2025.03.17 虾皮 1.5h【a2】

编程

Lc16

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

String s = in.nextLine();

int res = 1;

int n = s.length();

if (n <= 1) {

System.out.println(n);

return;

}

int left = 0, right = 0;

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

while (right < n) {

char rc = s.charAt(right);

int idx = mp.getOrDefault(rc, 0);

left = Math.max(left, idx);

res = Math.max(res, right-left+1);

mp.put(rc, right+1);

right++;

}

System.out.println(res);

return;

}

}

Lc21

import java.util.Scanner;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

String ss = in.nextLine();import java.util.\*;

import java.util.Scanner;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

import java.util.stream.Collectors;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

String ss = in.nextLine();

// System.out.println(ss);

Pattern pattern = Pattern.compile("L1 = \\[(.\*?)\\], L2 = \\[(.\*?)\\]");

Matcher matcher = pattern.matcher(ss);

List<Integer> L1 = new LinkedList<>();

List<Integer> L2 = new LinkedList<>();

if (matcher.find()) {

L1 = Arrays.stream(matcher.group(1).split(","))

.map(Integer::parseInt)

.collect(Collectors.toCollection(LinkedList::new));

L2 = Arrays.stream(matcher.group(2).split(","))

.map(Integer::parseInt)

.collect(Collectors.toCollection(LinkedList::new));

// System.out.println(L1);

// System.out.println(L2);

}

List<Integer> merged = new LinkedList<>();

Iterator<Integer> it1 = L1.iterator();

Iterator<Integer> it2 = L2.iterator();

Integer val1 = it1.hasNext() ? it1.next() : null;

Integer val2 = it2.hasNext() ? it2.next() : null;

while (val1!=null || val2!=null) {

if (val1 == null) {

merged.add(val2);

val2 = it2.hasNext() ? it2.next() : null;

} else if (val2 == null) {

merged.add(val1);

val1 = it1.hasNext() ? it1.next() : null;

} else if (val1 <= val2) {

merged.add(val1);

val1 = it1.hasNext() ? it1.next() : null;

} else {

merged.add(val2);

val2 = it2.hasNext() ? it2.next() : null;

}

}

StringBuilder sb = new StringBuilder();

sb.append("[");

for (Integer num : merged) {

sb.append(num).append(",");

}

sb.deleteCharAt(sb.length()-1);

sb.append("]");

System.out.println(sb.toString());

return;

}

}

// System.out.println(ss);

Pattern pattern = Pattern.compile("s = \"(.\*?)\", k = (\\d+)");

Matcher matcher = pattern.matcher(ss);

StringBuilder sb = new StringBuilder();

int k = 0;

if (matcher.find()) {

sb.append(matcher.group(1));

k = Integer.parseInt(matcher.group(2));

// System.out.println(sb);

// System.out.println(k);

}

char[] ca = sb.toString().toCharArray();

int res = 0;

int vow = 0;

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

if (ca[i] == 'a' || ca[i] == 'e' || ca[i] == 'i' || ca[i] == 'o' || ca[i] == 'u') {

vow++;

}

if (i < k-1) {

continue;

}

res = Math.max(res, vow);

char c = ca[i-k+1];

if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {

vow--;

}

}

System.out.println(res);

return;

}

}

Lc1456

import java.util.Scanner;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

String ss = in.nextLine();

// System.out.println(ss);

Pattern pattern = Pattern.compile("s = \"(.\*?)\", k = (\\d+)");

Matcher matcher = pattern.matcher(ss);

StringBuilder sb = new StringBuilder();

int k = 0;

if (matcher.find()) {

sb.append(matcher.group(1));

k = Integer.parseInt(matcher.group(2));

// System.out.println(sb);

// System.out.println(k);

}

char[] ca = sb.toString().toCharArray();

int res = 0;

int vow = 0;

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

if (ca[i] == 'a' || ca[i] == 'e' || ca[i] == 'i' || ca[i] == 'o' || ca[i] == 'u') {

vow++;

}

if (i < k-1) {

continue;

}

res = Math.max(res, vow);

char c = ca[i-k+1];

if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {

vow--;

}

}

System.out.println(res);

return;

}

}

2025.03.15 美团 1.5h【a1.5】

选择

模板模式、组合模式、装饰器模式、原型模式哪个是行为型

哪种垃圾回收方式按照时间分割任务，垃圾回收和增变者交错

49 38 65 97 76 13 27 49，前三轮排序结果都是38.49.65.97.76.13.27.49，可能是什么排序算法

6个数据链路层帧，物理信道失败率10%，无连接服务，则发送网络层分组成功率

XYZN四个字符任意方式进栈，任意方式出栈得到字符串，由多少种入栈排列得到XYZN（卡塔兰数）

已经知道段号、段首地址、段长度和逻辑地址1262，怎么计算对应物理地址

在create table时候，unique（a,b）语句在a，b字段语句之前之后之中，有什么影响

按照时间分割工作任务是什么回收方式

部分回收、世代垃圾回收、增量回收

编程

1、签到题，字符串处理，if

2、i范围l1r1 j范围l2r2，排列组合，如果i是j的倍数，res++（二维前缀和）

3、最小生成树

2025.03.15 OPPO 2h【a3】

选择

try后面必须跟catch快，finally可以没有，对吗

53842，用冒泡排序，两次完整遍历后是什么样

数据表1有abbc，数据表2有aaabbcd，那么select\*from1 2 where1.x==2。x返回结果是什么

HTTP缓存作用是减少流量还是增加带宽

深度为3的完全二叉树节点最少多少

linux通配符解释? \* [] {}

列出将-r--r--r--权限改为631的所有方法

15，9，7，23，45，8，67，93，3快速排序，以第一个记录为基准，从小到大第一次划分结果.3.9.7.8.15.23.45.93.67还是3.9.7.8.15.45.67.93.23？

范式级别越高，数据库性能越好

53842冒泡

编程

1、签到题

2、MEX

3、有一个数字，位数很多，里面有一些数字被污染遮盖（可以是0-9），如果整个数是3的倍数，有多少种排列组合数？注意（1）避免前导0（2）结果记得取余（3）用dp思维来做

2025.03.19 美的 1.5h【a3】

单选

操作系统动态分区算法回收内存，如果回收区仅仅与空闲区链插入点前一个分区相邻接，那么需要？

java8引用特性，names.stream().map(string.touppercase.foreachsystem.,out.println),为我修改使其语法正确

java定义类可以不写构造函数吗

类里面的private的x属性可以被实例obj.x读取到吗

被中断的进程可以立即获得CPU控制权恢复执行，对还是错

被中断的进程将进入就绪队列等待执行，对还是错

为我介绍ICMP

书写方法头，使用类名作为前缀就可以调用A.function，方法头应该写成什么样

linux哪个指令为脚本取消可执行权限

多选

mysql的四种常见日志哪些需要解析后才能查看

SSL协议客户端和服务器握手阶段交互的信息有？使用的对称算法、不重数、MAC算法、使用的公钥算法，哪些是? SSL握手阶段交互MAC算法了吗？

XSS攻击可分为存储型和反射型？对吗？XSS攻击主要方法是什么？

对特殊字符转义处理可以预防SQL注入吗？

为我枚举java实现多线程的方法？自定义实现runnable缺点是没有返回值吗？

需要构造一个thread实例去启动？

按需分配可以破坏死锁的什么条件？银行家算法是死锁解除算法，正确吗？资源动态分配过程防止系统安全状态，可以避免发生死锁，对吗？在资源动态分配过程中防止系统进入安全状态可以避免死锁吗？

为我判断sql是否正确 update set id = 10 from a where id = 1？为我判断sql是否正确select \* from a limit(7,10)？

编程

最长递增子序列

给你一个nums整数数组，求最长递增子序列长度

循环字符串

一个循环字符串的子字符串s，可以多次连接到自身以获得此循环字符串的最小可能字符串长度是多少？

输入“cabca“

输出3

因为cab长度为3

public class MinRepeatedSubstring {

public static int minRepeatedSubstringLength(String s) {

int n = s.length();

// 尝试每个可能的子串长度 i

for (int i = 1; i <= n; i++) {

// 如果字符串长度可以被 i 整除

if (n % i == 0) {

String sub = s.substring(i, j);

// 构造一个由 sub 重复拼接的字符串

StringBuilder sb = new StringBuilder();

for (int j = 0; j < n / i; j++) {

sb.append(sub);

}

// 如果拼接得到的字符串 sb 包含原字符串 s，返回子串的长度 i

if (sb.toString().contains(s)) {

return i;

}

}

}

return n; // 如果找不到，返回整个字符串长度

}

public static void main(String[] args) {

String s = "cabca";

System.out.println(minRepeatedSubstringLength(s)); // 输出 3

}

}

孤岛

4 5

00000

00\*00

0\*0\*0

00\*00

Bfs