针l移动到该索引的下一个位置
51         l = letterIdx.popleft() + 1
52
53     # 如果右指针r等于左指针l, 跳过当前循环
54     if r == l:
55         r += 1
56         continue
57
58     # 更新最长子串长度
59     maxLen = max(maxLen, r - l + 1)
60     # 移动右指针
61     r += 1
62
63 # 如果没有找到包含字母的子串, 输出-1
64 if not hasLetter:
65     print(-1)
66 else:
67     # 否则输出最长子串长度
68     print(maxLen)
69

```

C语言

```

1 #include <stdio.h>
2 #include <ctype.h>
3 #include <string.h>
4 #include <stdbool.h>
5
6 #define MAX_LEN 1000
7
8 // 创建一个双端队列用于存储字母的索引
9

```



```
9  int letterIdx[MAX_LEN];
10 int front = 0, rear = -1;
11
12 // 入队操作
13 void push(int idx) {
14     rear++;
15     letterIdx[rear] = idx;
16 }
17
18 // 出队操作
19 void pop() {
20     front++;
21 }
22
23 // 获取队首元素
24 int getFront() {
25     return letterIdx[front];
26 }
27
28 // 判断队列是否为空
29 bool isEmpty() {
30     return rear < front;
31 }
32
33 int main() {
34     // 读取输入的字符串
35     char str[MAX_LEN];
36     scanf("%s", str);
37
38     // 初始化最长子串长度为-1
39     int maxLen = -1;
40     // 初始化一个标志, 表示是否找到了包含字母的子串
41     bool hasLetter = false;
42
43     // 初始化双指针l和r, 分别表示子串的左右边界
44     int l = 0, r = 0;
45
46     // 遍历字符串
47     while (r < strlen(str)) {
48         // 获取当前字符
49
```

```
50     char c = str[r];
51
52     // 如果当前字符是字母
53     if (isalpha(c)) {
54         // 设置标志为true, 表示找到了包含字母的子串
55         hasLetter = true;
56         // 将字母的索引添加到队列的尾部
57         push(r);
58
59         // 如果队列中有多于1个字母的索引
60         if (rear - front + 1 > 1) {
61             // 移除队列头部的字母索引, 并将左指针l移动到该索引的下一个位置
62             l = getFront() + 1;
63             pop();
64         }
65
66         // 如果右指针r等于左指针l, 跳过当前循环
67         if (r == l) {
68             r++;
69             continue;
70         }
71     }
72
73     // 更新最长子串长度
74     maxLen = r - l + 1 > maxLen ? r - l + 1 : maxLen;
75     // 移动右指针
76     r++;
77 }
78
79 // 如果没有找到包含字母的子串, 输出-1
80 if (!hasLetter) {
81     printf("-1\n");
82 } else {
83     // 否则输出最长子串长度
84     printf("%d\n", maxLen);
85 }
86
87 return 0;
}
```

完整用例

用例1

abC124ACb

用例2

a5

用例3

aBB9

用例4

abcdef

用例5

123456

用例6

aBcD1234567890EFG

用例7

aBcD1234567890EFGHijklmnopqrstuvwxyzZ

用例8

aBcD1234aBcD1234aBcD1234

用例9

aBcD1234aBcD1234aBcD12345

用例10

aBcD1234aBcD1234aBcD1234a

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

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