题目描述

小强在参加《密室逃生》游戏,当前关卡要求找到符合给定密码K(升序的不重复小写字母组成)的箱子,并给

出箱子编号,箱子编号为 1~N。 每个箱子中都有一个字符串s,字符串由大写字母、小写字母、数字、标点符号、空格组成,需要在这些字符串 中找到所有的字母,忽略大小写后排列出对应的密码串,并返回匹配密码的箱子序号。

提示:满足条件的箱子不超过1个。

输入描述

第一行为 key 的字符串,

第二行为箱子 boxes, 为数组样式, 以逗号分隔

- 箱子N数量満足1≤N≤10000,
- s 长度满足 0 ≤ s.length ≤ 50,
- 密码为仅包含小写字母的升序字符串, 且不存在重复字母,
- 密码 K 长度1≤K.length≤26

输出描述

返回对应箱子编号

如不存在符合要求的密码箱,则返回-1。

用例

输入	abc s,sdf134 A2c4b
输出	2
说明	第 2 个箱子中的 Abc,符合密码 abc。

题目解析

题目不难, 但是感觉本题的输入样例是有问题的, 因为题目中说:

每个箱子中都有一个字符串s,字符串由大写字母、小写字母、数字、标点符号、空格组成

也就是说输入第二行的字符串含有标点符号,那么标点符号就有可能含有逗号","

这就和第二行输入用于隔开每个字符串的逗号","冲突了。

输入描述中说

第二行为箱子 boxes, 为数组样式, 以逗号分隔

第二行输入为数组样式,因此我理解应该第二行输入应该张这样:

["s", "sdf134 A2c4b"]

这样的话, 才不会产生冲突。

因此,这里我将用例1的输入改成

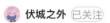
abc

["s", "sdf134 A2c4b"]

另外,本题最简单的解题思路就是,双重循环,外层循环每一个box,内层循环box的每一个字符,然后统计字符^Q 数量,在和key的字符数量对比,如果可以覆盖,即大于等于,则说明当前box就是可以匹配密码的箱子,返回当 前箱子的序号。如果一直找不到,则返回-1。

但是这个复杂度是O(n*slen),即大概50万次循环,可能有超时风险。

因此, 我们可以优化一下, 措施如下:



题目解析

题目不难, 但是感觉本题的输入样例是有问题的, 因为题目中说:

每个箱子中都有一个字符串s,字符串由大写字母、小写字母、数字、标点符号、空格组成

也就是说输入第二行的字符串含有标点符号,那么标点符号就有可能含有逗号","

这就和第二行输入用于隔开每个字符串的逗号","冲突了。

输入描述中说

```
第二行为箱子 boxes, 为数组样式, 以逗号分隔
```

第二行输入为数组样式,因此我理解应该第二行输入应该张这样:

```
["s", "sdf134 A2c4b"]
```

这样的话, 才不会产生冲突。

因此,这里我将用例1的输入改成

```
abc
["s", "sdf134 A2c4b"]
```

另外,本题最简单的解题思路就是,双重循环,外层循环每一个box,内层循环box的每一个字符,然后<mark>统计字符^Q数</mark>量,在和key的字符数量对比,如果可以覆盖,即大于等于,则说明当前box就是可以匹配密码的箱子,返回当前箱子的序号。如果一直找不到,则返回-1。

但是这个复杂度是O(n*slen),即大概50万次循环,可能有超时风险。

因此, 我们可以优化一下, 措施如下:

先将key转成数组,进行字典序排序^Q

外层循环依旧遍历每个box,

如果box.length < key.length,则肯定不行,直接下次循环。

如果box.length >= key.length,则先将box转为小写模式,然后将box转为数组字典序排序,接下来利用两个指针 k,j,分别从key的0索引位置,和box的0索引位置开始扫描,如果扫描到的key[k] == box[j],则k++,j++,否则只有j++。

当k===key.length时,则说明当前box中就是匹配密码的箱子。

这种算法,最坏的情况是O(n*slen),但是其他情况都有优化可能。

JavaScript算法源码

Java算法源码

```
| import java.util.Arrays; import java.util.Scanner; | import java.util.Scanner; | import java.util.Scanner; | import java.util.Scanner; | import java.util.Scanner sc = new Scanner(System.in); | Scanner sc = new Scanner(System.in); | Scanner sc = new Scanner(System.in); | String key = sc.nextLine(); | import java.util.Scanner; | import
```

Python算法源码