题目描述

给定a-z，26个英文字母小写字符串组成的字符串A和B，其中A可能存在重复字母，B不会存在重复字母，现从字符串A中按规则挑选一些字母可以组成字符串B。

挑选规则如下：

- 同一个位置的字母只能挑选一次，
- 被挑选字母的相对先后顺序不能被改变，
- 求最多可以同时从A中挑选多少组能组成B的字符串。

输入描述

输入为2行，第一行输入字符串a,第二行输入字符串b，行首行尾**没有多余空格**

输出描述

输出一行，包含一个数字，表示最多可以同时从a中挑选多少组能组成b的字符串，行末没有多余空格

用例

输入	badc bac
输出	1
说明	无

题目解析

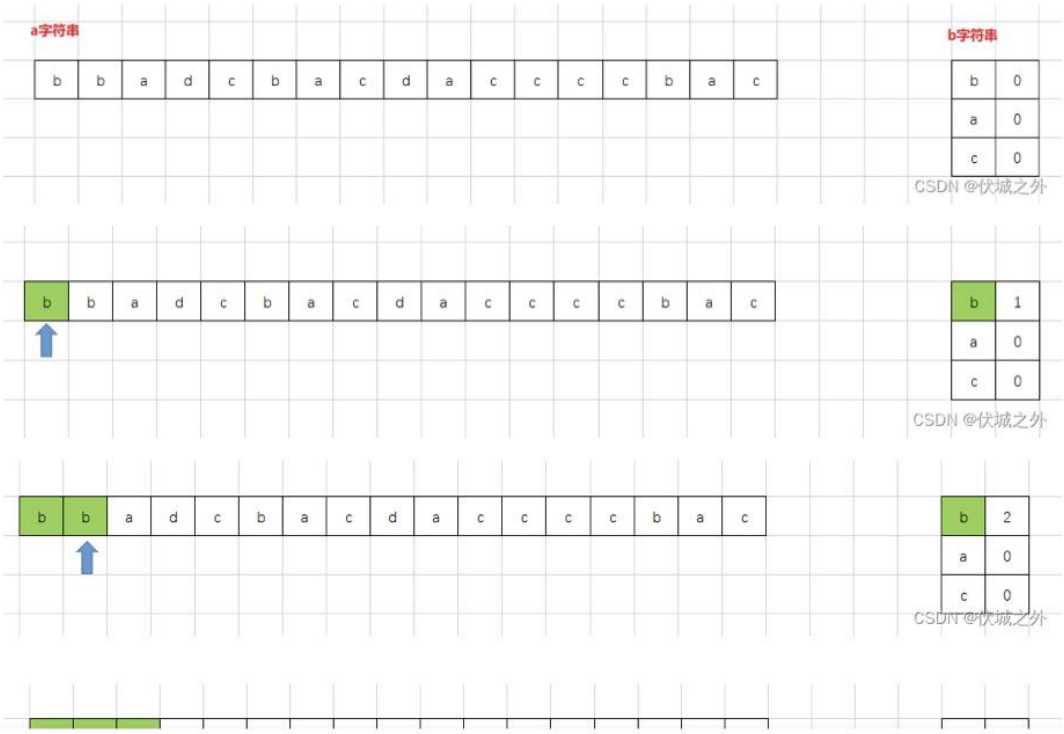
本题求解可以参考

LeetCode - 1419 数青蛙_伏城之外的博客-CSDN博客

华为机试 - 数大雁_伏城之外的博客-CSDN博客

题目的用例不能说明问题，我们可以通过下面用例

```
bbadcbacdaccccbac
bac
```



题目解析

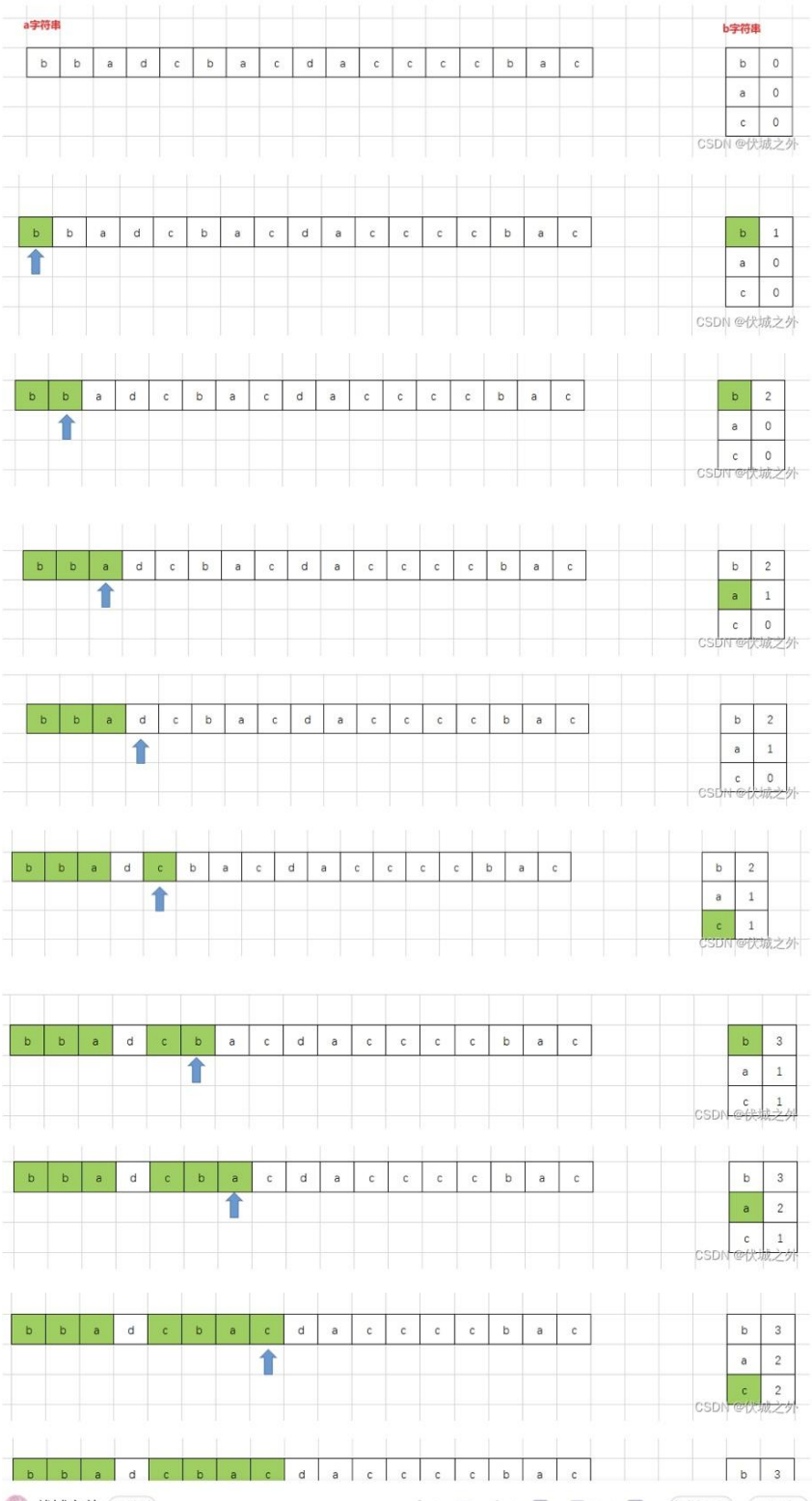
本题求解可以参考

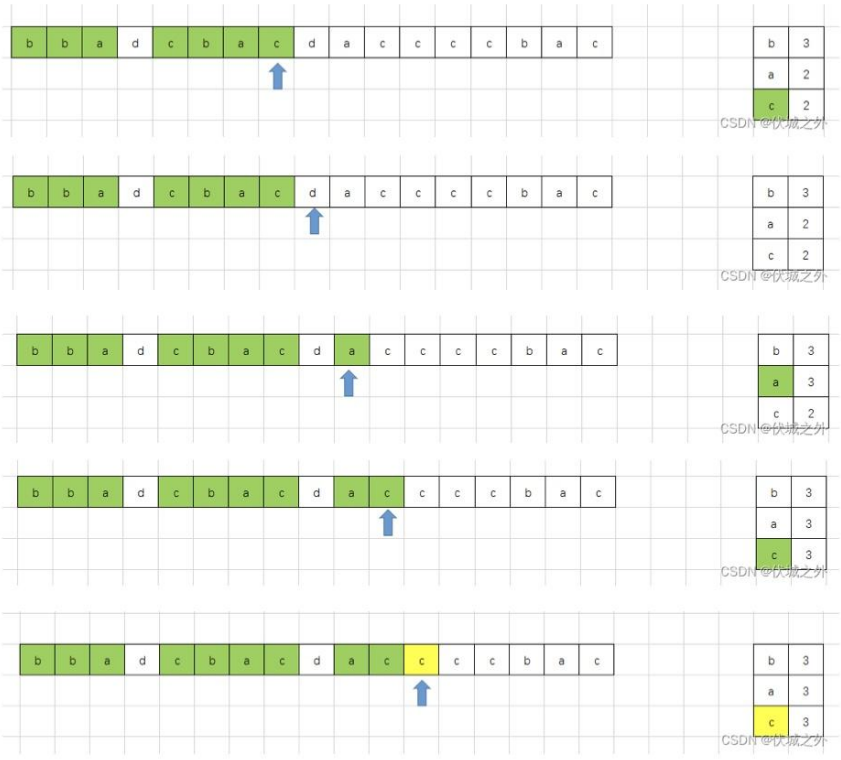
LeetCode - 1419 数青蛙_伏城之外的博客-CSDN博客

华为机试 - 数大雁_伏城之外的博客-CSDN博客

题目的用例不能说明问题，我们可以通过下面用例

```
bbadcbacdaccccbac
bac
```

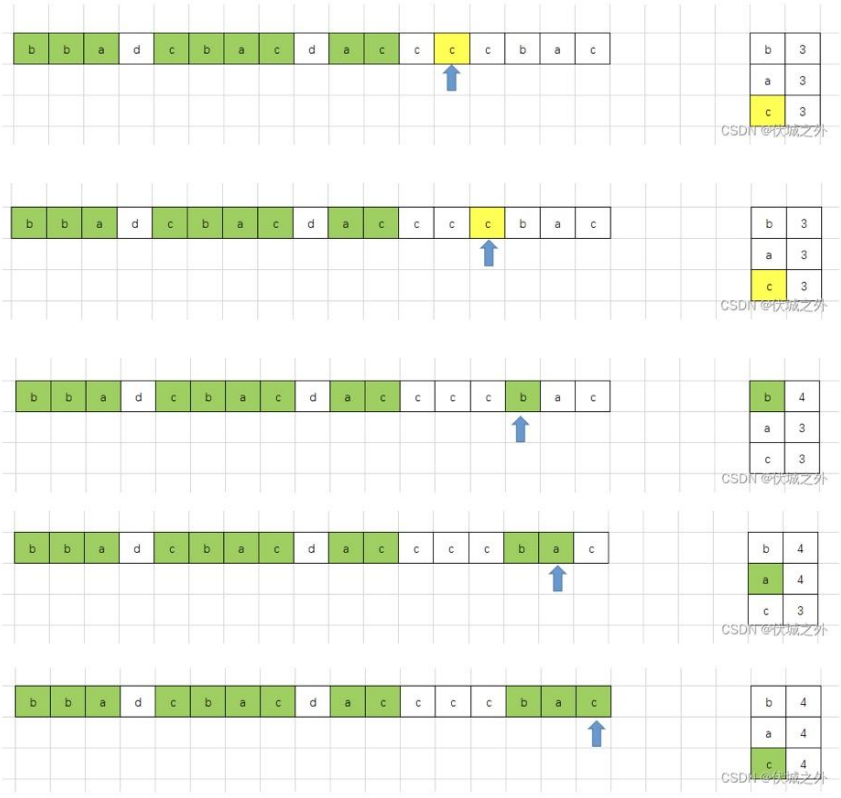




注意：统计时，a的数量应该小于等于b的数量，c的数量应该小于等于a的数量，这样才能满足顺序要求：

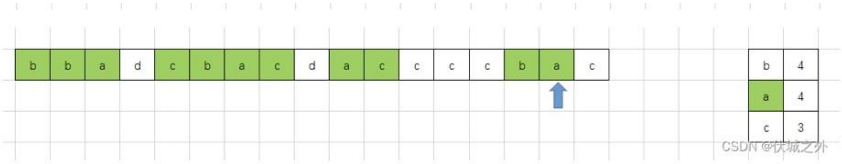
> 被挑选字母的相对先后顺序不能被改变

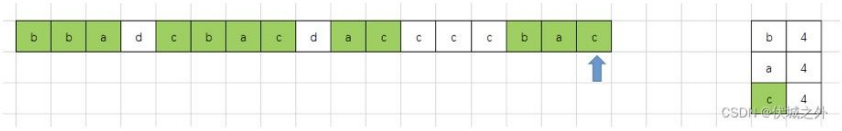
因此上面这步统计到的c不应该被计入。



由于c <= b <= a，因此有几个c，字符串a中就能挑选出几个字符串b。

上面算法只需要一次遍历，即可完成题解，时间复杂度O（n）





由于c <= b <= a，因此有几个c，字符串a中就能挑选出几个字符串b。

上面算法只需要一次遍历，即可完成题解，时间复杂度O（n）

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) {
    console.log(getResult(lines[0], lines[1]));
    lines.length = 0;
  }
});

function getResult(a, b) {
  // idxs对象记录字符串b中每个字符的索引
  const idxs = {};
  for (let i = 0; i < b.length; i++) {
    idxs[b[i]] = i;
  }

  // count对象用于记录遍历字符串a每个字符串过程中，统计到的符合顺序要求的字符串b中字符出现次数
  const count = new Array(b.length).fill(0);

  for (let c of a) {
    const idx = idxs[c];
    // 下面判断逻辑请看图解
    if (idx !== undefined && (idx === 0 || count[idx] < count[idx - 1])) {
      count[idx]++;
    }
  }

  return count.at(-1);
}
```

Java算法源码

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

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

    String a = sc.next();
    String b = sc.next();

    System.out.println(getResult(a, b));
  }

  public static int getResult(String a, String b) {
    // idxs对象记录字符串b中每个字符的索引
    HashMap<Character, Integer> idxs = new HashMap<>();
    for (int i = 0; i < b.length(); i++) {
      Character c = b.charAt(i);
      idxs.put(c, i); // B不会存在重复字母
    }

    // count对象用于记录遍历字符串a每个字符串过程中，统计到的符合顺序要求的字符串b中字符出现次数
    int[] count = new int[b.length()];
    for (int i = 0; i < a.length(); i++) {
      Character c = a.charAt(i);

      if (idxs.containsKey(c)) {
        int idx = idxs.get(c);
        // 下面判断逻辑请看图解
```

Java算法源码

复制

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

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

    String a = sc.next();
    String b = sc.next();

    System.out.println(getResult(a, b));
  }

  public static int getResult(String a, String b) {
    // idxs对象记录字符串b中每个字符的索引
    HashMap<Character, Integer> idxs = new HashMap<>();
    for (int i = 0; i < b.length(); i++) {
      Character c = b.charAt(i);
      idxs.put(c, i); // B不会存在重复字母
    }

    // count对象用于记录遍历字符串a每个字符串过程中，统计到的符合顺序要求的字符串b中字符出现次数
    int[] count = new int[b.length()];
    for (int i = 0; i < a.length(); i++) {
      Character c = a.charAt(i);

      if (idxs.containsKey(c)) {
        int idx = idxs.get(c);
        // 下面判断逻辑请看图解
        if (idx == 0 || count[idx] < count[idx - 1]) {
          count[idx]++;
        }
      }
    }

    return count[count.length - 1];
  }
}
```

Python算法源码

```python
# 输入获取
a = input()
b = input()


# 算法入口
def getResult(a, b):
    # idxs对象记录字符串b中每个字符的索引
    idxs = {}
    for i in range(len(b)):
        idxs[b[i]] = i

    # count对象用于记录遍历字符串a每个字符串过程中，统计到的符合顺序要求的字符串b中字符出现次数
    count = [0] * len(b)
    for c in a:
        idx = idxs.get(c)

        # 下面判断逻辑请看图解
        if idx is not None and (idx == 0 or count[idx] < count[idx - 1]):
            count[idx] += 1

    return count[-1]


# 算法调用
print(getResult(a, b))
```