**SOLUTION**

class Solution {

public:

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

vector<int> goal(26), cur(26), res;

for(char c : p) goal[c - 'a']++;

for(int i = 0; i < s.size(); i++) {

cur[s[i] - 'a']++;

if(i >= p.size()) cur[s[i - p.size()] - 'a']--;

if(cur == goal) res.push\_back(i - p.size() + 1);

}

return res;

}

};

**TIME COMPLEXITY= O(N)**

**SPACE COMPLEXITY= O(N)**

/\*

s : cbaebabacd

p : abc

goal = {a:1, b:1, c:1}

for i = 0

curr = {a:1}

for i = 1

cur = {a:1, b:1}

for i = 2

cur = {a:1, b:1, c:1}

cur == goal? true, so push\_back (2-3) + 1 i.e. -> 0 = + 1 because we are dealing with 0 based indices for i..

so this is like ((i + 1) - p.size()) where, i + 1 is normal 1-based indexing..

for i = 3

cur = {a:1, b:1, c:1, e:1}

Now, i = p.size(), so we start trimming from the beginning to consider only those elements that are within p.size() from the end.

So cur = {b:1, c:1, e:1}

Here, we do cur[s[i-p.size()] - 'a']++;

The reason is p.size() and cur.size() aren't 0 based, {when we call p.size(), it returns 3 not 2}

Thus, to find start in cur, we need to subtract the size of p from from the current i.

And so on..

\*/