题目描述

一款大型动画片，故事一个字符串字符串记录地图上的游戏环境的大型叫声，通过此叫声最少由几只大鹰组出。

具体如

1.大鹰发出的叫声为"quack"，因为有多只大鹰同一时间爆发叫声，所以字符串中可能会混合多个"quack"。

2.大鹰会按顺序发出"quack"，即字符串中'q'、'u'、'a'、'c'、'k'这5个字母按顺序完整的出现为一只大鹰，如果不完整或者没有按顺序出现则不计数。

3.如果字符串不是由'q'、'u'、'a'、'c'、'k'字符组合而成，或者没有找到一只大鹰，请返回-1。

输入描述

一个字符串，包含大鹰quack的叫声，1 <= 字符串长度 <= 1000，字符串中的字符只有'q'、'u'、'a'、'c'、'k'。

输出描述

大鹰的数量

用例

输入	quackquack
输出	1
说明	无

输入	qaauccqckk
输出	-1
说明	无

输入	quackquac
输出	1
说明	无

输入	qqquaaaqccauqkcauqqkcauuqkcaaukccakkck
输出	3
说明	无

题目解析

本题虽然看起来长 ，

俗话说，你是来家要大个leetcode公益题，那就是

> 大鹰会依次完整的出"quack"，即字符串中'q'、'u'、'a'、'c'、'k'这5个字母按顺序完整的出现为一只大鹰。如果不完整或者没有按顺序出现不计数。

用leetcode的原题

> 如果字符串 croak 不是由若干有效的 "croak" 字符混合而成，请返回 -1 。

leetcode公益题，如果存在不完整或者不按顺序的叫声，则直接返回-1，结束程序。

反本题，统计不完整或者不按顺序的叫声，只要不予计数，在最后都被剔除出。

比如 quauack

如果按照leetcode的原题来做的话，结果应该返回-1。

而本题要编写需要返回1。

leetcode提供2的原因是，下面要考虑部除不完整或者不按顺序的叫声。

而本题毫无疑问是，需要考虑除除不完整或者不按顺序的叫声。

我的解题思路是，依出每个叫声quack的区间范围，用一个叫声的：jq索引位置，k索引位置。

实现点：从左到右遍历叫声字符串，遇到q，就向其索引位置加入栈中。

遇到u，到叫声u出现的字数值自减过了u出现的次数，若是就忽略本次u，否则u次数++

遇到a，到叫声a出现的字数值自减过了a出现的次数，若是就忽略本次a，否则a次数++

遇到c，到叫声c出现的字数值自减过了a出现的次数，若是就忽略本次c，否则c次数++

遇到k，到叫声k的数值自减过了k出现的次数，若是就忽略本次k，否则k次数++

遇到q，到叫声q的数值自减过了q出现的次数，若是就忽略本次q，否则q次数++

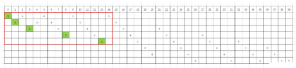
现将上图遍历，用红对以图例顺序说属区间不完整的叫声，并且把所有每个会出叫声的区间范围。

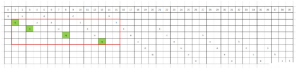
接下来就是把每一个区间，和后面区间比较，看是否有交集，若有，则已承交集数。

最终最大的交集的数是最多大鹰数。

下图是例例5：qqquaaaqccauqkcauqqkcauuqkcaaukccakkck

包含由叫声区间的交集示意图





JavaScript算法源码

```
// 引入 readline 模块
const readline = require("readline");

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

rl.on("line", (line) => {
  console.log(getResult(line));
});

function getResult(quacks) {
  let q, u, a, c, k;
  q = [];
  u = [];
  a = [];
  c = [];

  const range = [];

  for (let i = 0; i < quacks.length; i++) {
    switch (quacks[i]) {
      case "q":
        q.push(i);
        break;
      case "u":
        if (u + 1 >= q.length) u++;
      break;
      case "a":
        if (a + 1 >= u) a++;
        break;
      case "c":
        if (c + 1 >= a) c++;
      break;
      case "k":
        if (k >= c) {
          range.push([q.shift(), i]);
          u--;
          a--;
          c--;
        }
        break;
      default:
        return -1;
    }
  }

  if (! range.length) return -1;

  let max = 1;
  for (let i = 0; i < range.length; i++) {
    let count = 1;
    for (let j = i + 1; j < range.length; j++) {
      if (range[i][1] >= range[j][0]) count++;
    }
    max = Math.max(max, count);
  }

  return max;
}
```

Java算法源码

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String quack = sc.next();
    System.out.print(getResult(quack));
  }

  public static int getResult(String quack) {
    LinkedList<Integer> q = new LinkedList<>();

    int u = 0, a = 0, c = 0;

    ArrayList<Integer[]> ranges = new ArrayList<>();

    for (int i = 0; i < quack.length(); i++) {
      switch (quack.charAt(i)) {
        case 'q':
          q.add(i);
          break;
        case 'u':
          if (u + 1 >= q.size()) u++;
          break;
        case 'a':
          if (a + 1 >= u) a++;
          break;
        case 'c':
          if (c >= a) {
            ranges.add(new Integer[] {q.removeFirst(), i});
            u--;
            a--;
            c--;
          }
          break;
        default:
          return -1;
      }
    }

    if (ranges.size() == 0) return -1;

    int ans = 1;
    for (int i = 0; i < ranges.size(); i++) {
      int count = 1;
      for (int j = i + 1; j < ranges.size(); j++) {
        if (ranges.get(i)[1] >= ranges.get(j)[0]) {
          count++;
        }
      }
      ans = Math.max(ans, count);
    }

    return ans;
  }
}
```

Python算法源码

```
# 输入获取
quacks = input()

# 算法实现
def getResult():
    q = []
    u, a, c = 0, 0, 0

    ranges = []

    for i in range(len(quacks)):
        char = quacks[i]

        if char == 'q':
            q.append(i)
        elif char == 'u':
            if u + 1 >= len(q):
                u += 1
        elif char == 'a':
            if a + 1 >= u:
                a += 1
        elif char == 'c':
            if c + 1 >= a:
                c += 1
        elif char == 'k':
            if c >= 1:
                ranges.append([q.pop(0), i])
                u -= 1
                a -= 1
                c -= 1
        else:
            return -1

    if len(ranges) == 0:
        return -1

    ans = 1
    for i in range(len(ranges)):
        count = 1
        for j in range(i + 1, len(ranges)):
            if ranges[i][1] >= ranges[j][0]:
                count += 1

        ans = max(ans, count)

    return ans

# 输出结果
print(getResult())
```