05 March 2025 18:26

1358. Number of Substrings Containing All Three Characters

Solved

Medium | Topics | Companies | Hint

Given a string `s` consisting only of characters *a*, *b* and *c*.

Return the number of substrings containing **at least** one occurrence of all these characters *a*, *b* and *c*.

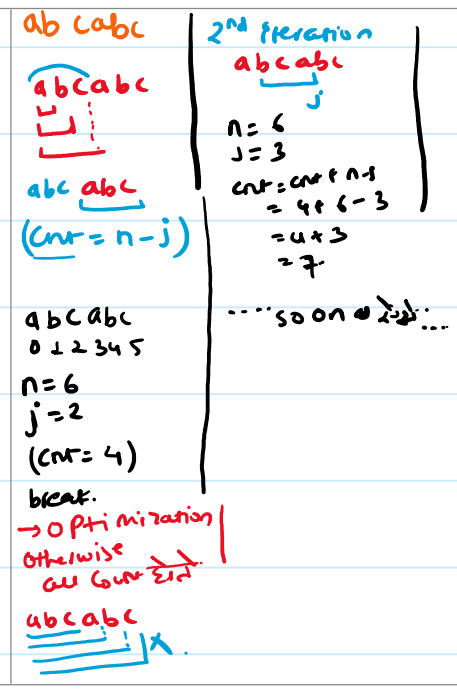
Example 1:

```
Input: s = "abcabc"
Output: 10
Explanation: The substrings containing at least one
occurrence of the characters a, b and c are "abc", "abca",
"abcab", "abcabc", "bca", "bcab", "bcabc", "cab", "cabc" and
"abc" (again).
```

Example 2:

```
Input: s = "aaacb"
Output: 3
```

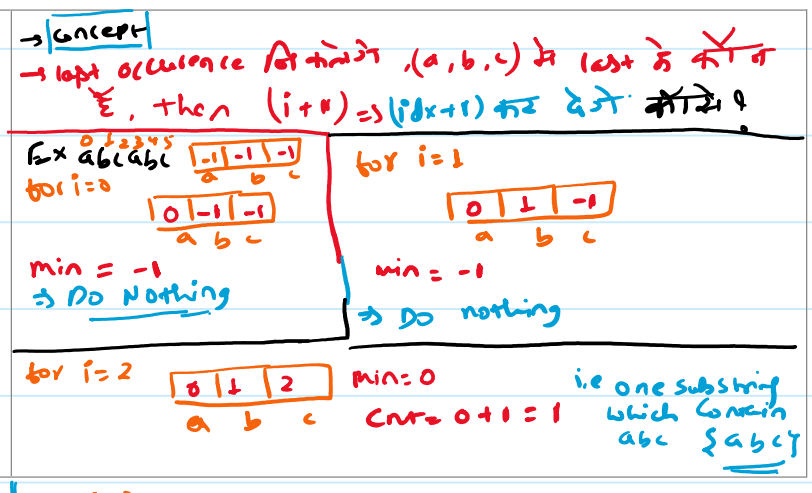
```java
class Solution {
    public int numberOfSubstrings(String s) {

        int cnt=0 ,n=s.length();
        for(int i=0;i<n;i++){
            int cnta=0 , cntb=0, cntc=0 ;
            for(int j=i;j<n;j++){
            char ch=s.charAt(j);
            switch (ch){
                case 'a': cnta++; break;
                case 'b' : cntb++ ; break;
                case 'c' :cntc++; break;
            }
             if(cnta>0 && cntb>0 && cntc>0){
                cnt+=(n-j); break;
            }
            }
        }
        return cnt;
    }
}
```



$TC = O(n^2)$

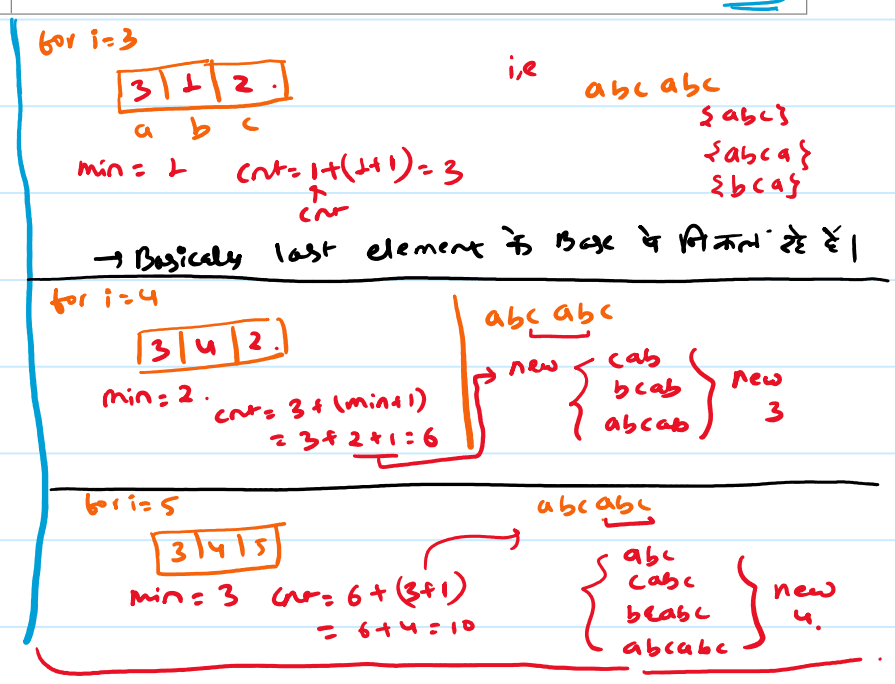
Optimal Approach

```java
class Solution {
    public int numberOfSubstrings(String s) {
        int[] lastseen={-1 ,-1,-1};
        int n=s.length() , cnt=0;
        for(int i=0;i<n;i++){
            lastseen[s.charAt(i)-'a']=i;
            int min=Math.min(lastseen[0], Math.min(lastseen[1] , lastseen[2]));
            if(min!=-1){
                cnt+=min+1;
            }
        }
        return cnt;
    }
}
```



$TC = O(n)$

$SC = O(1)$



11 March 2025 18:21

This is Done By Mc — Tc - O(N)

```java
class Solution {
    public int numberOfSubstrings(String s) {
        int cnta=0 , cntb=0 , cntc=0;
        int cnt=0 , l=0 , r=0 , n=s.length() ;
        while(r<n){
            char ch=s.charAt(r);
            switch (ch){
                case 'a': cnta++; break;
                case 'b' : cntb++ ; break;
                case 'c' :cntc++; break;
            }
            if(cnta>0 && cntb>0 && cntc>0){
                cnt+=(n-r);
            }
             while(cnta!=0 && cntb!=0 && cntc!=0){

              ch=s.charAt(l);
              switch (ch){
                 case 'a': cnta--; break;
                 case 'b' : cntb-- ; break;
                 case 'c' :cntc--; break;
              }
             if(cnta>0 && cntb>0 && cntc>0){
                  cnt+=(n-r);
             }
              l++;
             }
             r++;
        }
        return cnt;
    }
}
```

Tc - O(N)

Concept of Expansion

abcabc

→ Expand until get valid string

i.e abcabc (l … r)

⇒ cnt = n - r = 6 - 2 = 4 = { abc, abca, abcab, abcabc }

→ Shink until get Not valid

i.e abcabc

↳ do nothing

Suppose while shinking is valid

Ex a(0) b(1) b(2) a(3) c(4) a(5) b(6) c(7)

cnt = 8 - 4 = 4 = { abbac, abbaca, abbacab, abbacabc }

→ shinking

abbacabc

window is valid

"bbac" is also valid

cnt = 8 - 4 = (4) ↳ { bbac, bbaca, bbacab, bbacabc }

already is over

abb → left side of this