有效的字母异位词

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。

进阶:
如果输入字符串包含 unicode 字符怎么办？你能否调整你的解法来应对这种情况?

方法一：

暴力求解

给字符串s和t排序，排完后，依次判断s[i] == t[i], 都相等，返回true，否则返回false

```python
def isAnagram(self, s, t):
        return True if sorted(s)==sorted(t) else False
```

方法二：利用hash表

C语言：

```c
bool isAnagram(char * s, char * t){
    int n = strlen(s), m = strlen(t);
    if(n != m)
        return false;
            // 设置一个数组，保存26个英文字母
    int a[26] = {0};
    for (int i=0; i<n; i++){
            //如果在字符串s里有这个字母，就++
        a[s[i] - 'a']++;
            //如果在字符串t里有这个字母，就--
         a[t[i] - 'a']--;
     }
            //如果s和t对应的字母数量相同，a[i]就为0，不为0 说明不想等
     for(int i =0; i<26; i++){
         if (a[i] != 0){
             return false;
         }
```

```
    }
    return true;
}
```

存值dict或set

python :

一个 Counter 是一个 dict 的子类，用于计数可哈希对象。它是一个集合，元素像字典键(key)一样存储，它们的计数存储为值。

1 利用collection.counter(s) --> 可以计算s里每个元素的数量

```
def isAnagram(self, s, t):
        return collections.Counter(s) == collections.Counter(t)
```

2 python3 用set保存元素，如果里面元素相等则在比较对应元素是否相等

```
bool isAnagram(char * s, char * t){
    int n = strlen(s), m = strlen(t);
    if(n != m)
        return false;
    int a[26] = {0};
    for (int i=0; i<n; i++){
        a[s[i] - 'a']++;
        a[t[i] - 'a']--;
    }
    for(int i =0; i<26; i++){
        if (a[i] != 0){
            return false;
        }
    }
    return true;
}
```

49 字母异位词分组

给定一个字符串数组，将字母异位词组合在一起。字母异位词指字母相同，但排列不同的字符串。

示例:

```
输入: ["eat", "tea", "tan", "ate", "nat", "bat"]
输出:
[
  ["ate","eat","tea"],
  ["nat","tan"],
  ["bat"]
]
```

说明:

- 所有输入均为小写字母。
- 不考虑答案输出的顺序。

方法1

维护一个ans：{string -> list} 的映射，key是排序后的字符串

python中，将存储值转换为元组

collections.defaultdict，将 键值，转换为键，列表

["eat", "tea", "tan", "ate", "nat", "bat"]

{('e','a','t'),["eat","tea"...]}

```
def groupAnagrams(self, strs):
        """
        :type strs: List[str]
        :rtype: List[List[str]]
        """
        ans = collections.defaultdict(list)
        for s in strs:
            ans[tuple(sorted(s))].append(s)
        return ans.values()
```

方法2 利用计数分类

{(5,0,...),["eat","tea"...]}

```
def groupAnagrams(self, strs):
        """
        :type strs: List[str]
        :rtype: List[List[str]]
        """
        ans = collections.defaultdict(list)
        for s in strs:
            count = [0] * 26   # 这里记录26个英文字母
            for c in s:
                 count[ord(c) - ord('a')] += 1      #把字符串中的计数+1
             ans[tuple(count)].append(s)
         return ans.values()
```