242. Valid Anagram

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

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

Constraints:

- `1 <= s.length, t.length <= 5 * 104`
- `s` and `t` consist of lowercase English letters.

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

```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        # If lengths are different, they cannot be anagrams
        if len(s) != len(t):
            return False

        # Use a dictionary to count character frequencies in bo
        char_count = {}

        for char in s:
            char_count[char] = char_count.get(char, 0) + 1

        for char in t:
            if char not in char_count:
                return False
            char_count[char] -= 1
            if char_count[char] < 0:
                return False

        return all(count == 0 for count in char_count.values())
```

The following code checks if the length of each string matches first before comparing the character frequency.

We use char_count to count characters for keys and values for their frequeuncies

using

```
char_count[char] = char_count.get(char, 0) + 1
```

We fetch a key to retrieve its count, otherwise if it doesn’t exist we increase the count of the character

Important to note that **.get()** is used to return the value of the item of the specified key

After iterating through string s

It will check for characters in string t

For every character it see’s it will decrement the count as it goes through the string

So it should return a 0 if its an anagram

This code is O(n) both time complexity and space complexity as it linearly scales with the size of the string.