# 567. Permutation in String

Add to List

Description | HintsSubmissionsSolutions

- Total Accepted: 3167
- Total Submissions: 8667
- Difficulty: Medium
- Contributors:fallcreek

Given two strings **s1** and **s2**, write a function to return true if **s2** contains the permutation of **s1**. In other words, one of the first string's permutations is the **substring** of the second string.

**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
```

**Note:**

1. The input strings only contain lower case letters.
2. The length of both given strings is in range [1, 10,000].

Subscribe to see which companies asked this question.

```
//c++
bool equal(vector<int> &a, vector<int> &b, int sz)
{
    bool ret = false;
    for(int i=0;i<sz;i++)
    {
        if(a[i]!=b[i]) return ret;
    }
    return !ret;
}

bool checkInclusion(string s1, string s2) {

        int a[(int)s1.size()];int b[(int)s2.size()];
        for(int i=0;i<(int)s1.size();i++)
        {
            a[i] = s1[i]-'a';
        }
        for(int i=0;i<(int)s2.size();i++)
        {
            b[i] = s2[i]-'a';
        }
        vector<int> target(26,0);
        vector<int> window(26,0);
        for(auto tmp:a)
        {
            target[tmp]+=1;
        }
        for(int i=0;i<(int)s2.size();i++)
        {
```

```
            window[b[i]]++;
            if(i>=s1.size())
            {
                window[b[i-s1.size()]]--;
            }
            if(equal(target,window,26)) return true;
        }
        return false;
}
```

//python

For each `window` representing a substring of `s2` of length `len(s1)`, we want to check if the count of the window is equal to the count of `s1`. Here, the count of a string is the list of: [the number of `a`'s it has, the number of `b`'s,... , the number of `z`'s.]

We can maintain the window by deleting the value of s2[i - len(s1)] when it gets larger than `len(s1)`. After, we only need to check if it is equal to the target. Working with list values of [0, 1,..., 25] instead of 'a'-'z' makes it easier to count later.

```
def checkInclusion(self, s1, s2):
    A = [ord(x) - ord('a') for x in s1]
    B = [ord(x) - ord('a') for x in s2]

    target = [0] * 26
    for x in A:
        target[x] += 1

    window = [0] * 26
    for i, x in enumerate(B):
        window[x] += 1
        if i >= len(A):
            window[B[i - len(A)]] -= 1
        if window == target:
            return True
    return False
```