<https://leetcode.com/problems/permutation-in-string/>

**Permutation in String**

**Given two strings s1 and s2, return true if s2 contains a permutation of s1, or false otherwise.**

**In other words, return true if one of s1's permutations is the substring of s2.**

Example 1:

Input: s1 = "ab", s2 = "eidbaooo"

Output: true

Explanation: s2 contains one permutation of s1 ("ba").

Example 2:

Input: s1 = "ab", s2 = "eidboaoo"

Output: false

Constraints:

1 <= s1.length, s2.length <= 104

s1 and s2 consist of lowercase English letters**.**

**Method 1: (Brute Force)**

Find every substring s of length n1 in s2 and compare s and s1

Time Complexity: O(n3 )*[for i element all j=>n2]*

Space Complexity: O(1)

**Method 2: (Sliding window)**

1. Create freq counter vector for s1 and freq counter vector of first n1 characters of s2
2. Compare both vectors : (C++ also allows direct comparison of two vectors)
   1. if equal s2 contains a permutation of s1
   2. else slide window by 1 place and update freq counter of s2 and repeat Step 2.

Time Complexity: O(n) *[]*

Space Complexity: O(1) *[ freq counters are of constant size 26]*

bool isPerm(vector<int>& freqS1,vector<int>& freqS2){

        for(int i=0;i<26;i++){

            if(freqS1[i]-freqS2[i])

             return false;

        }

        return true;

    }

    bool checkInclusion(string s1, string s2) {

        int n1=s1.size(), n2=s2.size();

        if(n1>n2) return false;

        vector<int> freqS1(26), freqS2(26);

        for(char x: s1)

            freqS1[x-'a']++;

        int l=0,r=0;

        while(r<n1){

            freqS2[s2[r]-'a']++;

            r++;

        }

        if(isPerm(freqS1, freqS2)) return true;;

        while(r<n2){

            freqS2[s2[l++]-'a']--;

            freqS2[s2[r]-'a']++;

            if(isPerm(freqS1, freqS2)) return true;

            r++;

        }

        return false;

    }