

华为OD机试 - 字符串摘要 (Java & JS & Python)

原创

伏城之外

已于 2023-06-03 16:43:11 修改

1247

收藏 6

版权

分类专栏：

华为OD机试AB (Java & JS & Python)

文章标签：

算法

华为机试

Java

JavaScript

Python

OD

华为OD机试AB (Ja... 同时被 2 个专栏收录

该专栏为热销专栏榜 第2名

¥59.90

¥99.00

3382 订阅

371 篇文章

已订阅

题目描述

给定一个字符串的摘要算法，请输出给定字符串的摘要。

- 去除字符串中非字母的符号。
- 如果出现连续字符(不区分大小写)，则输出：该字符 (小写) + 连续出现的次数。
- 如果是非连续的字符(不区分大小写)，则输出：该字符(小写) + 该字母之后字符串中出现的该字符的次数
- 对按照以上方式表示后的字符串进行排序：字母和紧随的数字作为一组进行排序，数字大的在前，数字相同的，则按字母进行排序，字母小的在前。

输入描述

一行字符串，长度为[1,200]

输出描述

摘要字符串

用例

输入	aabbcc
输出	a2b2c2
说明	无

输入	bAaAcBb
输出	a3b2b2c0
说明	bAaAcBb: 第一个b非连续字母，该字母之后字符串中还出现了2次(最后的两个Bb)，所以输出b2 a连续出现3次，输出a3， c非连续，该字母之后字符串再没有出现过c，输出c0 Bb连续2次，输出b2 对b2a3c0b2进行排序，最终输出a3b2b2c0

题目解析

本题主要考察逻辑分析。

本题主要难点在于：

如果是非连续的字符(不区分大小写)，则输出：该字符(小写) + 该字母之后字符串中出现的该字符的次数

如果当前位置的字母是一个非连续字符，那么我们需要统计当前位置之后的该字母出现次数。

为了避免重复的扫描统计，我们可以一开始就统计好所有字母的出现次数到count中，每扫描一个位置，则对于位置的字母数量count[letter]--，表示该字母在后面还剩多少个。

这样的话，碰到非连续字母letter时，我们只需要获取count[letter]即可知道其后续还有多少个letter字母。

```
1  import java.util.ArrayList;
2  import java.util.Scanner;
3
4  public class Main {
5      // 字母数字类
6      static class Letter {
7          char letter;
8          int num;
9
10         public Letter(char letter, int num) {
11             this.letter = letter;
12             this.num = num;
13         }
14
15         @Override
16         public String toString() {
17             return this.letter + "" + this.num;
18         }
19     }
20
21     public static void main(String[] args) {
22         Scanner sc = new Scanner(System.in);
23         System.out.println(getResult(sc.nextLine()));
24     }
25
26     public static String getResult(String s) {
27         // 不区分大小写
28         s = s.toLowerCase();
```

```
29
30 // 统计每个字母出现的次数
31 int[] count = new int[128];
32
33 // 去除字符串中的非字母
34 StringBuilder sb = new StringBuilder();
35
36 for (int i = 0; i < s.length(); i++) {
37     char c = s.charAt(i);
38     if (c >= 'a' && c <= 'z') {
39         count[c]++;
40         sb.append(c);
41     }
42 }
43
44 // 加空格是为了避免后续的收尾操作，如果有疑问可以移除下面加空格操作
45 s = sb + " ";
46
47 // 记录连续字母和非连续字母
48 ArrayList<Letter> ans = new ArrayList<>();
49
50 // 上一个位置的字母
51 char pre = s.charAt(0);
52 // 该字母连续次数记为1
53 int repeat = 1;
54 // 后续该字母还有count[pre]-=1个
55 count[pre]--;
56
57 for (int i = 1; i < s.length(); i++) {
58     // 当前位置的字母
59     char cur = s.charAt(i);
60     // 后续该字母还有count[cur]-=1个
61     count[cur]--;
62
63     if (cur == pre) {
64         // 如果当前位置和上一个位置的字母相同，则产生连续
65         // 连续次数+1
66         repeat++;
67     } else {
68         // 如果当前位置和上一个位置的字母不同，则连续打断
```

```

69         // 如果pre字母连续次数>1, 则是真连续, 那么就是pre+repeat, 否则就是假连续, 是
pre+count[pre]
70         ans.add(new Letter(pre, repeat > 1 ? repeat : count[pre]));
71         // 更新pre为cur
72         pre = cur;
73         // 更新pre连续次数为1
74         repeat = 1;
75     }
76 }
77
78 // 字母和紧随的数字作为一组进行排序, 数字大的在前, 数字相同的, 则按字母进行排序, 字母小的在
前
79 ans.sort((a, b) -> a.num != b.num ? b.num - a.num : a.letter - b.letter);
80
81 StringBuilder res = new StringBuilder();
82 for (Letter an : ans) {
83     res.append(an.toString());
84 }
85 return res.toString();
86 }
87 }

```