242. Valid Anagram

Initial thoughts

1. To solve this question..we can use hashmap(dict) and get the count of each char in the string..from both s and t
2. Later if the count of char's in t and s are equal ..then they are anagrams

Solution :

1.
```python
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        count = defaultdict(int)

        for x in s:
            count[x] += 1

        for x in t:
            count[x] -= 1

        for val in count.values():
            if val != 0:
                return False
        return True
```
2. Here the output of count is below:
3.
```
defaultdict(<class 'int'>, {'a': 0, 'n': 0, 'g': 0, 'r': 0, 'm': 0})
```
4. Basically here we are iterating the string **s** and storing its frequency in the hashmap count
5. In the next iteration we iterate the string **t** and decrement its frequency in the count
6. So if every value in count is '0' then it is a anagram

Solution2 (Hash Table using an array):

1.
```python
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        count = [0] * 26

        # Count the frequency of characters in string s
        for x in s:
            count[ord(x) - ord('a')] += 1

        # Decrement the frequency of characters in string t
        for x in t:
            count[ord(x) - ord('a')] -= 1

        # Check if any character has non-zero frequency
        for val in count:
            if val != 0:
                return False

        return True
```
2. Here in this code..we are using array as hashmap
3. We initialize an array count with len of alphabets
4. Now we iterate the string **s** and count the occurrence of a char
5. we do same as solution 1