Find All Anagrams in String - QOTD 5th Feb 23

Leetcode Link : Click

Approach

we will use sliding window approach here

Complexity

- Time complexity:$O(n * 26)$ - where n is length of longer string 's'
- Space complexity: $O(52)$ - for creating 2 vectors v2,v2

Approach Steps

```
/* ⭐✔️Approach - 1 (using sliding window)

        explanation :-

                -> // Fun.2 : areVectorsEqual() - returns truee is 2 vectors are
exactly equal in everything

                    step 1 :  to check if both the vectors have same number of element
with same freq, run a loop from 0 to 25 and if at any index if the freq is not same,
return false
                    step 2 : when the loop ends, means all the freq of all the
characters are same, so return true


                -> // Main function

                    step 0 : excption case- when p's length is more then s length,
then in that case , no anagram of p can ever exist in 's'
                    step 1 : create a vector<int> 'ans' to return at the end, and
create 2 vectors v1 (for string 'p') and v2 (for string 's' the longer one) - both
vectors of size 26
                    step 2 : now push all characters frequencied of string p into v1,
and all the first p.length() characters frequencies in v2(make sure to map their
characters asci values with the indexes of the vectors)
                    step 3 :if the v1 == v2 then store index 0 in ans vector
                    step 4 : run loop while index is lesser then the length of s
                        step 5 : we need to remove the start char of the window
                        fetch the index in 's' of charcter to be removed from the
start of the window , then fetch that character and find its asci value and then
decrement that index by 1 in the v2
                        step 6 : fetch the next characters asci which we are inserting
in our sliding window , and increment it by 1
                        step 7 : if v1 == v2, insert the start index i.e
'startcharIndexInS + 1 ' in the ans . this is because now the windows new start index
is startcharIndexInS + 1, since (startcharIndexInS already is removed)
                        step 8 : increment the index
                    step 9 : return the ans vector

        ☑Time : O(n*26) - where n is length of longer string 's'
        ☑Space : O(52) - for creating 2 vectors v2,v2

    */
```

Code

```
class Solution {

private:

    // Fun.2 : returns true only when 2 vectors are exactly same, else return false

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



        // to check if both the vectors have same number of element with same freq, run a loop from 0 to 25 and if at any index if the freq is not same, return false

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

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

        // step 2 : when the loop ends, means all the freq of all the characters are same, so return true

        return true;

    }

public:

    // Main function

    vector<int> findAnagrams(string s, string p) {

        vector<int> ans;

        // step 0 : excption case- when p's length is more then s length, then in that case , no anagram of p can ever exist in 's'

        if(p.length() > s.length()) return ans;

        // step 1 : create a vector<int> 'ans' to return at the end, and create 2 vectors v1 (for string 'p') and v2 (for string 's' the longer one) - both vectors of size 26

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

        // step 2 : now push all characters frequencied of string p into v1, and all the first p.length() characters frequencies in v2(make sure to map their characters asci values with the indexes of the vectors)

        int index = 0; // we need to use this index further

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

```
            int v1Index = p[index] - 'a';

            int v2Index = s[index] - 'a';

            v1[v1Index]++;

            v2[v2Index]++;

        }

        // step 3 :if the v1 == v2 then store index 0 in ans vector

        if(areVectorsEqual(v1,v2)) ans.push_back(0);

        // step 4 : run loop while index is lesser then the length of s

        while(index < s.length()){

            // we need to remove the start char of the window

            // fetch the index in 's' of charcter to be removed from the start of the window , then fetch that character and find its asci value and then decrement that index by 1 in the v2

            int startcharIndexInS = index - p.length();

            char asciOfStartChar = s[startcharIndexInS] - 'a';

            v2[asciOfStartChar]--;

            // fetch the next characters asci which we are inserting in our sliding window , and increment it by 1;

            int nextCharAsci = s[index] - 'a';

            v2[nextCharAsci]++;

            // if v1 == v2, insert the start index i.e 'startcharIndexInS + 1 ' in the ans . this is because now the windows new start index is startcharIndexInS + 1, since (startcharIndexInS already is removed)

            if(areVectorsEqual(v1,v2)) ans.push_back(startcharIndexInS + 1);

            // increment the index

            index++;

        }

        // return the ans vector
```

```
        return ans;
    }
};
```

---END---