# 242. Valid Anagram

Given two strings `s` and `t`, return `true` *if* `t` *is an anagram of* `s`*, and* `false` *otherwise*

**Example 1:**

**Input:** s = "anagram", t = "nagaram"
**Output:** true

**Example 2:**

**Input:** s = "rat", t = "car"
**Output:** false

```
def isAnagram(self, s: str, t: str) -> bool:
        if len(s)!=len(t):
            return False
        counter = [0]*26
        for i in range(len(s)):
            counter[ord(s[i])-ord('a')] = counter[ord(s[i])-ord('a')]+1
            counter[ord(t[i])-ord('a')] = counter[ord(t[i])-ord('a')]-1
        for ele in counter:
            if ele!=0:
                return False
        return True


#         if len(s)!=len(t):
#             return False
#         s = sorted(s)
#         t = sorted(t)
#         for i in range(0,len(t)):
#             if s[i]!=t[i]:
#                 return False

#         return True
```

**Follow up**

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

**Answer**

Use a hash table instead of a fixed size counter. Imagine allocating a large size array to fit the entire range of unicode characters, which could go up to more than 1 million. A hash table is a more generic solution and could adapt to any range of characters.