【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++

```cpp
#include <iostream>
#include <deque>
#include <string>

using namespace std;

int main() {
    // 读取输入的字符串
    string str;
    cin >> str;

    // 初始化最长子串长度为-1
    int maxLen = -1;
    // 初始化一个标志，表示是否找到了包含字母的子串

```

```
    bool hasLetter = false;

    // 初始化双指针l和r，分别表示子串的左右边界
    int l = 0, r = 0;
    // 创建一个双端队列用于存储字母的索引
    deque<int> letterIdx;

    // 遍历字符串
    while (r < str.length()) {
        // 获取当前字符
        char c = str[r];

        // 如果当前字符是字母
        if (isalpha(c)) {
            // 设置标志为true，表示找到了包含字母的子串
            hasLetter = true;
            // 将字母的索引添加到队列的尾部
            letterIdx.push_back(r);

            // 如果队列中有多于1个字母的索引
            if (letterIdx.size() > 1) {
                // 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
                l = letterIdx.front() + 1;
                letterIdx.pop_front();
            }

            // 如果右指针r等于左指针l，跳过当前循环
            if (r == l) {
                r++;
                continue;
            }
        }

        // 更新最长子串长度
        maxLen = max(maxLen, r - l + 1);
        // 移动右指针
        r++;
    }

    // 如果没有找到包含字母的子串，输出-1
```

```
    if (!hasLetter) {
        cout << -1 << endl;
    } else {
        // 否则输出最长子串长度
        cout << maxLen << endl;
    }

    return 0;
}
```

java

```
import java.util.ArrayDeque;
import java.util.Deque;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // 创建Scanner对象用于读取输入
        Scanner sc = new Scanner(System.in);
        // 读取输入的字符串
        String str = sc.next();
        // 初始化最长子串长度为-1
        int maxLen = -1;
        // 初始化一个标志，表示是否找到了包含字母的子串
        boolean hasLetter = false;

        // 初始化双指针l和r，分别表示子串的左右边界
        int l = 0, r = 0;
        // 创建一个双端队列用于存储字母的索引
        Deque<Integer> letterIdx = new ArrayDeque<>();

        // 遍历字符串
        while (r < str.length()) {
            // 获取当前字符
            char c = str.charAt(r);

            // 如果当前字符是字母
```

```
            if (Character.isLetter(c)) {
                // 设置标志为true，表示找到了包含字母的子串
                hasLetter = true;
                // 将字母的索引添加到队列的尾部
                letterIdx.addLast(r);

                // 如果队列中有多于1个字母的索引
                if (letterIdx.size() > 1) {
                    // 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
                    l = letterIdx.removeFirst() + 1;
                }

                // 如果右指针r等于左指针l，跳过当前循环
                if (r == l) {
                    r++;
                    continue;
                }
            }

            // 更新最长子串长度
            maxLen = Math.max(maxLen, r - l + 1);
            // 移动右指针
            r++;
        }

        // 如果没有找到包含字母的子串，输出-1
        if (!hasLetter) {
            System.out.println(-1);
        } else {
            // 否则输出最长子串长度
            System.out.println(maxLen);
        }
    }
}
```

javaScript

```
const readline = require('readline').createInterface({
  input: process.stdin,
```

```
  output: process.stdout
});

// 读取输入的字符串
readline.on('line', (str) => {
  // 初始化最长子串长度为-1
  let maxLen = -1;
  // 初始化一个标志，表示是否找到了包含字母的子串
  let hasLetter = false;

  // 初始化双指针l和r，分别表示子串的左右边界
  let l = 0, r = 0;
  // 创建一个双端队列用于存储字母的索引
  let letterIdx = [];

  // 遍历字符串
  while (r < str.length) {
    // 获取当前字符
    let c = str.charAt(r);

    // 如果当前字符是字母
    if (c.match(/[a-zA-Z]/)) {
      // 设置标志为true，表示找到了包含字母的子串
      hasLetter = true;
      // 将字母的索引添加到队列的尾部
      letterIdx.push(r);

      // 如果队列中有多于1个字母的索引
      if (letterIdx.length > 1) {
        // 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
        l = letterIdx.shift() + 1;
      }

      // 如果右指针r等于左指针l，跳过当前循环
      if (r === l) {
        r++;
        continue;
      }
    }
```

```
    // 更新最长子串长度
    maxLen = Math.max(maxLen, r - l + 1);
    // 移动右指针
    r++;
  }

  // 如果没有找到包含字母的子串，输出-1
  if (!hasLetter) {
    console.log(-1);
  } else {
    // 否则输出最长子串长度
    console.log(maxLen);
  }

  readline.close();
});
```

python

```
from collections import deque

# 读取输入的字符串
str = input()

# 初始化最长子串长度为-1
maxLen = -1
# 初始化一个标志，表示是否找到了包含字母的子串
hasLetter = False

# 初始化双指针l和r，分别表示子串的左右边界
l, r = 0, 0
# 创建一个双端队列用于存储字母的索引
letterIdx = deque()

# 遍历字符串
while r < len(str):
    # 获取当前字符
    c = str[r]

```

```
    # 如果当前字符是字母
    if c.isalpha():
        # 设置标志为true，表示找到了包含字母的子串
        hasLetter = True
        # 将字母的索引添加到队列的尾部
        letterIdx.append(r)

        # 如果队列中有多于1个字母的索引
        if len(letterIdx) > 1:
            # 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
            l = letterIdx.popleft() + 1

        # 如果右指针r等于左指针l，跳过当前循环
        if r == l:
            r += 1
            continue

    # 更新最长子串长度
    maxLen = max(maxLen, r - l + 1)
    # 移动右指针
    r += 1

# 如果没有找到包含字母的子串，输出-1
if not hasLetter:
    print(-1)
else:
    # 否则输出最长子串长度
    print(maxLen)
```

C语言

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

#define MAX_LEN 1000

// 创建一个双端队列用于存储字母的索引
```

```
int letterIdx[MAX_LEN];
int front = 0, rear = -1;

// 入队操作
void push(int idx) {
    rear++;
    letterIdx[rear] = idx;
}

// 出队操作
void pop() {
    front++;
}

// 获取队首元素
int getFront() {
    return letterIdx[front];
}

// 判断队列是否为空
bool isEmpty() {
    return rear < front;
}

int main() {
    // 读取输入的字符串
    char str[MAX_LEN];
    scanf("%s", str);

    // 初始化最长子串长度为-1
    int maxLen = -1;
    // 初始化一个标志，表示是否找到了包含字母的子串
    bool hasLetter = false;

    // 初始化双指针l和r，分别表示子串的左右边界
    int l = 0, r = 0;

    // 遍历字符串
    while (r < strlen(str)) {
        // 获取当前字符

```

```
        char c = str[r];

        // 如果当前字符是字母
        if (isalpha(c)) {
            // 设置标志为true，表示找到了包含字母的子串
            hasLetter = true;
            // 将字母的索引添加到队列的尾部
            push(r);

            // 如果队列中有多于1个字母的索引
            if (rear - front + 1 > 1) {
                // 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
                l = getFront() + 1;
                pop();
            }

            // 如果右指针r等于左指针l，跳过当前循环
            if (r == l) {
                r++;
                continue;
            }
        }

        // 更新最长子串长度
        maxLen = r - l + 1 > maxLen ? r - l + 1 : maxLen;
        // 移动右指针
        r++;
    }

    // 如果没有找到包含字母的子串，输出-1
    if (!hasLetter) {
        printf("-1\n");
    } else {
        // 否则输出最长子串长度
        printf("%d\n", maxLen);
    }

    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

aBcD1234567890EFGHijkImnopqrstuvwxyZ

用例8

aBcD1234aBcD1234aBcD1234

用例9

aBcD1234aBcD1234aBcD12345

用例10

aBcD1234aBcD1234aBcD1234a

文章目录


机考真题
华为OD
No No No
CSDN @算法大师