Permutation In String - QOTD 4 Feb 23

Leetcode Link : Click

Intuition

We will use sliding window approach to solve this question.

Complexity

- Time complexity: $O(n * 26)$ - where n is length of str2 (longer string) and for each character we check after inserting it whether v1 == v2, so in checking them worst case time will be O(26) for each char of s2
- Space complexity: $O(52)$ - v1 and v2 of 26 blocks each

Code

```
class Solution {

    private: // Fun.2 : checks whether 2 vectors exactly have same values or not

    bool areVectorsEqual(vector<int> &v1, vector<int> &v2){

        // we need to compare the frequencies of all the characters of v1 with those characters in v2, so for that, if for a index 'i' if freq of character is equal in both the vectors, then keep doing, if not then return false

        for(int i = 0; i < 26; i++)

            if(v1[i] != v2[i]) return false;


        // if both checked and no fault found, return true

        return true;



    }



public:

    // Main function

    bool checkInclusion(string s1, string s2) {

        // exception case - we need to check if the str1's any 1 permuation using all char of s1 is present in s2 or not, so just think if s1 is longer then s2, then there can never be a permutation of s1 in s2 so return false

        if(s1.length() > s2.length()) return false;

        // create 2 vector<int> coz we want to map characters with index and set the values as the frequency, make sure to declare both of size 26, else runtime error will occur

        vector<int> v1(26),v2(26);



        // map all characters of s1, with index of v1, and store their frequencies, also map 1st 'str1.length()' characters of s2 in v2

        int index = 0;
```

```
        for( ; index < s1.length(); index++){

            int indexOfV1 = s1[index] - 'a';

            char indexOfV2 = s2[index] - 'a';

            v1[indexOfV1]++;

            v2[indexOfV2]++;

        }



        // check if both the vectors , are equal then return true

        if(areVectorsEqual(v1,v2)) return true;



        // if not equal then run loop from where we left the index, till the end of s2

        while(index < s2.length()){

            // fetch the last inserted character'a index in the string s2, then find the character at this index, then decrement this characters freq  (using formula = index - s1.length()) , if s1 = "abc" s2 = "a[bcd]ef" so using formula lets sat we are on index =  4 (e), and we need to find the ind of 1st character of window in s2, using formulaIndex = 4 - 3 = 1, so lets fetch the character = s2[formulaIndex] => s2[1] i.e b so combining all the story we have [ windowStartCharIndex = s2[index - s1.length()] - 'a' ]

            int windowStartCharIndex = s2[index - s1.length()] - 'a';

            v2[windowStartCharIndex]--;



            //fetch the next character from the s2 and find its index in v2, then increment it by 1

            int nextCharIndex = s2[index] - 'a';

            v2[nextCharIndex]++;

            // increament the index

            index++;

            // if v1 == v2, then return yes
```

```
            if(areVectorsEqual(v1,v2)) return true;

        }


        // loop ends means no permutation of s1 found in s2

        return false;

    }


};
```

dry run and formulas used above are explained :-

building Approach

let say str1 = "hel" str2 = "balhecd"

'v1'

we will store the str1's i.e smaller string all ch with their freq in a vec. (by mapping char to index & values as freq)

then we create another vector 'v2' & store the same no of char as v1 in v2 also & with their freq.

char ki ascii nikalo usse value ko +1 karo

v1 [1 1 1]
'h' 'e' 'e'

[ ]

note: we cant store complete str2 in vector2 at once coz eg. when str1 = "hel" & str2 = "hole"

v1 [1 1 1]
e h l

v2 [1 1 1 1]
e h l o

now when we compare the 1st 3 char of both v1 & v2 the ans = true

but in reality ans is false coz h le are not together in str2

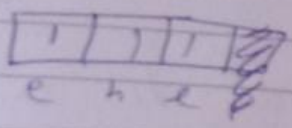
Lets say str1's leng = x

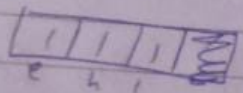
So what we do is we store all x elements of str1 in vector1 & last x elements of str2 in vector2

Then we compare both if equal → True else we remove char. we inserted at first in V2 & insert a new char in V2.

eg. s1 = h e l

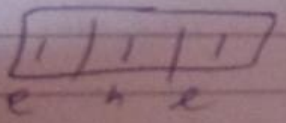
s2 = h o l e

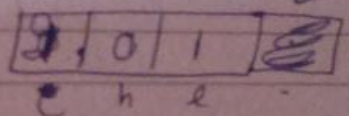




V1 = V2 (no).

Then keep going





V1 = V2 (no).

Keep going

here str2 ends

∴ return false

how does formula 'i - s1.len()' works?

Suppose s1 = "abc"

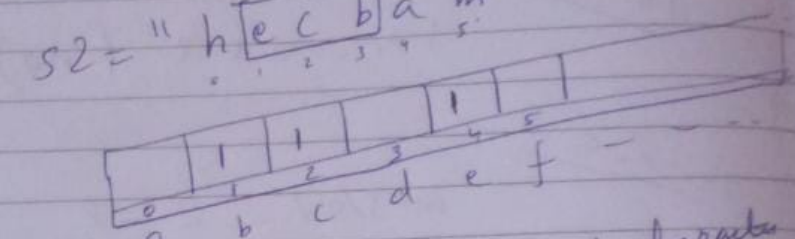
s2 = "hecbam"

so our window will be of size 3 right?

suppose we reached here i = 4

s2 = "h e c b a m"

& our vector v2 =



now we need to remove the start character from sliding window by decrementing its freq by 1.

So to get the character we have to find where is that character inside s2.

s-1 index in s2 = i - s1.len()

= 4 - 3

= 1

∴ lets fetch this character from s2 string

s-2 char := s[index_in_s2] = s[1]

char = e

now we need to find where this 'e' is stored in vector V2.

int. index in V2 = char - 'a' (finding asci vals.)
= e - 'a'
= 4 - 0
= 4.

∴ go to V[index in V2] i.e V[4] & decrement its frequency by 1.

Step 4 ∴ V[index-in-V2]--;

Thats how it works, so combing all 4's steps

we have, int windowStartCharIndex;

```
windowStartCharIndex = s2[i - s1.length()] - 'a';
v2[windowStartCharIndex]--;
```

done.

---------- END ----------