242 Valid Anagram

2018年4月7日 15:51

Question:

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.
Follow up:
What if the inputs contain unicode characters? How would you adapt your solution to such case?

来自 <https://leetcode.com/problems/valid-anagram/description/>

给定两个字符串 s 和 t，编写一个函数来判断 t 是否是 s 的一个字母异位词。
例如，
s = "anagram", t = "nagaram"，返回 true
s = "rat", t = "car"，返回 false
注意:
假定字符串只包含小写字母。
提升难度:
输入的字符串包含 unicode 字符怎么办？你能能否调整你的解法来适应这种情况？

Solution for Python3:

```python
class Solution1:
    def isAnagram(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        d1, d2 = {}, {}
        for i in s:
            if i in d1:
                d1[i] += 1
            else:
                d1[i] = 1
        for j in t:
            if j in d2:
                d2[j] += 1
            else:
                d2[j] = 1
        return d1 == d2

```

```python
class Solution2:
    def isAnagram(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        if len(s) != len(t):
            return False
        d1, d2 = {}, {}
        for item in zip(s, t):
            d1[item[0]] = d1.get(item[0], 0) + 1
            d2[item[1]] = d2.get(item[1], 0) + 1
        return d1 == d2

class Solution3:
    def isAnagram(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        if len(s) != len(t):
            return False
        d = [0] * 26
        for i in range(len(s)):
            d[ord(s[i]) - ord('a')] += 1
            d[ord(t[i]) - ord('a')] -= 1
        for i in d:
            if i:
                return False
        return True

class Solution4:
    def isAnagram(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        return sorted(s) == sorted(t)
```

Solution for C++:

```cpp
class Solution1 {
public:
    bool isAnagram(string s, string t) {
```

```
4           if (s.length() != t.length()) {
5               return false;
6           }
7           int n = s.length();
8           unordered_map<char, int> counts;
9           for (int i = 0; i < n; i++) {
10              counts[s[i]]++;
11              counts[t[i]]--;
12          }
13          for (auto count : counts) {
14              if (count.second)
15                  return false;
16          }
17          return true;
18      }
19  };
20
21  class Solution2 {
22  public:
23      bool isAnagram(string s, string t) {
24          if (s.length() !+ t.length())
25              return false;
26          int n = s.length();
27          int counts[26] = {0};
28          for (int i = 0; i < n; i++) {
29              counts[s[i] - 'a']++;
30              counts[t[i] - 'a']--;
31          }
32          for (int i = 0; i < 26; i++)
33              if (counts[i])
34                  return false;
35          return true;
36      }
37  };
38
39  class Solution3 {
40  public:
41      bool isAnagram(string s, string t) {
42          sort(s.begin(), s.end());
43          sort(t.begin(), t.end());
44          return s == t;
45      }
46  };
```