Problem Link:

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

Problem Description:

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

`s` and `t` consist of lowercase English letters.

Problem Approach:

There are two approaches, one is language independent and the other is python specific! We'll use frequency counting in both the approaches. In python, there's an internal library for the same. The Time complexity of this would be O(m+n) and the space complexity would be O(1)

Sample Test Case:

Input: s = "anagram", t = "nagaram"

Output: true

Solution:

We start by checking the lengths of the strings. Obviously, if the strings don't have the same amount of characters, they cannot be anagrams of each other. Then, we create an array of size 26, as there are 26 letters in the alphabets where we'll store the frequency of the characters in the string. We also would need to make both of these input strings lowercase, but it's mentioned in the problem description itself that the input strings would be lowercase only. So, let's visualize this: What's the ord(character) or ascii/unicode value of 'a'? 97! Right. So what will be 97-97? 0! So, 'a' goes to the 0th index of the count. Similarly, in the case of 'z', it would be 122! 122-97 gives 25, and hence, it fits the end of the array. Since there can be no other characters than alphabets, we're good to go with this. Now, the problem is easy! We add one to the frequency of every character in string s, and then, we remove one from the frequency of every character in string t. By the end, we should have an array of 0s if the two strings are anagrams. If that's the case, we return True, else we return False.

Visualize this as an empty bucket. You fill the bucket with a certain amount of mugs of water. And then, remove that many mugs from the bucket. By the end of this, the bucket must be empty. That's what we did in this problem.

Code (Python):

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

Counter is an inbuilt function in python that returns a dictionary of an object (String in this case, where the keys are characters, and the values are their frequencies). So, we compare the counter objects of both the strings, and return if they're equal or not. That'll do the same thing which we did previously, but with lesser code.

```
def isAnagram(self, s: str, t: str) -> bool:
       countS, countT = Counter(s),Counter(t)
       return countS == countT
```