【模拟】-数大雁

视频直播讲解：2024/08/31 真题讲解（2024E卷）

题目描述与示例

题目描述

一群大雁往南飞，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：

1. 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 `"q" ,"u", "a", "c", "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。
3. 如果字符串不是由 `"q" ,"u", "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

一个字符串，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

大雁的数量

示例一

输入

```
1 quackquack
```

输出

```
1 1
```

示例二

输入

```
1 qaauucqcaa
```

输出

```
1 -1
```

示例三

输入

```
1 quacqkuackquack
```

输出

```
1 2
```

说明

用不同的颜色表示同一只大雁， `quacqkuackquack` ，最少需要 `2` 只大雁。

以下情况都是 `2` 只大雁。

`quacqkuackquack`

`quacqkuacqkuack`

`quacqkuaqckuack`

`quacqkuqackuack`

`quacqkquackuack`

以下情况需要 `3` 只大雁。

`quacqqkuackuack`

`quaqcqkuackuack`

示例四

输入

```
1 quacqkuquacqkacuqkackuack
```

输出

```
1 3
```

说明

用不同的颜色表示同一只大雁， `quacqkuquacqkacuqkackuack` ，最少需要 `3` 只大雁。

解题思路

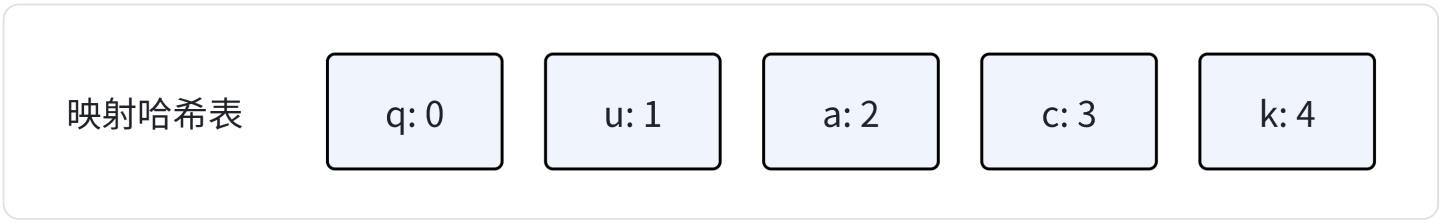
> 注意，本题和LC1419. 数青蛙完全一致。

说实话这题乍一看很让人一头雾水。但其实就是根据题意直接模拟即可。

我们考虑用一个长度为 `5` 的 `cnt` 数组，来记录 `"quack"` 中每个字符被叫了几次，即每个字符的出现个数。

> 要注意， `cnt` 中的计数，并不代表大雁的总个数。 `cnt` 的更新直接在循环中进行即可。这个后面会讲。

这里可以使用一个哈希表 `word` 用于取 `"quack"` 的各个字符分别对应的 `01234` 在 `cnt` 中的索引 `idx` ，即存在 `word = {ch: i for i, ch in enumerate("quack")}` ， `idx` 在**计数过程**和**判断无法完成雁叫**两个地方都会用到。



通过 `idx = word[ch]` ，就可以得到某个字符 `ch` 对应的索引 `idx` 了。

考虑返回 `-1` ，即无法完成雁叫的情况：

- 原字符串 `s` 中，存在其中无关字符，这个可以在遍历过程中进行字符 `ch` 的判断

> 譬如例子： `s = quacky`

- 原字符串 `s` 中，所有字符出现的数目不一致，这个可以在最后做特殊情况的判断。

> 譬如例子： `s = quac` 或 `s = qquack`

- 遍历到某个字符 `ch` 时，比如 `'c'` ，发现其前一个字符即 `'a'` 出现的次数少于当前字符，那么无法实现一个完整的雁叫。

> 譬如例子： `s = qucak`

主要本题的难点，在于大雁**能否可以复用**这个问题比较麻烦。

考虑大雁是否可以复用的问题。假设遇到一个新的 `'q'` ，表示出现了一声新的 `"quack"` ，如果此时：

- `'k'` 的计数大于 `0` ，表示之前有某只大雁叫完了，那么这只大雁可以立马抓来复用，我们对 `cnt` 整体减 `1` 。
- `'k'` 的计数等于 `0` ，表示之前没有任何大雁完成了雁叫，或是先前完成了雁叫的大雁被抓去复用了（无需分辨这两种情况的区别），因此再无需进行任何操作。

思路基本就结束了。梳理一下整个遍历过程的代码框架。

首先我们需要遍历原字符串 `s` 中的字符 `ch` ，在确定其是 `"quack"` 中的字符的前提下，获取其对应在 `cnt` 中的 `idx` ，同时把 `ch` 对应在 `cnt` 中的计数 `+1` 。

```
for ch in s:
    if ch not in word:
        isError = True
        break
    idx = word[ch]
    cnt[idx] += 1

    pass
```

接着我们要判断 `ch` 是否为 `"q"` 。

如果 `ch` 不是 `"q"` ，说明 `ch` 是 `"auck"` 中的其中一个字符。

`ch` 能否作为雁叫中的一个字符，取决于其**前一个字符的个数是否不小于这个字符的个数**。

如果前一个字符的个数 `cnt[idx-1]` 小于当前字符个数 `cnt[idx]` ，说明前一个字符的个数太少，当前的字符 `ch` 不足以构成一声完整的雁叫。

```
for ch in s:
    if ch not in word:
        isError = True
        break
    idx = word[ch]
    cnt[idx] += 1

    if ch != "q" and cnt[idx] > cnt[idx-1]:
        isError = True
        break

    pass
```

如果 `ch` 是 `"q"` ，说明此时产生了一声新的雁叫。

我们必须考虑，之前出现的大雁是否能够复用的问题。

如果之前存在 `"k"` 出现过，即 `cnt[4] >= 1` ，说明已经存在一只完整的大雁。这只大雁可以复用， `cnt` 整体 `-1` ，表示整体可以少算一只大雁。

```python
for ch in s:
    if ch not in word:
        isError = True
        break
    idx = word[ch]
    cnt[idx] += 1

    if ch != "q" and cnt[idx] > cnt[idx-1]:
        isError = True
        break

    if ch == "q" and cnt[4] >= 1:
        for j in range(5):
            cnt[j] -= 1
```

这就完成了整个遍历过程的代码。

代码

Python

```python
# 题目：2024E-数大雁
# 分值：100
# 作者：许老师-闭着眼睛学数理化
# 算法：模拟
# 代码看不懂的地方，请直接在群上提问


s = input()

# 构建一个标记，表示是否出现了异常，初始化为False表示没有异常
isError = False

# 构建哈希表，"quack"分别对应01234一共五个索引
word = {ch: i for i, ch in enumerate("quack")}
# cnt列表：记录"quack"中每个字符被叫了几次，即每个字符的出现个数。
# 要注意，cnt中的计数，并不代表大雁的总个数。
# 另外，cnt[0]也可以用来表示，当前已经使用了几只大雁。
cnt = [0, 0, 0, 0, 0]
for ch in s:
    # 如果字符ch并不是"quack"中的任意一个字符
```

```
    # 说明s中出现了其他无关字符，直接退出循环
    if ch not in word:
        isError = True
        break
    # 获得字符ch在列表cnt中的索引idx，即表示"quack"对应的01234一共五个索引
    idx = word[ch]
    # ch对应的计数+1
    cnt[idx] += 1
    # ch不是"q"，且前一个字符数目少于当前字符数目
    # 说明前一个字符的个数，不足构成以一声新的"quack"
    # 出现错误
    if ch != "q" and cnt[idx] > cnt[idx-1]:
        # 标记isError改为True，同时退出循环
        isError = True
        break
    # 遇到"q"，表示出现新的雁叫，可能可以复用之前的大雁，如果：
    # 1. 之前有某大雁叫过"k"，那么这只大雁可以复用，cnt整体-1，表示可以少算一只大雁
    # 2. 之前没有大雁叫过"k"，那么无法进行大雁的复用，无需做任何操作
    # cnt[4]表示之前叫过"k"的大雁的个数，
    # cnt[4] >= 1即表示，存在某大雁叫过"k"，这只大雁可以拿来复用
    if ch == "q" and cnt[4] >= 1:
        for j in range(5):
            cnt[j] -= 1


# 排除特殊情况，最终计算结束时，cnt中的元素应该值相等
# 如果存在不相等的元素，则说明各个字符的总数不一致，出现异常
if len(set(cnt)) != 1:
    isError = True


# 如果isError为True，说明出现了异常，输出-1
# 否则最终cnt中所有计数一致，这个数即为所需要的大雁的个数，输出之
print(-1) if isError else print(cnt[0])
```

Java

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
```

```java
        Scanner scanner = new Scanner(System.in);
        String s = scanner.nextLine();
        scanner.close();

        // 构建一个标记，表示是否出现了异常，初始化为false表示没有异常
        boolean isError = false;

        // 构建哈希表，"quack"分别对应01234一共五个索引
        Map<Character, Integer> word = new HashMap<>();
        word.put('q', 0);
        word.put('u', 1);
        word.put('a', 2);
        word.put('c', 3);
        word.put('k', 4);

        // cnt数组：记录"quack"中每个字符被叫了几次，即每个字符的出现个数。
        // 要注意，cnt中的计数，并不代表大雁的总个数。
        int[] cnt = new int[5];

        for (char ch : s.toCharArray()) {
            // 获得字符ch在数组cnt中的索引idx，即表示"quack"对应的01234一共五个索引
            int idx = word.get(ch);
            // ch对应的计数+1
            cnt[idx]++;

            // ch不是"q"，且前一个字符数目少于当前字符数目，出现错误
            if (ch != 'q' && cnt[idx] > cnt[idx - 1]) {
                // 标记isError改为true，同时退出循环
                isError = true;
                break;
            }

            // 遇到"q"，表示出现新的雁叫，可能可以复用之前的大雁，如果：
            // 1. 之前有某大雁叫过"k"，那么这只大雁可以复用，cnt整体-1，表示可以少算一只大雁
            // 2. 之前没有大雁叫过"k"，那么无法进行大雁的复用，无需做任何操作
            // cnt[4]表示之前叫过"k"的大雁的个数，
            // cnt[4] >= 1即表示，存在某大雁叫过"k"，这只大雁可以拿来复用
            if (ch == 'q' && cnt[4] >= 1) {
                for (int i = 0; i < 5; i++) {
                    cnt[i]--;
                }
            }
        }

        // 排除特殊情况，最终计算结束时，cnt中的元素应该相等
        // 如果存在不相等的元素，则说明各个字符的总数不一致，出现异常
```

```
        int firstCount = cnt[0];
        for (int i = 1; i < cnt.length; i++) {
            if (cnt[i] != firstCount) {
                isError = true;
                break;
            }
        }

        // 如果isError为true，说明出现了异常，输出-1
        // 否则最终cnt中所有计数一致，这个数即为所需要的大雁的个数，输出之
        if (isError) {
            System.out.println(-1);
        } else {
            System.out.println(cnt[0]);
        }
    }
}

```

C++

```
#include <iostream>
#include <unordered_map>
#include <string>

using namespace std;

int main() {
    string s;
    cin >> s;

    // 构建一个标记，表示是否出现了异常，初始化为false表示没有异常
    bool isError = false;

    // 构建哈希表，"quack"分别对应01234一共五个索引
    unordered_map<char, int> word;
    word['q'] = 0;
    word['u'] = 1;
    word['a'] = 2;
    word['c'] = 3;
    word['k'] = 4;

    // cnt数组：记录"quack"中每个字符被叫了几次，即每个字符的出现个数。
    // 要注意，cnt中的计数，并不代表大雁的总个数。
    int cnt[5] = {0};
```

```
25
26      for (char ch : s) {
27          // 获得字符ch在数组cnt中的索引idx，即表示"quack"对应的01234一共五个索引
28          int idx = word[ch];
29          // ch对应的计数+1
30          cnt[idx]++;
31
32          // ch不是"q"，且前一个字符数目少于当前字符数目，出现错误
33          if (ch != 'q' && cnt[idx] > cnt[idx - 1]) {
34              // 标记isError改为true，同时退出循环
35              isError = true;
36              break;
37          }
38
39          // 遇到"q"，表示出现新的雁叫，可能可以复用之前的大雁，如果：
40          // 1. 之前有某大雁叫过"k"，那么这只大雁可以复用，cnt整体-1，表示可以少算一只大雁
41          // 2. 之前没有大雁叫过"k"，那么无法进行大雁的复用，无需做任何操作
42          // cnt[4]表示之前叫过"k"的大雁的个数，
43          // cnt[4] >= 1即表示，存在某大雁叫过"k"，这只大雁可以拿来复用
44          if (ch == 'q' && cnt[4] >= 1) {
45              for (int i = 0; i < 5; i++) {
46                  cnt[i]--;
47              }
48          }
49      }
50
51      // 排除特殊情况，最终计算结束时，cnt中的元素应该相等
52      // 如果存在不相等的元素，则说明各个字符的总数不一致，出现异常
53      int firstCount = cnt[0];
54      for (int i = 1; i < 5; i++) {
55          if (cnt[i] != firstCount) {
56              isError = true;
57              break;
58          }
59      }
60
61      // 如果isError为true，说明出现了异常，输出-1
62      // 否则最终cnt中所有计数一致，这个数即为所需要的大雁的个数，输出之
63      if (isError) {
64          cout << -1 << endl;
65      } else {
66          cout << cnt[0] << endl;
67      }
68
69      return 0;
70  }
71
```

时空复杂度

时间复杂度：`O(N)`。仅需一次遍历数组。

空间复杂度：`O(1)`。仅需若干常数变量。