

# [LeetCode] Valid Anagram - 字符串排序比较系列

原创

Eastmount

2015-09-14 07:01:49

👁

2692

☆

收藏

版权

分类专栏:

LeetCode

文章标签:

leetcode

字符串处理



## Python+TensorFlow人工智能

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相关算法。前期介绍安装流程、基础语法、神...



Eastmount

¥9.90

订阅博主

### 题目概述:

Given two strings *s* and *t*, write a function to determine if *t* is an anagram of *s*.  
For example,

*s* = "anagram", *t* = "nagaram", return true.

*s* = "rat", *t* = "car", return false.

**Note:** You may assume the string contains only lowercase alphabets.

### 解题方法:

该题意是比较两个字符串*s*和*t*，其中*t*是次序打乱的字符串，如果两个字符串相同则返回true，否则false。方法包括：[\(参考\)](#)

#### 方法一

最简单的方法就是字符串*s*和*t*分别排序，在比较两个字符串是否相同。但是会报错  
TLE-Time Limit Exceeded

同样采用选择排序每次比较最小字符，不同则跳出循环返回false也TLE。

```
bool isAnagram(char* s, char* t) {
    int ls,lt;    //字符串长度
    int i,j;
    char ch;
    if(s==NULL&& t==NULL)
        return true;
    ls=strlen(s);
    lt=strlen(t);
    if(ls!=lt)
        return false;

    //方法一 排序后判断字符串是否相等
    for(i=0; i<ls; i++) {
        for(j=i+1; j<ls; j++) {
            if(s[i]>s[j]) {
                ch=s[i];
                s[i]=s[j];
                s[j]=ch;
            }
        }
    }
```

```

        s[i]=s[j];
                                s[j]=ch;
    }
    if(t[i]>=t[j]) {
        ch=t[i];
        t[i]=t[j];
        t[j]=ch;
    }
}
}
if(strcmp(s,t)==0)
    return true;
else
    return false;
}

```

## 方法二

后来百度下发现如果采用Java代码，通过调用内部的sort排序则会AC，但个人不喜欢调用内部函数的方法。

```

public class Solution {
    public boolean isAnagram(String s, String t) {
        char[] sArr = s.toCharArray();
        char[] tArr = t.toCharArray();
        Arrays.sort(sArr);
        Arrays.sort(tArr);
        return String.valueOf(sArr).equals(String.valueOf(tArr));
    }
}

```

## C++调用sort排序代码如下：

```

class Solution {
public:
    bool isAnagram(string s, string t) {
        sort(s.begin(), s.end());
        sort(t.begin(), t.end());
        return s == t;
    }
};

```

## 方法三

计算字符串字母个数，比较值都相同则true，否则返回false。

#### 方法四 (强推)

通过一个长度为26的整形数组，对应英文中的26个字母a-z。从前向后循环字符串s和t，s中出现某一字母则在该字母在数组中对应的位置上加1，t中出现则减1。如果在s和t中所有字符都循环完毕后，整型数组中的所有元素都为0，则可认为s可由易位构词生成t。

```
bool isAnagram(char* s, char* t) {
    int ls,lt;    //字符串长度
    int i;
    int num[26]={0};

    if(s==NULL&& t==NULL)
        return true;
    ls=strlen(s);
    lt=strlen(t);
    if(ls!=lt)
        return false;

    //方法四 计算字母个数 s中出现+1,t中出现-1,整个数组26个数都为0时则表示相同
    for(i=0; i<ls; i++) {
        num[s[i]-'a']++;
        num[t[i]-'a']--;
    }
    for(i=0; i<26; i++) {
        if(num[i]!=0)
            return false;
    }
    return true;
}
```

而且最后的时间结果也比较优秀：C++调用sort代码-76ms；Java调用sort代码-288ms；C语言计算字母个数-0ms。



(By:Eastmount 2015-9-14 清晨7点半 <http://blog.csdn.net/eastmount/>)