【华为OD机考 统一考试机试C卷】字符串摘要（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

给定一个字符串的摘要算法，请输出给定字符串的摘要值

1. 去除字符串中非字母的符号。
2. 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。
3. 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
4. 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1

输入

```
aabbcc
```

输出

```
a2b2c2
```

用例2

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

C++

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main() {
    string input;
    getline(cin, input);
    transform(input.begin(), input.end(), input.begin(), ::tolower); // 将输入的字符串转换为小写

```

```cpp

    // 统计每个字符出现的次数
    int charCount[128] = {0}; // ASCII 码表中共有 128 个字符
    string sb; // 用于存储去除非字母的符号后的字符串
    for (char c : input) { // 遍历字符串的每个字符
        if (c >= 'a' && c <= 'z') { // 如果该字符是字母
            charCount[c]++; // 该字符出现次数加 1
            sb.push_back(c); // 将该字符添加到 sb 中
        }
    }

    // 在每个字符后面加上其出现的次数或者连续出现的次数
    sb.push_back(' '); // 在字符串末尾加上一个空格，方便统计最后一个字符的出现次数
    vector<string> ans; // 用于存储每个字符的摘要值
    char pre = sb[0]; // pre 表示当前正在处理的字符的前一个字符
    int repeat = 1; // repeat 表示当前正在处理的字符的连续出现次数
    charCount[pre]--; // 将 pre 出现的次数减 1
    for (int i = 1; i < sb.length(); i++) { // 遍历字符串的每个字符
        char cur = sb[i]; // cur 表示当前正在处理的字符
        charCount[cur]--; // 将 cur 出现的次数减 1
        if (cur == pre) { // 如果 cur 和 pre 相等，表示出现了连续字符
            repeat++; // 连续出现次数加 1
        } else { // 如果 cur 和 pre 不相等，表示出现了非连续字符
            ans.push_back(string(1, pre) + (repeat > 1 ? to_string(repeat) : to_string(charCount[pre]))); // 将 pre 的摘要值添加到 ans 中
            pre = cur; // 更新 pre 为当前字符
            repeat = 1; // 重置连续出现次数为 1
        }
    }

    // 排序
    sort(ans.begin(), ans.end(), [](string a, string b) { // 根据题目要求对 ans 进行排序
        if (a.back() != b.back()) { // 如果最后一个字符不相等，按照数字大小排序
            return b.back() < a.back();
        } else { // 如果最后一个字符相等，按照字母顺序排序
            return a.front() < b.front();
        }
    });

    // 输出结果
    string res; // 用于存储最终的摘要值

```

```
    for (string an : ans) { // 遍历 ans
        res += an; // 将每个元素添加到 res 中
    }
    cout << res << endl; // 输出最终的摘要值

    return 0;
}
```

Java

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

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine();
        input = input.toLowerCase(); // 将输入的字符串转换为小写

        // 统计每个字符出现的次数
        int[] charCount = new int[128]; // ASCII 码表中共有 128 个字符
        StringBuilder sb = new StringBuilder(); // 用于存储去除非字母的符号后的字符串
        for (char c : input.toCharArray()) { // 遍历字符串的每个字符
            if (c >= 'a' && c <= 'z') { // 如果该字符是字母
                charCount[c]++; // 该字符出现次数加 1
                sb.append(c); // 将该字符添加到 sb 中
            }
        }

        // 在每个字符后面加上其出现的次数或者连续出现的次数
        input = sb + " "; // 在字符串末尾加上一个空格，方便统计最后一个字符的出现次数
        ArrayList<String> ans = new ArrayList<>(); // 用于存储每个字符的摘要值
        char pre = input.charAt(0); // pre 表示当前正在处理的字符的前一个字符
        int repeat = 1; // repeat 表示当前正在处理的字符的连续出现次数
        charCount[pre]--; // 将 pre 出现的次数减 1
        for (int i = 1; i < input.length(); i++) { // 遍历字符串的每个字符
            char cur = input.charAt(i); // cur 表示当前正在处理的字符
            charCount[cur]--; // 将 cur 出现的次数减 1
```

```
            if (cur == pre) { // 如果 cur 和 pre 相等，表示出现了连续字符
                repeat++; // 连续出现次数加 1
            } else { // 如果 cur 和 pre 不相等，表示出现了非连续字符
                ans.add(pre + "" + (repeat > 1 ? repeat : charCount[pre])); // 将 pre 的摘要值添加到 ans 中
                pre = cur; // 更新 pre 为当前字符
                repeat = 1; // 重置连续出现次数为 1
            }
        }

        // 排序
        Object[] ansArray = ans.toArray(); // 将 ans 转换为数组
        Arrays.sort(ansArray, (a, b) -> { // 根据题目要求对 ansArray 进行排序
            String s1 = (String) a;
            String s2 = (String) b;
            if (s1.charAt(s1.length() - 1) != s2.charAt(s2.length() - 1)) { // 如果最后一个字符不相等，按照数字大小排序
                return s2.charAt(s2.length() - 1) - s1.charAt(s1.length() - 1);
            } else { // 如果最后一个字符相等，按照字母顺序排序
                return s1.charAt(0) - s2.charAt(0);
            }
        });

        // 输出结果
        StringBuilder res = new StringBuilder(); // 用于存储最终的摘要值
        for (Object an : ansArray) { // 遍历 ansArray
            res.append(an); // 将每个元素添加到 res 中
        }
        System.out.println(res.toString()); // 输出最终的摘要值
    }
}
```

JavaScript

```
const readline = require('readline');

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

rl.on('line', (input) => {
```

```
input = input.toLowerCase(); // 将输入的字符串转换为小写

// 统计每个字符出现的次数
const charCount = new Array(128).fill(0); // ASCII 码表中共有 128 个字符
let sb = ''; // 用于存储去除非字母的符号后的字符串
for (let i = 0; i < input.length; i++) { // 遍历字符串的每个字符
  const c = input[i];
  if (c >= 'a' && c <= 'z') { // 如果该字符是字母
    charCount[c.charCodeAt()]++; // 该字符出现次数加 1
    sb += c; // 将该字符添加到 sb 中
  }
}

// 在每个字符后面加上其出现的次数或者连续出现的次数
input = sb + ' '; // 在字符串末尾加上一个空格，方便统计最后一个字符的出现次数
const ans = []; // 用于存储每个字符的摘要值
let pre = input.charAt(0); // pre 表示当前正在处理的字符的前一个字符
let repeat = 1; // repeat 表示当前正在处理的字符的连续出现次数
charCount[pre.charCodeAt()]--; // 将 pre 出现的次数减 1
for (let i = 1; i < input.length; i++) { // 遍历字符串的每个字符
  const cur = input.charAt(i); // cur 表示当前正在处理的字符
  charCount[cur.charCodeAt()]--; // 将 cur 出现的次数减 1
  if (cur === pre) { // 如果 cur 和 pre 相等，表示出现了连续字符
    repeat++; // 连续出现次数加 1
  } else { // 如果 cur 和 pre 不相等，表示出现了非连续字符
    ans.push(pre + (repeat > 1 ? repeat : charCount[pre.charCodeAt()])); // 将 pre 的摘要值添加到 ans 中
    pre = cur; // 更新 pre 为当前字符
    repeat = 1; // 重置连续出现次数为 1
  }
}

// 排序
ans.sort((a, b) => { // 根据题目要求对 ans 进行排序
  if (a.charAt(a.length - 1) !== b.charAt(b.length - 1)) { // 如果最后一个字符不相等，按照数字大小排序
    return b.charAt(b.length - 1) - a.charAt(a.length - 1);
  } else { // 如果最后一个字符相等，按照字母顺序排序
    return a.charAt(0) - b.charAt(0);
  }
});
```

```
  // 输出结果
  let res = ''; // 用于存储最终的摘要值
  for (const an of ans) { // 遍历 ans
    res += an; // 将每个元素添加到 res 中
  }
  console.log(res); // 输出最终的摘要值
});
```

Python

```python
import sys

# 统计每个字符出现的次数
charCount = [0] * 128  # ASCII 码表中共有 128 个字符
sb = []  # 用于存储去除非字母的符号后的字符串
input = sys.stdin.readline().strip().lower()  # 将输入的字符串转换为小写
for c in input:
    if 'a' <= c <= 'z':  # 如果该字符是字母
        charCount[ord(c)] += 1  # 该字符出现次数加 1
        sb.append(c)  # 将该字符添加到 sb 中

# 在每个字符后面加上其出现的次数或者连续出现的次数
input = ''.join(sb) + ' '  # 在字符串末尾加上一个空格，方便统计最后一个字符的出现次数
ans = []  # 用于存储每个字符的摘要值
pre = input[0]  # pre 表示当前正在处理的字符的前一个字符
repeat = 1  # repeat 表示当前正在处理的字符的连续出现次数
charCount[ord(pre)] -= 1  # 将 pre 出现的次数减 1
for i in range(1, len(input)):  # 遍历字符串的每个字符
    cur = input[i]  # cur 表示当前正在处理的字符
    charCount[ord(cur)] -= 1  # 将 cur 出现的次数减 1
    if cur == pre:  # 如果 cur 和 pre 相等，表示出现了连续字符
        repeat += 1  # 连续出现次数加 1
    else:  # 如果 cur 和 pre 不相等，表示出现了非连续字符
        ans.append(pre + str(repeat) if repeat > 1 else pre + str(charCount[ord(pre)]))  # 将 pre 的摘要值添加到 ans 中
        pre = cur  # 更新 pre 为当前字符
        repeat = 1  # 重置连续出现次数为 1

# 排序
ans.sort(key=lambda x: (-int(x[-1]), x[0]))
```

```
30
31
# 输出结果
32
res = ''.join(ans)  # 用于存储最终的摘要值
33
print(res)  # 输出最终的摘要值
```

C语言

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX_LEN 200

// 定义一个结构体，用于存储字符和对应的次数
typedef struct {
    char ch;
    int count;
} CharCount;

// 定义一个比较函数，用于 qsort 函数
int cmp(const void *a, const void *b) {
    CharCount *cc1 = (CharCount *)a;
    CharCount *cc2 = (CharCount *)b;
    if (cc1->count != cc2->count)
        return cc2->count - cc1->count;
    return cc1->ch - cc2->ch;
}

int main() {
    // 输入的字符串
    char input[MAX_LEN + 1];
    fgets(input, MAX_LEN + 1, stdin);
    input[strcspn(input, "\n")] = '\0';  // 去掉末尾的换行符

    // 统计每个字符出现的次数
    int charCount[128] = {0}; // ASCII 码表中共有 128 个字符
    char sb[MAX_LEN + 1] = {0}; // 用于存储去除非字母的符号后的字符串
    int sb_len = 0;
    for (int i = 0; input[i]; ++i) { // 遍历字符串的每个字符
```

```
        char c = tolower(input[i]); // 将字符转换为小写
        if (c >= 'a' && c <= 'z') { // 如果该字符是字母
            charCount[c]++; // 该字符出现次数加 1
            sb[sb_len++] = c; // 将该字符添加到 sb 中
        }
    }

    // 在每个字符后面加上其出现的次数或者连续出现的次数
    CharCount ans[MAX_LEN] = {0}; // 用于存储每个字符的摘要值
    int ans_len = 0;
    char pre = sb[0]; // pre 表示当前正在处理的字符的前一个字符
    int repeat = 1; // repeat 表示当前正在处理的字符的连续出现次数
    charCount[pre]--; // 将 pre 出现的次数减 1
    for (int i = 1; i < sb_len; ++i) { // 遍历字符串的每个字符
        char cur = sb[i]; // cur 表示当前正在处理的字符
        charCount[cur]--; // 将 cur 出现的次数减 1
        if (cur == pre) { // 如果 cur 和 pre 相等，表示出现了连续字符
            repeat++; // 连续出现次数加 1
        } else { // 如果 cur 和 pre 不相等，表示出现了非连续字符
            ans[ans_len].ch = pre;
            ans[ans_len].count = repeat > 1 ? repeat : charCount[pre];
            ans_len++;
            pre = cur; // 更新 pre 为当前字符
            repeat = 1; // 重置连续出现次数为 1
        }
    }
    ans[ans_len].ch = pre;
    ans[ans_len].count = repeat > 1 ? repeat : charCount[pre];
    ans_len++;

    // 排序
    qsort(ans, ans_len, sizeof(CharCount), cmp);

    // 输出结果
    for (int i = 0; i < ans_len; ++i) { // 遍历 ans
        printf("%c%d", ans[i].ch, ans[i].count); // 输出每个元素
    }
    printf("\n"); // 输出换行符
```

```
            return 0;
        }
```

完整用例

用例1

aabbcc

用例2

abcde

用例3

AABBCC

用例4

bAaAcBb

用例5

abcdabcd

用例6

aaaAAA

用例7

abcABC

用例8

abcdABCD

用例9

aAaAaA

用例10

aabbccddeeffgghhiiijjj

文章目录


机考真题
华为OD
No No No
CSDN @算法大师