# **E卷-数大雁[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**

题目描述

群大雁往南飞，给定一个字符串记录地面上的游客听到的大雁叫声，请给出叫声最少由几只大雁发出。具体的:.大雁发出的完整叫声为"quack"，因为有多只大雁同一时间嘎嘎作响，所以字符串中可能会混合多个"quack"。2.大雁会依次完整发出"quack"，即字符串中'g,u,a,“”℃,K'这5个字母按顺序完整存在才能计数为一只大雁，如果不完整或者没有按顺序则不予计数。

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

输入描述

个字符串，包含大雁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

大雁的数量

示例1

输入 quackquack

输出 1

示例2

输入 qaauucqckk

输出 -1



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.nextLine(); // 读取输入的字符串

Map<Character, Integer> mp = new HashMap<>();

// 建立字符到索引的映射

mp.put('q', 0);

mp.put('u', 1);

mp.put('a', 2);

mp.put('c', 3);

mp.put('k', 4);

int f = 0; // 标志位，用于判断是否有非法字符或不完整的叫声

int[] cnt = new int[5]; // 记录每个字符出现的次数

int need = 0; // 记录需要的最少大雁数量

for (int i = 0; i < s.length(); i++) {

char now = s.charAt(i); // 当前字符

int index = mp.getOrDefault(now, -1); // 获取当前字符的索引

if (index == -1) { // 如果字符不在"quack"中

f = 1;

break;

}

if (index == 0) { // 如果是'q'字符

if (cnt[4] > 0) cnt[4]--; // 如果之前有完整的"quack"，则减少一个

else need++; // 否则需要增加一个新的大雁

cnt[0]++; // 增加'q'的计数

} else {

cnt[index - 1]--; // 减少前一个字符的计数

cnt[index]++; // 增加当前字符的计数

}

for (int j = 0; j < 4;

j++) if (cnt[j] < 0) f = 1; // 检查是否有非法的字符顺序

}

for (int i = 0; i < 4;

i++) if (cnt[i] > 0) f = 1; // 检查是否有不完整的"quack"

if (f == 1) System.out.println(

-1); // 如果有非法字符或不完整的叫声，输出-1

else System.out.println(need); // 否则输出需要的最少大雁数量

}

}



def main():

import sys

input = sys.stdin.read().strip()

s = input # 获取输入的字符串

mp = {"q": 0, "u": 1, "a": 2, "c": 3, "k": 4} # 建立字符到索引的映射

cnt = [0] \* 5 # 记录每个字符出现的次数

f = 0 # 标志位，用于判断是否有非法字符或不完整的叫声

need = 0 # 记录需要的最少大雁数量

for now in s:

index = mp.get(now, -1) # 获取当前字符的索引

if index == -1: # 如果字符不在"quack"中

f = 1

break

if index == 0: # 如果是'q'字符

if cnt[4] > 0:

cnt[4] -= 1 # 如果之前有完整的"quack"，则减少一个

else:

need += 1 # 否则需要增加一个新的大雁

cnt[0] += 1 # 增加'q'的计数

else:

cnt[index - 1] -= 1 # 减少前一个字符的计数

cnt[index] += 1 # 增加当前字符的计数

if any(x < 0 for x in cnt[:-1]): # 检查是否有非法的字符顺序

f = 1

if any(cnt[i] > 0 for i in range(4)): # 检查是否有不完整的"quack"

f = 1

if f == 1:

print(-1) # 输出-1

else:

print(need) # 输出需要的最少大雁数量

if \_\_name\_\_ == "\_\_main\_\_":

main()



#include <iostream>

#include <unordered\_map>

#include <vector>

using namespace std;

int main() {

string s;

cin >> s; // 输入字符串

unordered\_map<char, int> mp = {{'q', 0}, {'u', 1}, {'a', 2}, {'c', 3}, {'k', 4}}; // 建立字符到索引的映射

vector<int> cnt(5, 0); // 记录每个字符出现的次数

int f = 0; // 标志位，用于判断是否有非法字符或不完整的叫声

int need = 0; // 记录需要的最少大雁数量

for (char now : s) {

int index = mp.count(now) ? mp[now] : -1; // 获取当前字符的索引

if (index == -1) { // 如果字符不在"quack"中

f = 1;

break;

}

if (index == 0) { // 如果是'q'字符

if (cnt[4] > 0)

cnt[4]--; // 如果之前有完整的"quack"，则减少一个

else

need++; // 否则需要增加一个新的大雁

cnt[0]++; // 增加'q'的计数

} else {

cnt[index - 1]--; // 减少前一个字符的计数

cnt[index]++; // 增加当前字符的计数

}

for (int j = 0; j < 4;

j++) if (cnt[j] < 0) f = 1; // 检查是否有非法的字符顺序

}

for (int i = 0; i < 4;

i++) if (cnt[i] > 0) f = 1; // 检查是否有不完整的"quack"

if (f == 1)

cout << -1 << endl; // 输出-1

else

cout << need << endl; // 输出需要的最少大雁数量

return 0;

}



#include <stdio.h>

#include <string.h>

int main() {

char s[1000];

scanf("%s", s); // 输入字符串

int mp[256]; // ASCII字符到索引的映射

memset(mp, -1, sizeof(mp));

mp['q'] = 0;

mp['u'] = 1;

mp['a'] = 2;

mp['c'] = 3;

mp['k'] = 4;

int cnt[5] = {0}; // 记录每个字符出现的次数

int f = 0; // 标志位，用于判断是否有非法字符或不完整的叫声

int need = 0; // 记录需要的最少大雁数量

for (int i = 0; i < strlen(s); i++) {

char now = s[i];

int index = mp[now];

if (index == -1) { // 如果字符不在"quack"中

f = 1;

break;

}

if (index == 0) { // 如果是'q'字符

if (cnt[4] > 0)

cnt[4]--; // 如果之前有完整的"quack"，则减少一个

else

need++; // 否则需要增加一个新的大雁

cnt[0]++; // 增加'q'的计数

} else {

cnt[index - 1]--; // 减少前一个字符的计数

cnt[index]++; // 增加当前字符的计数

}

for (int j = 0; j < 4;

j++) if (cnt[j] < 0) f = 1; // 检查是否有非法的字符顺序

}

for (int i = 0; i < 4;

i++) if (cnt[i] > 0) f = 1; // 检查是否有不完整的"quack"

if (f == 1)

printf("-1\n"); // 输出-1

else

printf("%d\n", need); // 输出需要的最少大雁数量

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.on("line", (input) => {

const s = input.trim(); // 输入字符串

const mp = { q: 0, u: 1, a: 2, c: 3, k: 4 }; // 建立字符到索引的映射

const cnt = new Array(5).fill(0); // 记录每个字符出现的次数

let f = 0; // 标志位，用于判断是否有非法字符或不完整的叫声

let need = 0; // 记录需要的最少大雁数量

for (let i = 0; i < s.length; i++) {

const now = s[i];

const index = mp[now] !== undefined ? mp[now] : -1; // 获取当前字符的索引

if (index === -1) {

// 如果字符不在"quack"中

f = 1;

break;

}

if (index === 0) {

// 如果是'q'字符

if (cnt[4] > 0) cnt[4]--; // 如果之前有完整的"quack"，则减少一个

else need++; // 否则需要增加一个新的大雁

cnt[0]++; // 增加'q'的计数

} else {

cnt[index - 1]--; // 减少前一个字符的计数

cnt[index]++; // 增加当前字符的计数

}

for (let j = 0; j < 4; j++) if (cnt[j] < 0) f = 1; // 检查是否有非法的字符顺序

}

for (let i = 0; i < 4; i++) if (cnt[i] > 0) f = 1; // 检查是否有不完整的"quack"

if (f === 1) console.log(-1); // 输出-1

else console.log(need); // 输出需要的最少大雁数量

});



package main

import (

"fmt"

)

func main() {

var s string

fmt.Scan(&s) // 输入字符串

mp := map[byte]int{'q': 0, 'u': 1, 'a': 2, 'c': 3, 'k': 4} // 建立字符到索引的映射

cnt := [5]int{0, 0, 0, 0, 0} // 记录每个字符出现的次数

f := 0 // 标志位，用于判断是否有非法字符或不完整的叫声

need := 0 // 记录需要的最少大雁数量

for i := 0; i < len(s); i++ {

now := s[i]

index, exists := mp[now] // 获取当前字符的索引

if !exists { // 如果字符不在"quack"中

f = 1

break

}

if index == 0 { // 如果是'q'字符

if cnt[4] > 0 {

cnt[4]-- // 如果之前有完整的"quack"，则减少一个

} else {

need++ // 否则需要增加一个新的大雁

}

cnt[0]++ // 增加'q'的计数

} else {

cnt[index-1]-- // 减少前一个字符的计数

cnt[index]++ // 增加当前字符的计数

}

for j := 0; j < 4; j++ {

if cnt[j] < 0 {

f = 1 // 检查是否有非法的字符顺序

break

}

}

}

for i := 0; i < 4; i++ {

if cnt[i] > 0 {

f = 1 // 检查是否有不完整的"quack"

break

}

}

if f == 1 {

fmt.Println(-1) // 输出-1

} else {

fmt.Println(need) // 输出需要的最少大雁数量

}

}