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

**Valid Anagram**

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

**An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.**

**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.**

**Method 1:[Brute Force]**

Check if both strings have same length.

For each character in s, search the char in t and replace it with a unique character.

If a character is not found then they are not anagrams.

If all char of s are found in t then they are anagrams.

Time Complexity: O(n2) *[ O(s\*t) ]*

Space Complexity: O(1) or O(t) *[ depending on t can be modified or not ]*

**Method 2:**

Check if both strings have same length.

Sort both strings (nlogn) and compare the two(n).

Time Complexity: O(nlogn)

Space Complexity: O(1)

**Method 3:**

Check if length of s is same as that of t . Return false if they are not same.

Use a frequency counter vector array initialised with zero.

Increment for each char in s and decrement for each character in t.

If they are anagrams they should cancel out and counter should be zero for each char in the end.

Time Complexity: O(n) *[traversing first string O(s)]*

Space Complexity: O(1) *[constant extra space ]*

bool isAnagram(string s, string t) {

        if(s.size()!=t.size())

            return false;

vector<int> v(26,0); // initialising with 0 explicitly improves performance

        for(int i=0;i<s.size(); i++){

            v[s[i]-'a']++;

            v[t[i]-'a']--;

        }

        for(int i=0;i<26; i++){

            if(v[i]!=0)

                return false;

        }

        return true;

    }