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！

> 数组就是简单的哈希表，但是数组的大小可不是无限开辟的

242.有效的字母异位词

https://leetcode-cn.com/problems/valid-anagram/

给定两个字符串 s 和 t ，编写一个函数来判断 t 是否是 s 的字母异位词。

示例 1: 输入: s = "anagram", t = "nagaram" 输出: true

示例 2: 输入: s = "rat", t = "car" 输出: false

说明: 你可以假设字符串只包含小写字母。

思路

先看暴力的解法，两层for循环，同时还要记录字符是否重复出现，很明显时间复杂度是 O(n^2)。

暴力的方法这里就不做介绍了，直接看一下有没有更优的方式。

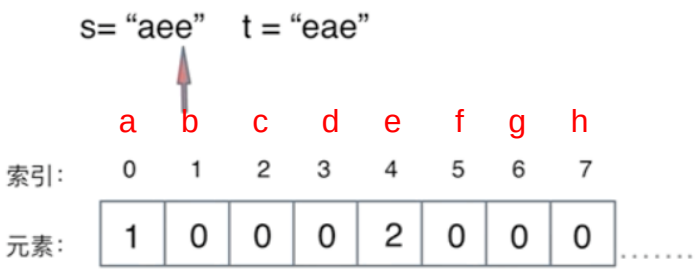
数组其实就是一个简单哈希表，而且这道题目中字符串只有小写字符，那么就可以定义一个数组，来记录字符串s里字符出现的次数。

如果对哈希表的理论基础关于数组，set，map不了解的话可以看这篇：关于哈希表，你该了解这些！

需要定义一个多大的数组呢，定一个数组叫做record，大小为26 就可以了，初始化为0，因为字符a到字符z的ASCII也是26个连续的数值。 97-122

为了方便举例，判断一下字符串s= "aee", t = "eae"。

操作动画如下：



定义一个数组叫做record用来上记录字符串s里字符出现的次数。

需要把字符映射到数组也就是哈希表的索引下表上，**因为字符a到字符z的ASCII是26个连续的数值，所以字符a映射为下表0，相应的字符z映射为下表25。**

再遍历 字符串s的时候，**只需要将 s[i] - ‘a’ 所在的元素做+1 操作即可，并不需要记住字符a的ASCII，只要求出一个相对数值就可以了。** 这样就将字符串s中字符出现的次数，统计出来了。

那看一下如何检查字符串t中是否出现了这些字符，同样在遍历字符串t的时候，对t中出现的字符映射哈希表索引上的数值再做-1的操作。

那么最后检查一下，**record数组如果有的元素不为零0，说明字符串s和t一定是谁多了字符或者谁少了字符，return false。**

最后如果record数组所有元素都为零0，说明字符串s和t是字母异位词，return true。

时间复杂度为O(n)，空间上因为定义是的一个常量大小的辅助数组，所以空间复杂度为O(1)。

C++ 代码如下：

```
class Solution {
public:
    bool isAnagram(string s, string t) {
        int record[26] = {0};                    数组初始化
        for (int i = 0; i < s.size(); i++) {
            // 并不需要记住字符a的ASCII，只要求出一个相对数值就可以了
            record[s[i] - 'a']++;
        }
        for (int i = 0; i < t.size(); i++) {
            record[t[i] - 'a']--;
        }
        for (int i = 0; i < 26; i++) {
            if (record[i] != 0) {
                // record数组如果有的元素不为零0，说明字符串s和t 一定是谁多了字符或者谁少了字符
                return false;
            }
        }
        // record数组所有元素都为零0，说明字符串s和t是字母异位词
        return true;
    }
};
```

其他语言版本

Java：

```
class Solution {
    public boolean isAnagram(String s, String t) {

        int[] record = new int[26];
        for (char c : s.toCharArray()) {
            record[c - 'a'] += 1;
        }
        for (char c : t.toCharArray()) {
            record[c - 'a'] -= 1;
        }
        for (int i : record) {
            if (i != 0) {
                return false;
            }
        }
        return true;
    }
}
```

Python：

```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        record = [0] * 26
        for i in range(len(s)):
            #并不需要记住字符a的ASCII，只要求出一个相对数值就可以了
            record[ord(s[i]) - ord("a")] += 1
        print(record)
        for i in range(len(t)):
            record[ord(t[i]) - ord("a")] -= 1
        for i in range(26):
            if record[i] != 0:
                #record数组如果有的元素不为零0，说明字符串s和t 一定是谁多了字符或者谁少了字符。
                return False
        return True
```

Python写法二（没有使用数组作为哈希表，只是介绍defaultdict这样一种解题思路）：

```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        from collections import defaultdict

        s_dict = defaultdict(int)
        t_dict = defaultdict(int)

        for x in s:
            s_dict[x] += 1

        for x in t:
            t_dict[x] += 1
```

```
        return s_dict == t_dict
```

Go：

```
func isAnagram(s string, t string) bool {
    if len(s)!=len(t){
        return false
    }
    exists := make(map[byte]int)
    for i:=0;i<len(s);i++{
        if v,ok:=exists[s[i]];v>=0&&ok{
            exists[s[i]]=v+1
        }else{
            exists[s[i]]=1
        }
    }
    for i:=0;i<len(t);i++{
        if v,ok:=exists[t[i]];v>=1&&ok{
            exists[t[i]]=v-1
        }else{
            return false
        }
    }
    return true
}
```

javaScript:

```
/**
 * @param {string} s
 * @param {string} t
 * @return {boolean}
 */
var isAnagram = function(s, t) {
    if(s.length !== t.length) return false;
    const resSet = new Array(26).fill(0);
    const base = "a".charCodeAt();
    for(const i of s) {
        resSet[i.charCodeAt() - base]++;
    }
    for(const i of t) {
        if(!resSet[i.charCodeAt() - base]) return false;
        resSet[i.charCodeAt() - base]--;
    }
    return true;
};
```

相关题目

- 383.赎金信
- 49.字母异位词分组

- 438.找到字符串中所有字母异位词

- 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