欢迎大家参与本项目，贡献其他语言版本的代码，拥抱开源，让更多学习算法的小伙伴们收益！

> 在哈希法中有一些场景就是为数组量身定做的。

383. 赎金信

https://leetcode-cn.com/problems/ransom-note/

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。)

注意：

你可以假设两个字符串均只含有小写字母。

canConstruct("a", "b") -> false
canConstruct("aa", "ab") -> false
canConstruct("aa", "aab") -> true

思路

这道题目和242.有效的字母异位词很像，242.有效的字母异位词相当于求 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

暴力解法

那么第一个思路其实就是暴力枚举了，两层for循环，不断去寻找，代码如下：

```
// 时间复杂度：O(n^2)
// 空间复杂度：O(1)
class Solution {
public:
    bool canConstruct(string ransomNote, string magazine) {
        for (int i = 0; i < magazine.length(); i++) {
            for (int j = 0; j < ransomNote.length(); j++) {
                // 在ransomNote中找到和magazine相同的字符
                if (magazine[i] == ransomNote[j]) {
                    ransomNote.erase(ransomNote.begin() + j); // ransomNote删除这个字符
                    break;
                }
            }
        }
        // 如果ransomNote为空，则说明magazine的字符可以组成ransomNote
        if (ransomNote.length() == 0) {
            return true;
        }
        return false;
    }
};
```

没有体现注意的两点啊，
不需要特殊考虑一些情况吗？

这里时间复杂度是比较高的，而且里面还有一个字符串删除也就是erase的操作，也是费时的，当然这段代码也可以过这道题。

哈希解法

因为题目所只有小写字母，那可以采用空间换取时间的哈希策略， 用一个长度为26的数组还记录magazine里字母出现的次数。

然后再用ransomNote去验证这个数组是否包含了ransomNote所需要的所有字母。

依然是数组在哈希法中的应用。

一些同学可能想，用数组干啥，都用map完事了，**其实在本题的情况下，使用map的空间消耗要比数组大一些的，因为map要维护红黑树或者哈希表，而且还要做哈希函数，是费时的！数据量大的话就能体现出来差别了。 所以数组更加简单直接有效！**

代码如下：

```
// 时间复杂度：O(n)
// 空间复杂度：O(1)
class Solution {
public:
    bool canConstruct(string ransomNote, string magazine) {
        int record[26] = {0};
        for (int i = 0; i < magazine.length(); i++) {
            // 通过recode数据记录 magazine里各个字符出现次数
            record[magazine[i]-'a'] ++;
        }
        for (int j = 0; j < ransomNote.length(); j++) {
            // 遍历ransomNote，在record里对应的字符个数做--操作
            record[ransomNote[j]-'a']--;
            // 如果小于零说明ransomNote里出现的字符，magazine没有
            if(record[ransomNote[j]-'a'] < 0) {
                return false;
            }
        }
        return true;
    }
};
```

其他语言版本

Java：

```
class Solution {
    public boolean canConstruct(String ransomNote, String magazine) {
        //记录杂志字符串出现的次数
        int[] arr = new int[26];
        int temp;
        for (int i = 0; i < magazine.length(); i++) {
            temp = magazine.charAt(i) - 'a';
            arr[temp]++;
        }
        for (int i = 0; i < ransomNote.length(); i++) {
            temp = ransomNote.charAt(i) - 'a';
            //对于金信中的每一个字符都在数组中查找
            //找到相应位减一，否则找不到返回false
            if (arr[temp] > 0) {
                arr[temp]--;
            } else {
                return false;
            }
        }
        return true;
    }
}
```

Python写法一（使用数组作为哈希表）：

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

```
        arr = [0] * 26

        for x in magazine:
            arr[ord(x) - ord('a')] += 1

        for x in ransomNote:
            if arr[ord(x) - ord('a')] == 0:
                return False
            else:
                arr[ord(x) - ord('a')] -= 1

        return True
```

Python写法二（使用defaultdict）：

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

        from collections import defaultdict

        hashmap = defaultdict(int)

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

        for x in ransomNote:
            value = hashmap.get(x)
            if value is None or value == 0:
                return False
            else:
                hashmap[x] -= 1

        return True
```

Python写法三：

```
class Solution(object):
    def canConstruct(self, ransomNote, magazine):
        """
        :type ransomNote: str
        :type magazine: str
        :rtype: bool
        """

        # use a dict to store the number of letter occurance in ransomNote
        hashmap = dict()
        for s in ransomNote:
            if s in hashmap:
                hashmap[s] += 1
            else:
                hashmap[s] = 1

        # check if the letter we need can be found in magazine
        for l in magazine:
```

```
            if l in hashmap:
                hashmap[l] -= 1

        for key in hashmap:
            if hashmap[key] > 0:
                return False

        return True
```

Go：

```
func canConstruct(ransomNote string, magazine string) bool {
    record := make([]int, 26)
    for _, v := range magazine {
        record[v-'a']++
    }
    for _, v := range ransomNote {
        record[v-'a']--
        if record[v-'a'] < 0 {
            return false
        }
    }
    return true
}
```

javaScript:

```
/**
 * @param {string} ransomNote
 * @param {string} magazine
 * @return {boolean}
 */
var canConstruct = function(ransomNote, magazine) {
    const strArr = new Array(26).fill(0),
        base = "a".charCodeAt();
    for(const s of magazine) {
        strArr[s.charCodeAt() - base]++;
    }
    for(const s of ransomNote) {
        const index = s.charCodeAt() - base;
        if(!strArr[index]) return false;
        strArr[index]--;
    }
    return true;
};
```

- 作者微信：程序员Carl
- B站视频：代码随想录
- 知识星球：代码随想录