383. 赎金信

力扣题目链接

给定一个赎金信 (ransom) 字符串和一个杂志(magazine)字符串，判断第一个字符串 ransom 能不能由第二个字符串 magazines 里面的字符构成。如果可以构成，返回 true ； 否则返回 false。

(题目说明：为了不暴露赎金信字迹，要从杂志上搜索各个需要的字母，组成单词来表达意思。杂志字符串中的每个字符只能在赎金信字符串中使用一次。)

给你两个字符串：`ransomNote` 和 `magazine` ，判断 `ransomNote` 能不能由 `magazine` 里面的字符构成。

如果可以，返回 `true` ；否则返回 `false` 。

`magazine` 中的每个字符只能在 `ransomNote` 中使用一次。

示例 1：

```
输入：ransomNote = "a", magazine = "b"
输出：false
```

示例 2：

```
输入：ransomNote = "aa", magazine = "ab"
输出：false
```

示例 3：

```
输入：ransomNote = "aa", magazine = "aab"
输出：true
```

思路

这道题目和242.有效的字母异位词 (opens new window)很像，242.有效的字母异位词 (opens new window)相当于求 字符串a 和 字符串b 是否可以相互组成 ，而这道题目是求 字符串a能否组成字符串b，而不用管字符串b 能不能组成字符串a。

本题判断第一个字符串ransom能不能由第二个字符串magazines里面的字符构成，但是这里需要注意两点。

- 第一点“为了不暴露赎金信字迹，要从杂志上搜索各个需要的字母，组成单词来表达意思” 这里*说明杂志里面的字母不可重复使用。*
- 第二点 “你可以假设两个字符串均只含有小写字母。” *说明只有小写字母*，这一点很重要

暴力法：

```python
class Solution:
    def canConstruct(self, ransomNote: str, magazine: str) -> bool:
        # 将 ransomNote 转为列表，方便删除字符
        ransom_list = list(ransomNote)

        # 遍历 magazine 中的每个字符
```

```
        for char in magazine:
            # 如果 ransom_list 还有字符，尝试匹配
            if char in ransom_list:
                ransom_list.remove(char)  # 删除第一个匹配的字符
                if not ransom_list:       # 提前退出优化
                    return True

        # 如果 ransom_list 为空，说明全部匹配成功
        return len(ransom_list) == 0
```

（版本一）**数组实现**：

```
class Solution:
    def canConstruct(self, ransomNote: str, magazine: str) -> bool:
        ransomNote_count = [0]*26
        magazine_count = [0] * 26
        for c in ransomNote:
            ransomNote_count[ord(c)-ord('a')] += 1

        for c in magazine:
            magazine_count[ord(c)-ord('a')] +=1

        for i in range(26):
            if(ransomNote_count[i] > magazine_count[i]):
                return False

        return True
```

> **Tip**
>
> **正确逻辑应该是**：
>
> > **只有当所有 26 个字母都满足 `ransomNote_count[i] <= magazine_count[i]` 时，才返回 `True`。**
> > 只要**有一个字母不满足**，就返回 `False`。
>
> 正确写法 1（显式循环）：
>
> ```
> for i in range(26):
>     if ransomNote_count[i] > magazine_count[i]:
>         return False
> return True  # 所有字母都满足条件
> ```
>
> 正确写法 2（使用 `all`）：
>
> ```
> return all(ransomNote_count[i] <= magazine_count[i] for i in range(26))
> ```

（版本二）使用defaultdict

```
from collections import defaultdict

class Solution:
    def canConstruct(self, ransomNote: str, magazine: str) -> bool:
```

```
        hashmap = defaultdict(int)

        for x in magazine:
            hashmap[x] += 1

        for x in ransomNote:
            value = hashmap.get(x)
            if not value:
                return False
            else:
                hashmap[x] -= 1

        return True
```

（版本四）使用Counter

```
from collections import Counter

class Solution:
    def canConstruct(self, ransomNote: str, magazine: str) -> bool:
        return not Counter(ransomNote) - Counter(magazine)
```

- `Counter(ransomNote) - Counter(magazine)` 的结果：
  - 如果 **为空**（即 `Counter()`），说明 `magazine` 中每个字符的数量都 ≥ `ransomNote` 中的需求 → **可以构造**；
  - 如果 **非空**，说明 `ransomNote` 中有某些字符“供不应求” → **不能构造。**
- `not Counter()` 是 `True`（因为空 `Counter` 在布尔上下文中为 `False`）；
- `not 非空Counter` 是 `False`。

（版本五）使用count

```
class Solution:
    def canConstruct(self, ransomNote: str, magazine: str) -> bool:
        return all(ransomNote.count(c) <= magazine.count(c) for c in
set(ransomNote))
```

(版本六）使用count(简单易懂)

```
class Solution:
    def canConstruct(self, ransomNote: str, magazine: str) -> bool:
        for char in ransomNote:
            if char in magazine and ransomNote.count(char) <=
magazine.count(char):
                continue
            else:
                return False
        return True
```