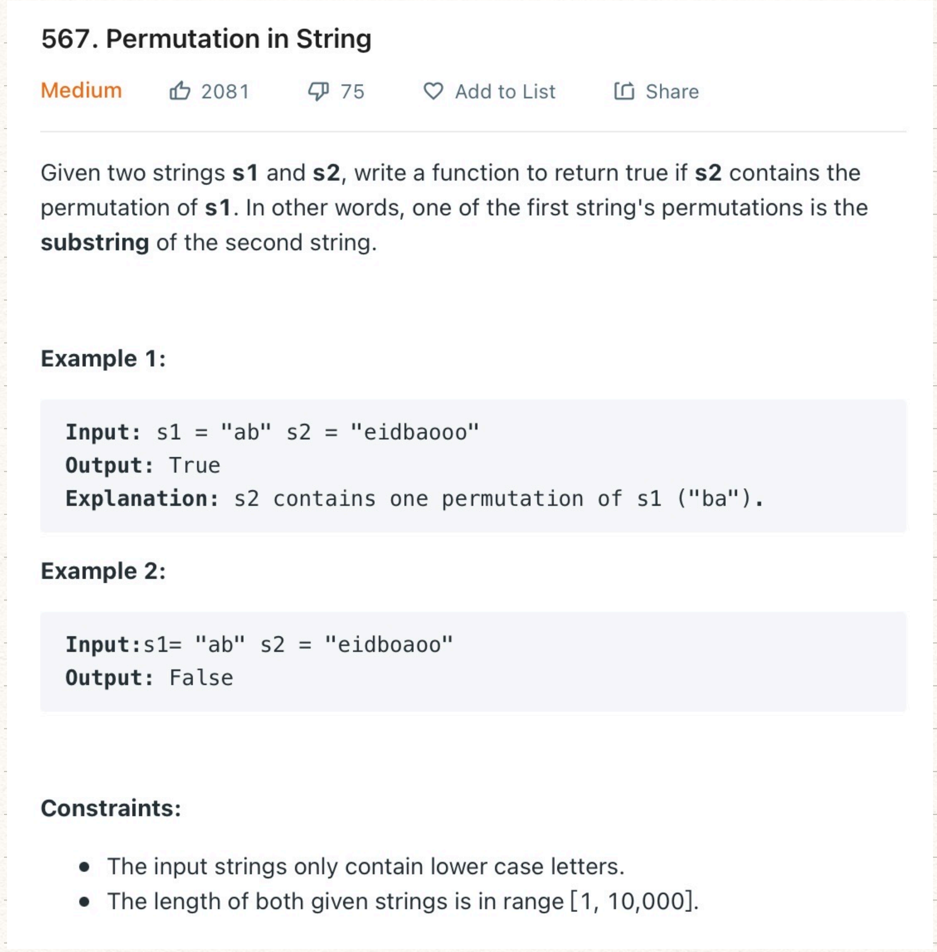

567. Permutation in String

Medium 2081 75 Add to List Share

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
```

Constraints:

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

While add right, update window counter.
When window.length == s1.length, shrink window.
While add left, reduce window counter.
After shrink window, update the final result.

```cpp
class Solution {
public:
    bool checkInclusion(string s1, string s2) {
        if (s1.size() > s2.size()) {
            return false;
        }

        unordered_map<char, int> need, window;
        for (char c : s1) {
            need[c]++;
        }

        int left = 0, right = 0;
        int valid = 0;

        for (int i = 0; i < s1.size(); i++) {
            // c is the char adding to window
            char c = s2[right];
            // move the right side of window
            right++;
            // update the window counter and valid
            if (need.count(c)) {
                window[c]++;
                if (window[c] == need[c]) {
                    valid++;
                }
            }
        }

        if(valid >= need.size()) {
            return true;
        }

        while (right < s2.size()) {
            // c is the char adding to window
            char c = s2[right];
            // move the right side of window
            right++;
            // update the window counter and valid
            if (need.count(c)) {
                window[c]++;
                if (window[c] == need[c]) {
                    valid++;
                }
            }

    //        cout << "window: [" << left << "," << right << ")" << endl;

            // d is the char removing to window
            char d = s2[left];
            // move the left side of window
            left++;
            // update the window counter and valid
            if (need.count(d)) {
                if (window[d] == need[d]) {
                    valid--;
                }
                window[d]--;
            }

            if(valid >= need.size()) {
                return true;
            }
        }

        return false;
    }
};
```