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 数量满足 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.l e n g t h ≤ 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，然后统计字符数量，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：

先将key转成数组，进行字典序排序

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

```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,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const key = lines[0];
    const boxes = JSON.parse(lines[1]);
    console.log(getResult(key, boxes));

    lines.length = 0;
  }
});

function getResult(key, boxes) {
  key = [...key].sort();

  for (let i = 0; i < boxes.length; i++) {
    let box = boxes[i];
    if (box.length < key.length) continue;
    box = [...String(box).toLowerCase()].sort();

    let j = 0;
    let k = 0;
    while (k < key.length && j < box.length) {
      if (box[j] == key[k]) k++;
      j++;
    }

    if (k === key.length) {
      return i + 1;
    }
  }
```

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,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const key = lines[0];
    const boxes = JSON.parse(lines[1]);
    console.log(getResult(key, boxes));

    lines.length = 0;
  }
});

function getResult(key, boxes) {
  key = [...key].sort();

  for (let i = 0; i < boxes.length; i++) {
    let box = boxes[i];
    if (box.length < key.length) continue;
    box = [...String(box).toLowerCase()].sort();

    let j = 0;
    let k = 0;
    while (k < key.length && j < box.length) {
      if (box[j] == key[k]) k++;
      j++;
    }

    if (k === key.length) {
      return i + 1;
    }
  }

  return -1;
}
```

Java算法源码

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

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

    String key = sc.nextLine();

    // 注意：用例的第二行输入有问题，这里假设第二行输入为：["s", "sdf134 A2c4b"]
    // 需要注意的是，sc.nextLine()会对输入中的空格做处理，导致我们用正则\\s无法匹配到空格，因此需要给sc.nextLine() + ""
    String str = sc.nextLine() + "";
    String[] boxes = str.substring(2, str.length() - 2).split("\",\\s?\"");

    System.out.println(getResult(key, boxes));
  }

  public static int getResult(String key, String[] boxes) {
    key = strSort(key);

    for (int i = 0; i < boxes.length; i++) {
      String box = boxes[i];
      if (box.length() < key.length()) continue;
      box = strSort(box.toLowerCase());

      int j = 0;
      int k = 0;
      while (k < key.length() && j < box.length()) {
        if (box.charAt(j) == key.charAt(k)) k++;
        j++;
      }

      if (k == key.length()) {
        return i + 1;
      }
    }
    return -1;
  }

  public static String strSort(String str) {
    char[] cArr = str.toCharArray();
    Arrays.sort(cArr);
    return new String(cArr);
  }
}
```

Java算法源码

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

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

    String key = sc.nextLine();

    // 注意：用例的第二行输入有问题，这里假设第二行输入为：["s", "sdf134 A2c4b"]
    // 需要注意的是，sc.nextLine()会对输入中的空格做处理，导致我们用正则\\s无法匹配到空格，因此需要给sc.nextLine() + ""
    String str = sc.nextLine() + "";
    String[] boxes = str.substring(2, str.length() - 2).split("\",\\s?\"");

    System.out.println(getResult(key, boxes));
  }

  public static int getResult(String key, String[] boxes) {
    key = strSort(key);

    for (int i = 0; i < boxes.length; i++) {
      String box = boxes[i];
      if (box.length() < key.length()) continue;
      box = strSort(box.toLowerCase());

      int j = 0;
      int k = 0;
      while (k < key.length() && j < box.length()) {
        if (box.charAt(j) == key.charAt(k)) k++;
        j++;
      }

      if (k == key.length()) {
        return i + 1;
      }
    }
    return -1;
  }

  public static String strSort(String str) {
    char[] cArr = str.toCharArray();
    Arrays.sort(cArr);
    return new String(cArr);
  }
}
```

Python算法源码

```python
# 输入获取
key = input()
# 注意：用例的第二行输入有问题，这里假设第二行输入为：["s", "sdf134 A2c4b"]
boxes = eval(input())


# 算法入口
def getResult(key, boxes):
    sorted(key)

    for i in range(len(boxes)):
        box = boxes[i]
        if len(box) < len(key):
            continue

        box = sorted(box.lower())

        j = 0
        k = 0
        while k < len(key) and j < len(box):
            if box[j] == key[k]:
                k += 1
            j += 1

        if k == len(key):
            return i + 1

    return -1


# 算法调用
print(getResult(key, boxes))
```