题目描述

给定一个字符串s，s包括以空格分隔的若干个单词，请对s进行如下处理后输出：
1、单词内部调整：对每个单词字母重新按字典序排序
2、单词间顺序调整：
1）统计每个单词出现的次数，并按次数降序排列
2）次数相同，按单词长度升序排列
3）次数和单词长度均相同，按字典升序排列

请输出处理后的字符串，每个单词以一个空格分隔。

输入描述

一行字符串，每个字符取值范围：[a-zA-z0-9]以及空格，字符串长度范围：[1，1000]

输出描述

输出处理后的字符串，每个单词以一个空格分隔。

用例

输入	This is an apple
输出	an is This aelpp
说明	无

输入	My sister is in the house not in the yard
输出	in in eht eht My is not adry ehosu eirsst
说明	无

题目解析

本题需要注意的是，先进行单词内部调整，然后再进行单词间顺序。

考察的是排序

JavaScript算法源码

复制

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

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

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

function getResult(arr) {
  arr = arr.map((str) => [...str].sort().join(""));

  const count = arr.reduce((p, c) => {
    p[c] ? p[c]++ : (p[c] = 1);
    return p;
  }, {});

  arr.sort((a, b) =>
    count[a] !== count[b]
      ? count[b] - count[a]
      : a.length !== b.length
      ? a.length - b.length
      : a > b
      ? 1
      : -1
  );

  return arr.join(" ");
```

题目解析

本题需要注意的是，先进行单词内部调整，然后再进行单词间顺序。

考察的是排序

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

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

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

function getResult(arr) {
  arr = arr.map((str) => [...str].sort().join(""));

  const count = arr.reduce((p, c) => {
    p[c] ? p[c]++ : (p[c] = 1);
    return p;
  }, {});

  arr.sort((a, b) =>
    count[a] !== count[b]
      ? count[b] - count[a]
      : a.length !== b.length
      ? a.length - b.length
      : a > b
      ? 1
      : -1
  );

  return arr.join(" ");
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] arr = sc.nextLine().split(" ");
    System.out.println(getResult(arr));
  }

  public static String getResult(String[] arr) {
    arr =
        Arrays.stream(arr)
            .map(
                str -> {
                  char[] cArr = str.toCharArray();
                  Arrays.sort(cArr);
                  return new String(cArr);
                })
            .toArray(String[]::new);

    HashMap<String, Integer> count = new HashMap<>();
    for (String s : arr) {
      count.put(s, count.getOrDefault(s, 0) + 1);
    }

    Arrays.sort(
        arr,
        (a, b) ->
            !count.get(a).equals(count.get(b))
                ? count.get(b) - count.get(a)
                : a.length() != b.length() ? a.length() - b.length() : a.compareTo(b));

    StringJoiner sj = new StringJoiner(" ", "", "");
    for (String s : arr) {
      sj.add(s);
    }
    return sj.toString();
  }
}
```

Python算法源码

```
24
25      HashMap<String, Integer> count = new HashMap<>();
26      for (String s : arr) {
27        count.put(s, count.getOrDefault(s, 0) + 1);
28      }
29
30      Arrays.sort(
31          arr,
32          (a, b) ->
33              !count.get(a).equals(count.get(b))
34                  ? count.get(b) - count.get(a)
35                  : a.length() != b.length() ? a.length() - b.length() : a.compareTo(b));
36
37      StringJoiner sj = new StringJoiner(" ", "", "");
38      for (String s : arr) {
39        sj.add(s);
40      }
41      return sj.toString();
42    }
43  }
```

Python算法源码

```
1  # 输入获取
2  arr = input().split()
3
4
5  # 算法入口
6  def getResult(arr):
7      for i in range(len(arr)):
8          arr[i] = "".join(sorted(arr[i]))
9
10     count = {}
11     for c in arr:
12         if count.get(c) is None:
13             count[c] = 0
14         count[c] += 1
15
16     arr.sort(key=lambda x: (-count[x], len(x), [ord(char) for char in x]))
17
18     return " ".join(map(str, arr))
19
20
21 # 算法调用
22 print(getResult(arr))
```