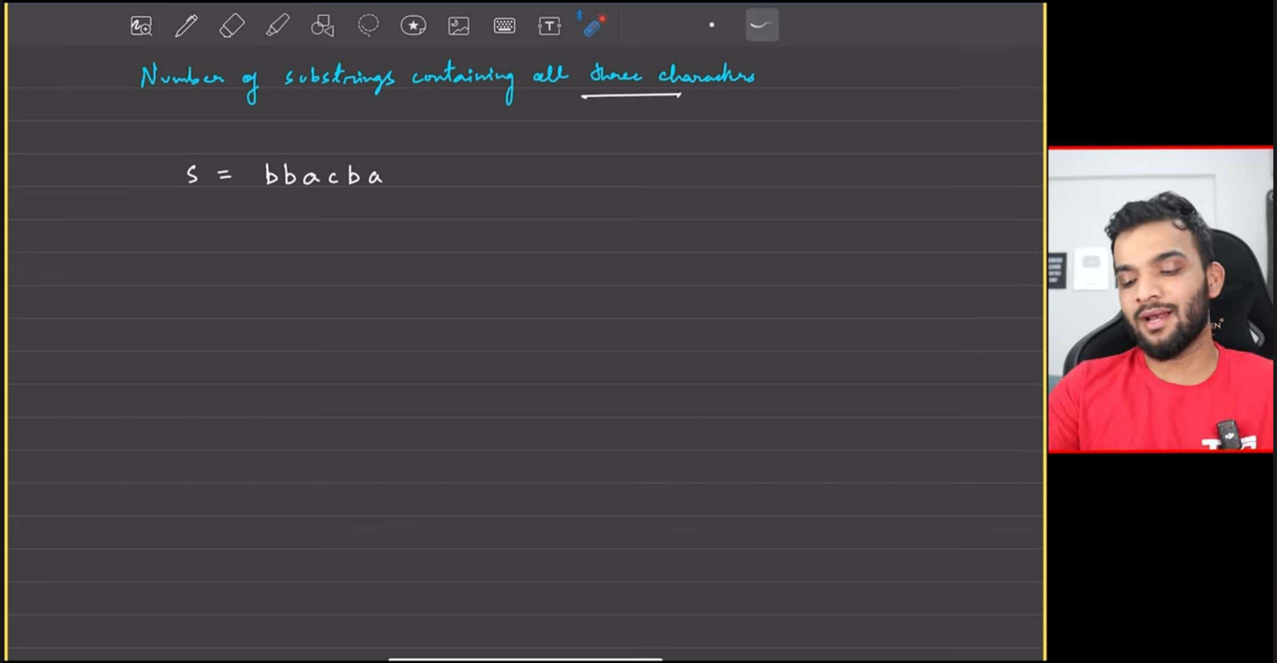
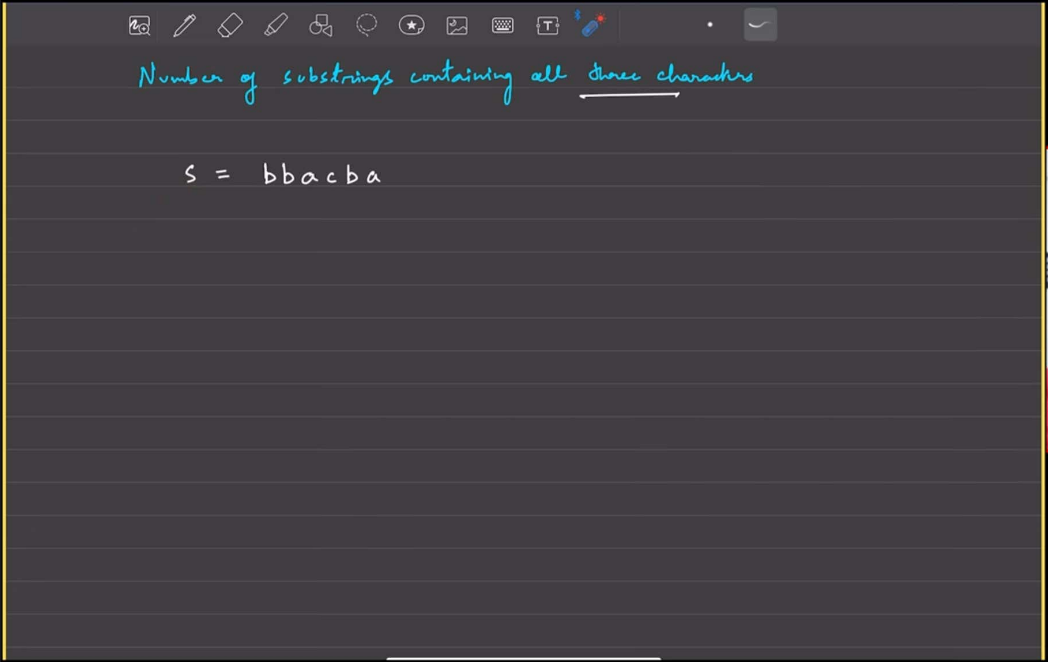


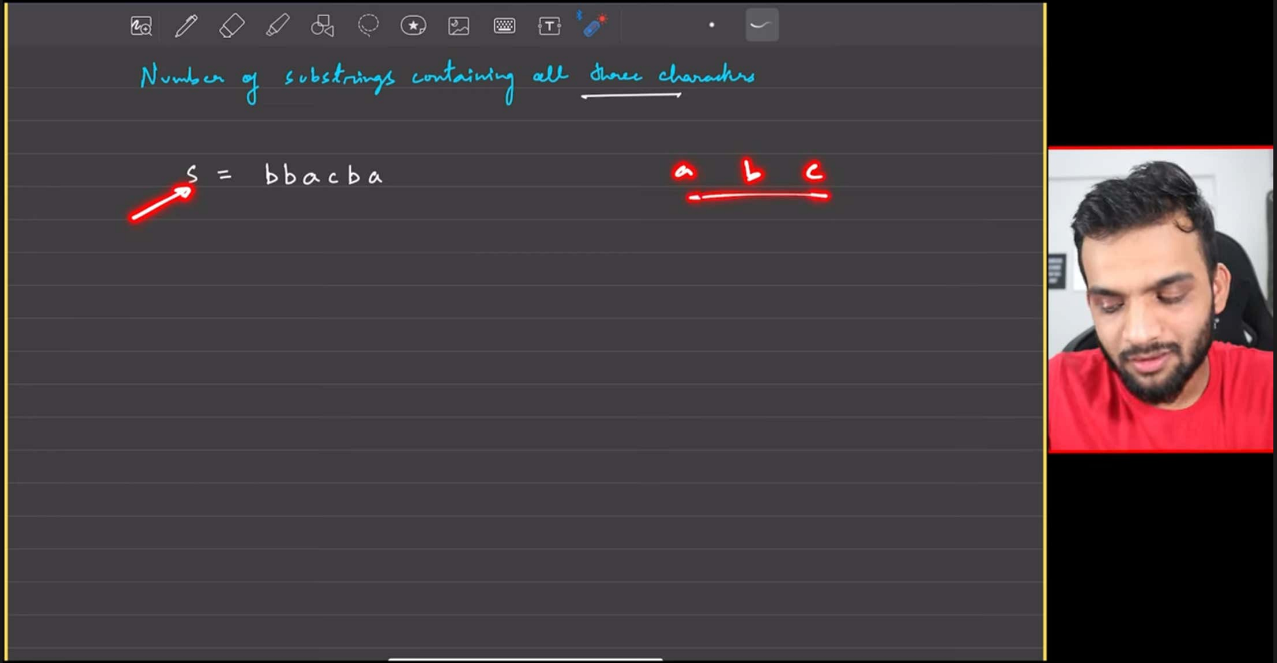
Number of substrings containing all three characters

s = bbacba



Number of substrings containing all three characters

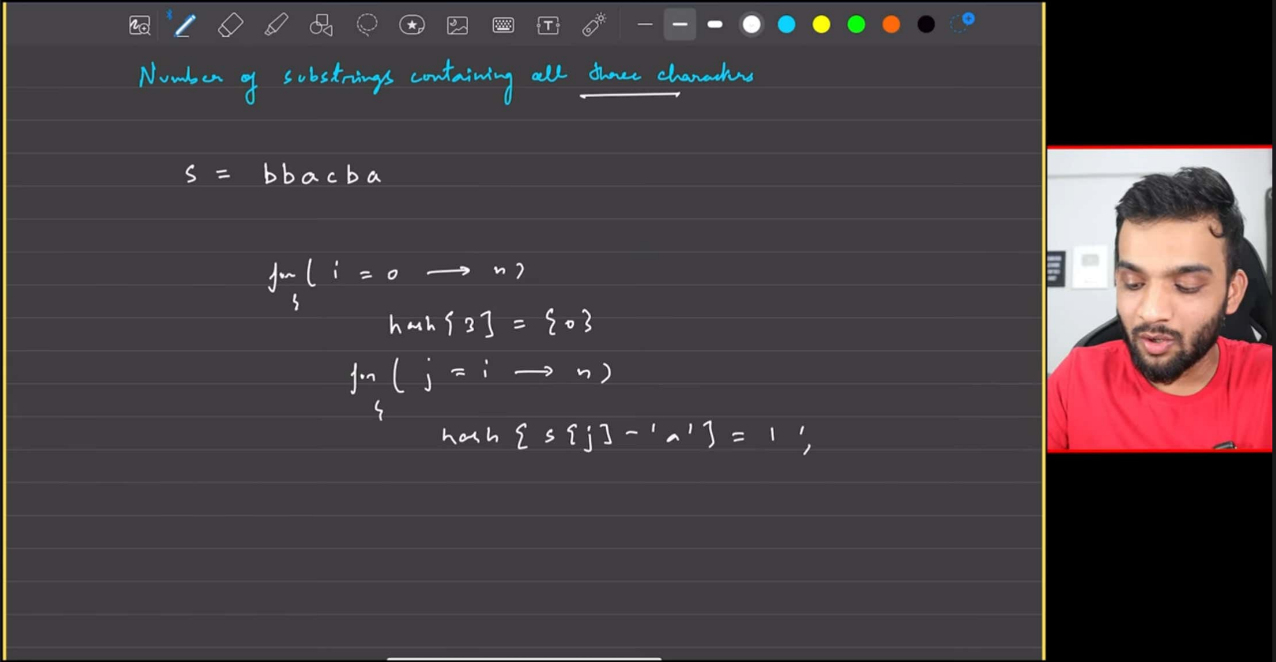
s = bbacba



Number of substrings containing all three characters

s = bbacba

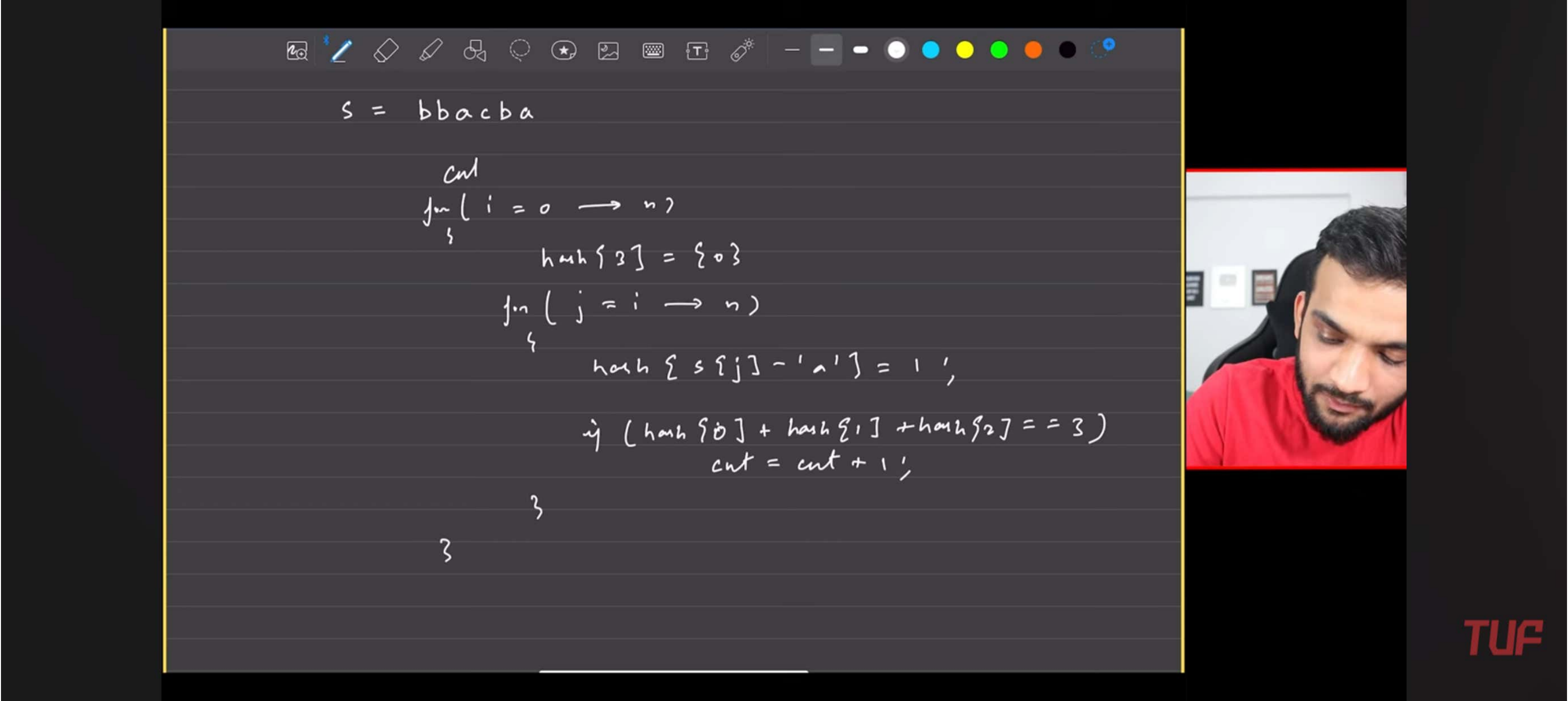
a b c



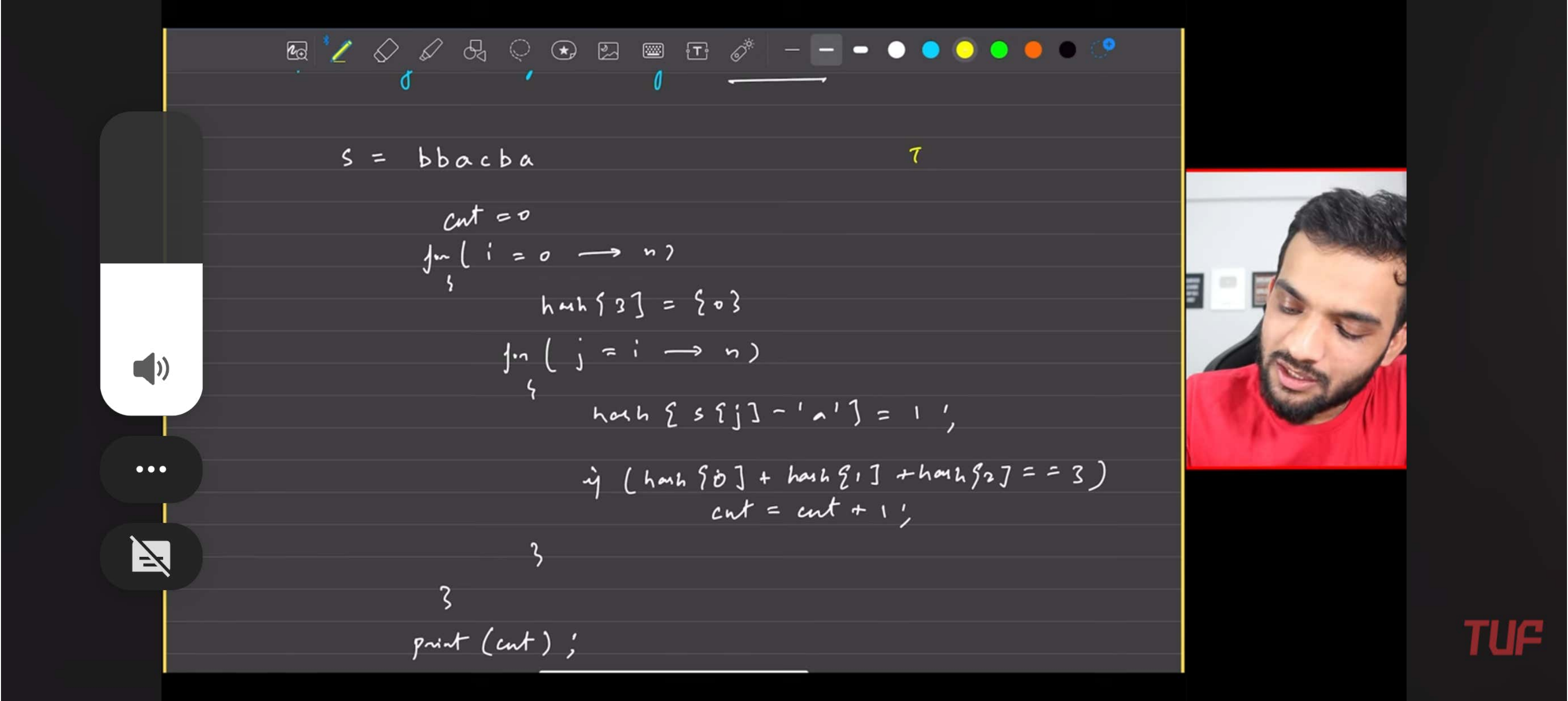
Number of substrings containing all three characters

s = bbacba

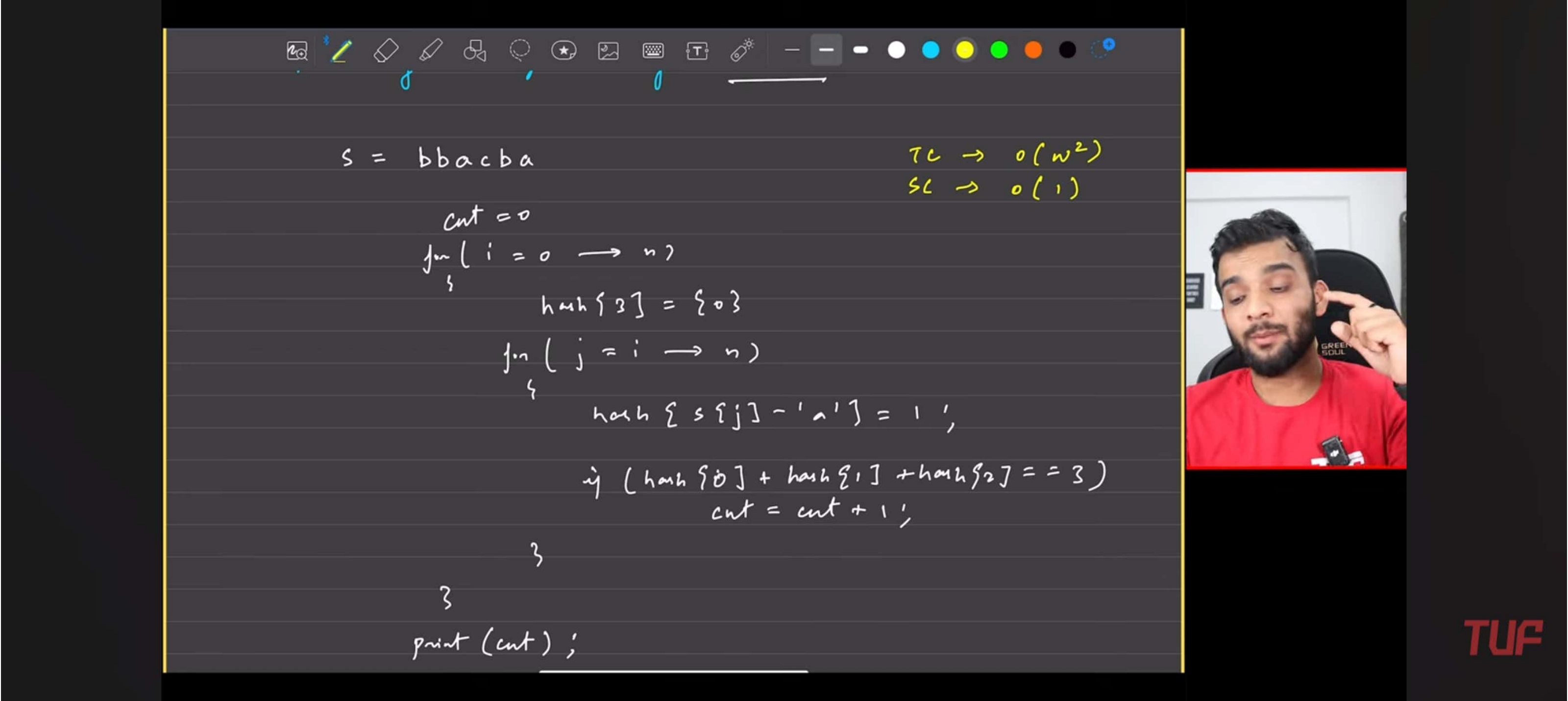
```
for ( i = 0 → n )
{
        hash[3] = {0}
    for ( j = i → n )
    {
        hash[ s[j] - 'a' ] = 1 ;
```



```
s = bbacba

cnt
for ( i = 0 → n )
{
        hash [3] = {0}
    for ( j = i → n )
    {
            hash [ s [j] - 'a' ] = 1 ;

            if ( hash [0] + hash [1] + hash [2] = = 3 )
                cnt = cnt + 1 ;

    }

}
```



```
s = bbacba                                   7

    cnt = 0
   for ( i = 0 ⟶ n)
    {
        hash[3] = {0}
        for ( j = i ⟶ n)
        {
            hash[s[j] - 'a'] = 1;

            if (hash[0] + hash[1] + hash[2] == 3)
                  cnt = cnt + 1;

        }

    }
   print (cnt);
```

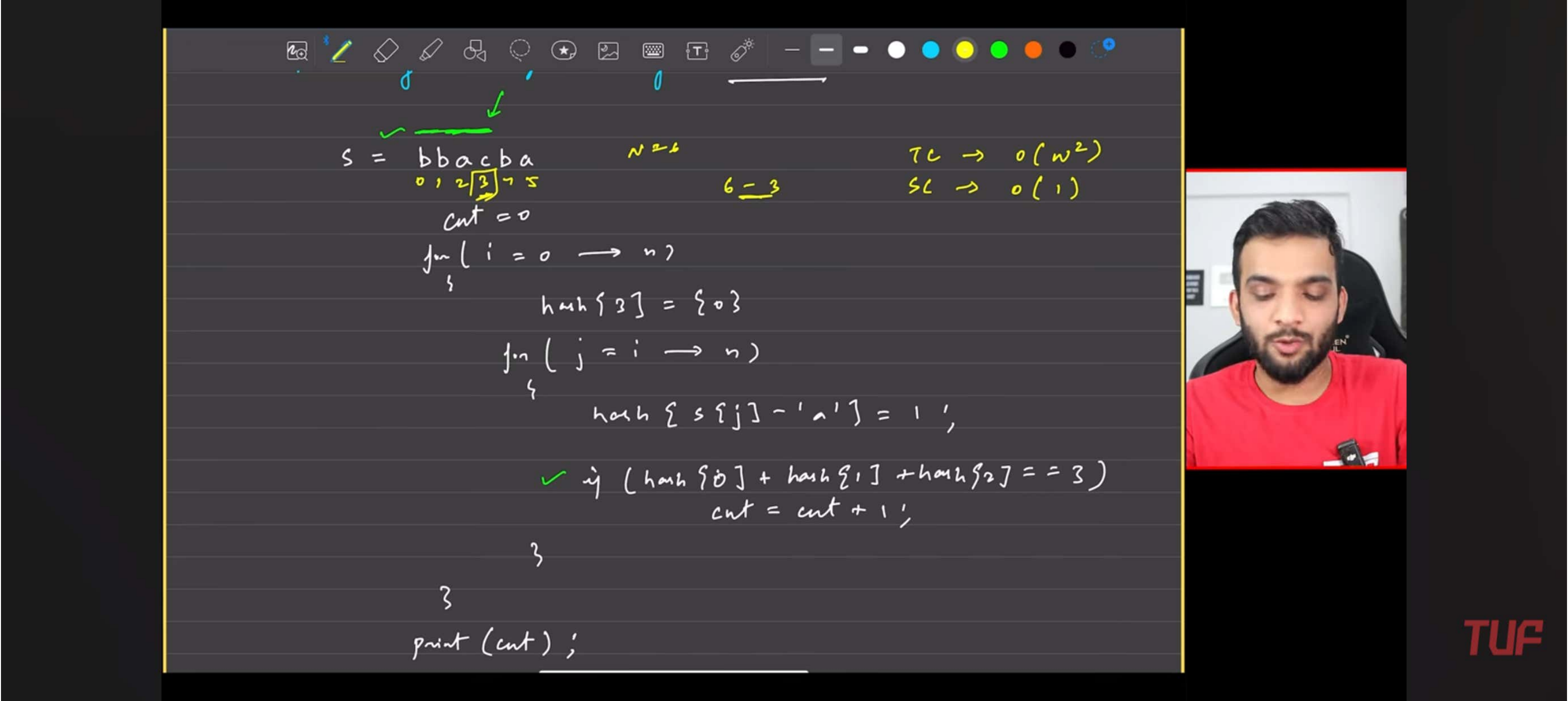


s = bbacba

TC → $o(n^2)$

SC → $o(1)$

```
cnt = 0
for ( i = 0 → n )
{
        hash[3] = {0}
     for ( j = i → n )
     {
          hash[ s[j] - 'a' ] = 1;

          if ( hash[0] + hash[1] + hash[2] == 3 )
                  cnt = cnt + 1;

     }
}
print (cnt);
```



s = bbacba

0 1 2 $\boxed{3}$ 4 5

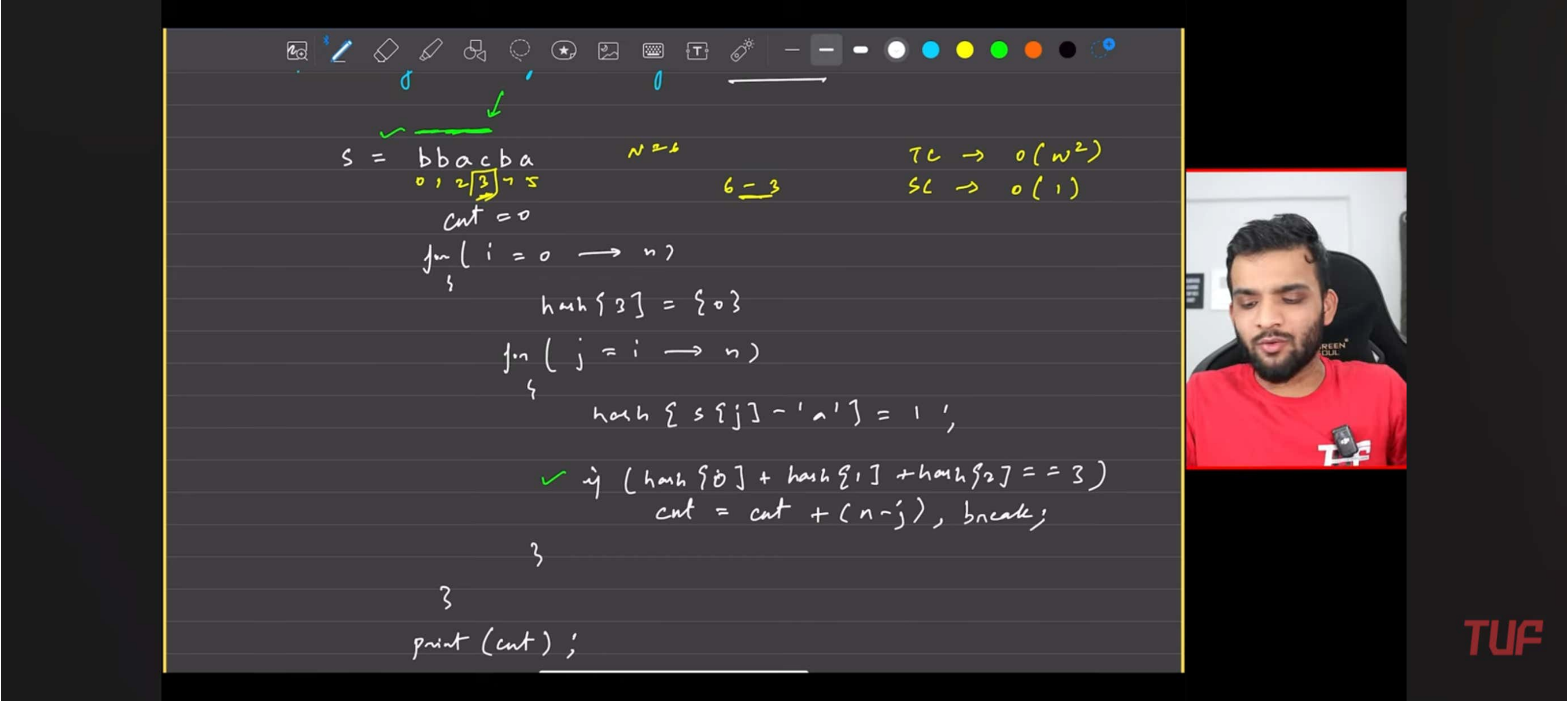
N=6

6 - 3

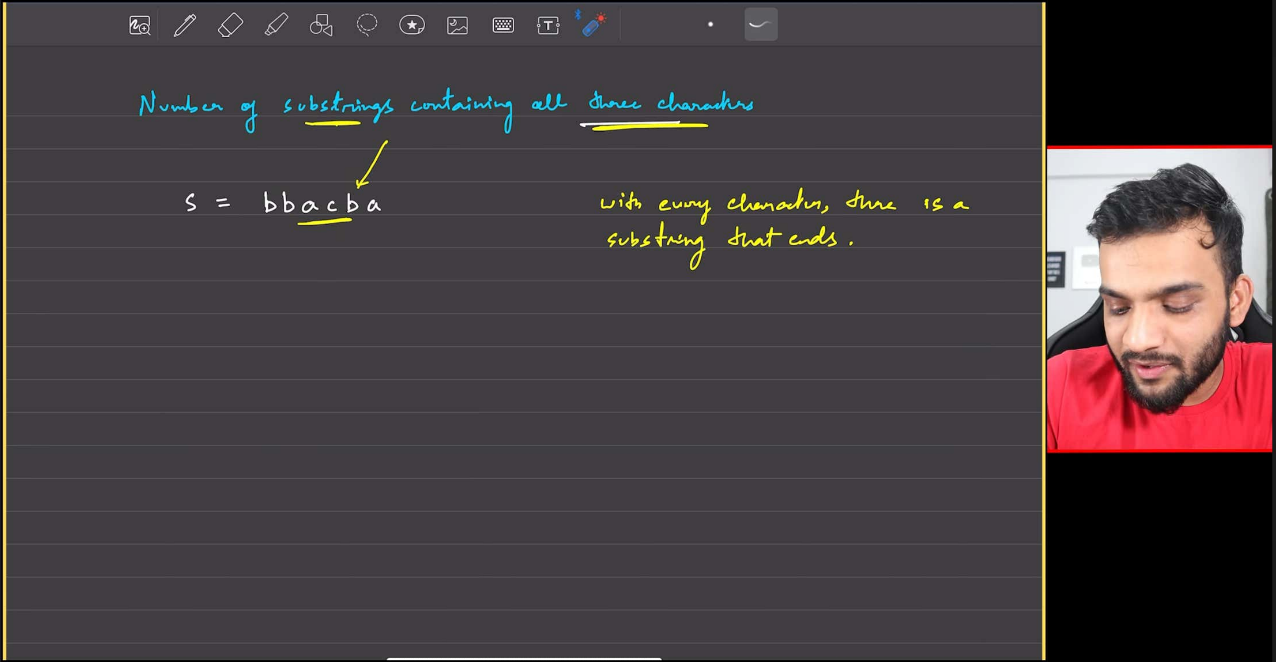
TC → $O(N^2)$

SC → $O(1)$

```
cnt = 0
for (i = 0 → n)
{
    hash[3] = {0}
    for (j = i → n)
    {
        hash[s[j] - 'a'] = 1;
        if (hash[0] + hash[1] + hash[2] == 3)
            cnt = cnt + 1;
    }
}
print(cnt);
```



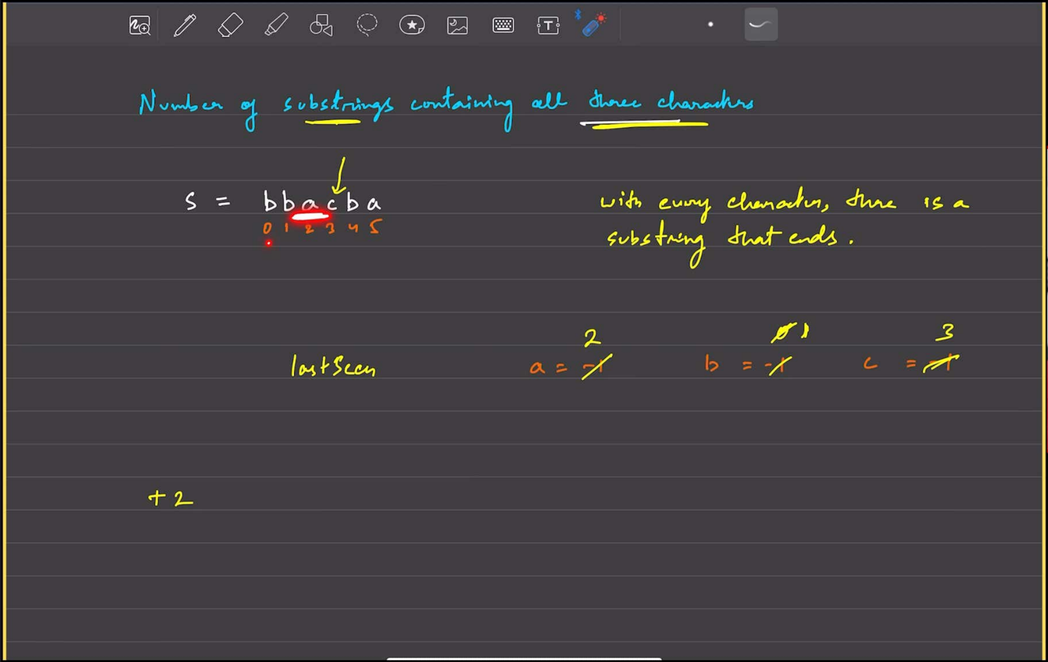
```
s = bbacba
    0 1 2 [3] 4 5            N=6          6 - 3          TC -> O(N^2)
                                                         SC -> O(1)
cnt = 0
for ( i = 0 -> n )
{
        hash[3] = {0}

    for ( j = i -> n )
    {
            hash[ s[j] - 'a' ] = 1 ;

        if ( hash[0] + hash[1] + hash[2] == 3 )
            cnt = cnt + (n-j), break;

    }

}

print (cnt) ;
```



Number of substrings containing all three characters

s = bbacba

with every character, there is a substring that ends.



Number of substrings containing all three characters

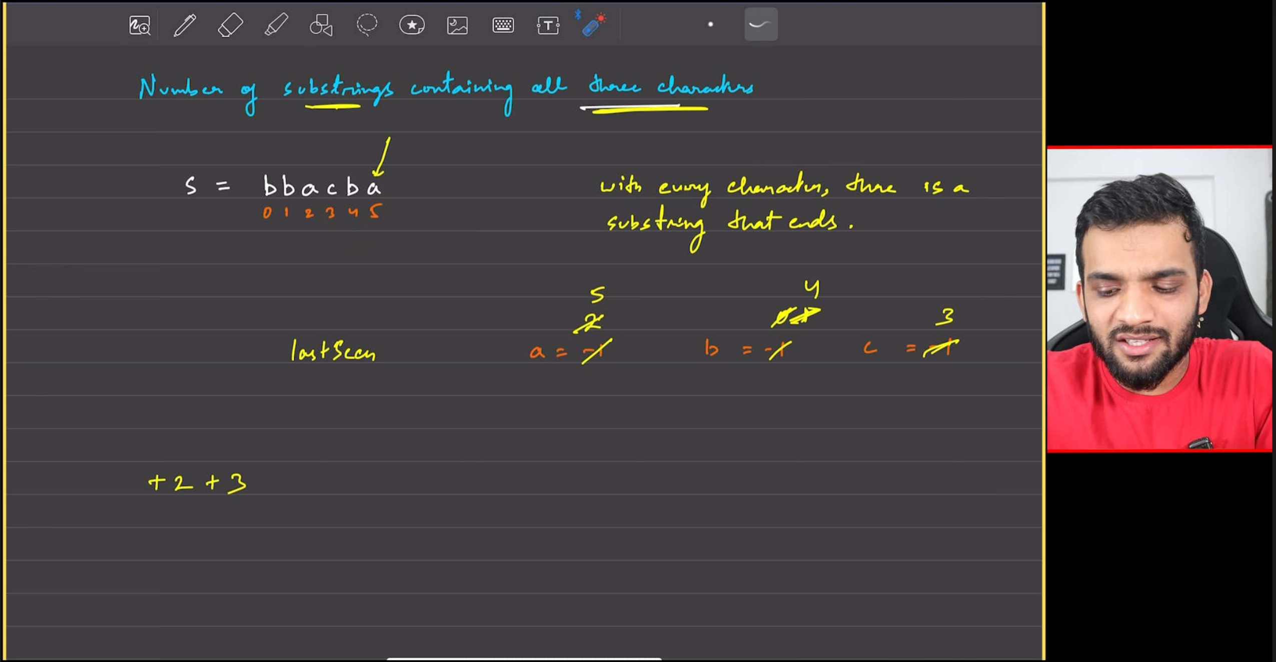
S = bbacba

0 1 2 3 4 5

with every character, there is a substring that ends.

lastSeen a = ~~-1~~ 2 b = ~~-1~~ ~~0~~ 1 c = ~~-1~~ 3

+2



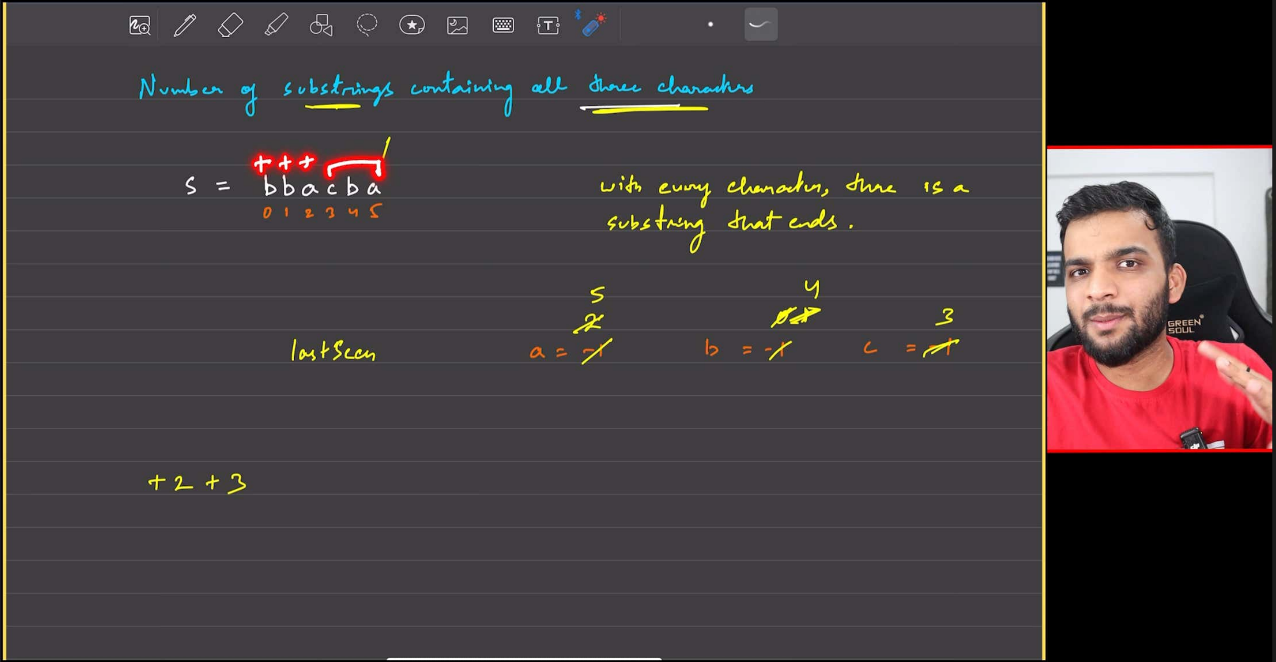
Number of substrings containing all three characters

s = bbacba
0 1 2 3 4 5

with every character, there is a substring that ends.

lastSeen a = ~~-1~~ ~~2~~ 5 b = ~~-1~~ ~~0 1~~ 4 c = ~~-1~~ 3

+2 +3



Number of substrings containing all three characters

s = bbacba

0 1 2 3 4 5

with every character, there is a substring that ends.

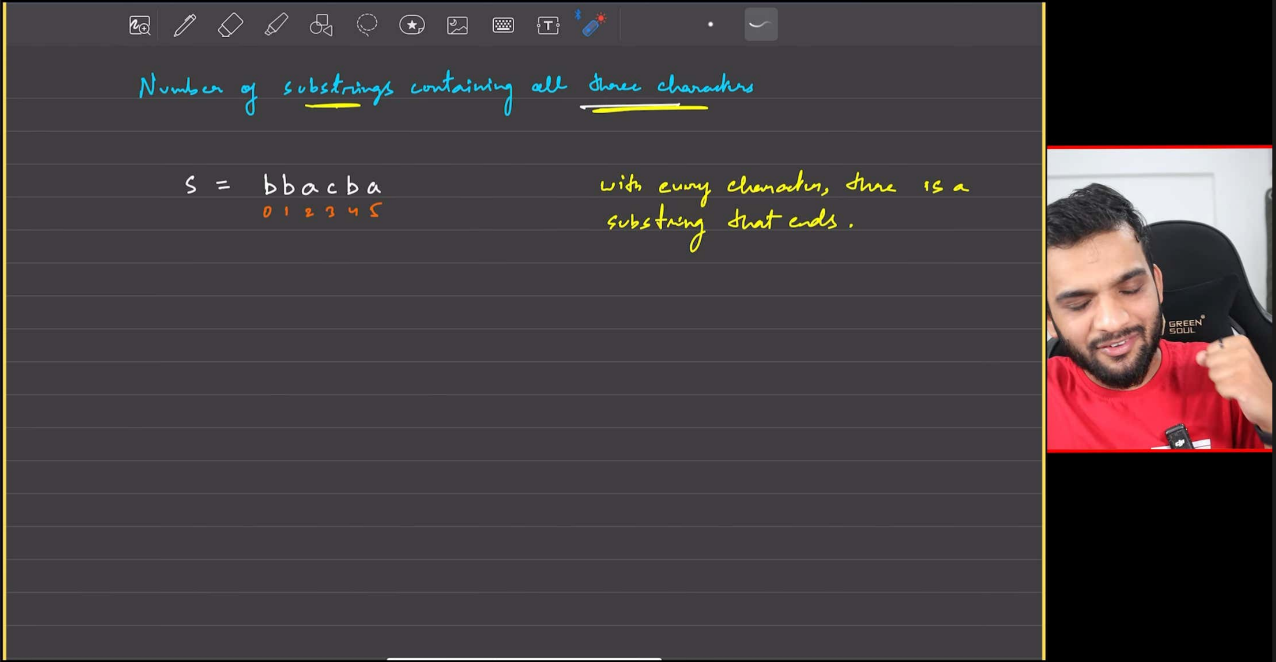
lastSeen

a = ~~-1~~ ~~2~~ 5

b = ~~-1~~ ~~0~~ ~~1~~ 4

c = ~~-1~~ 3

+2 +3

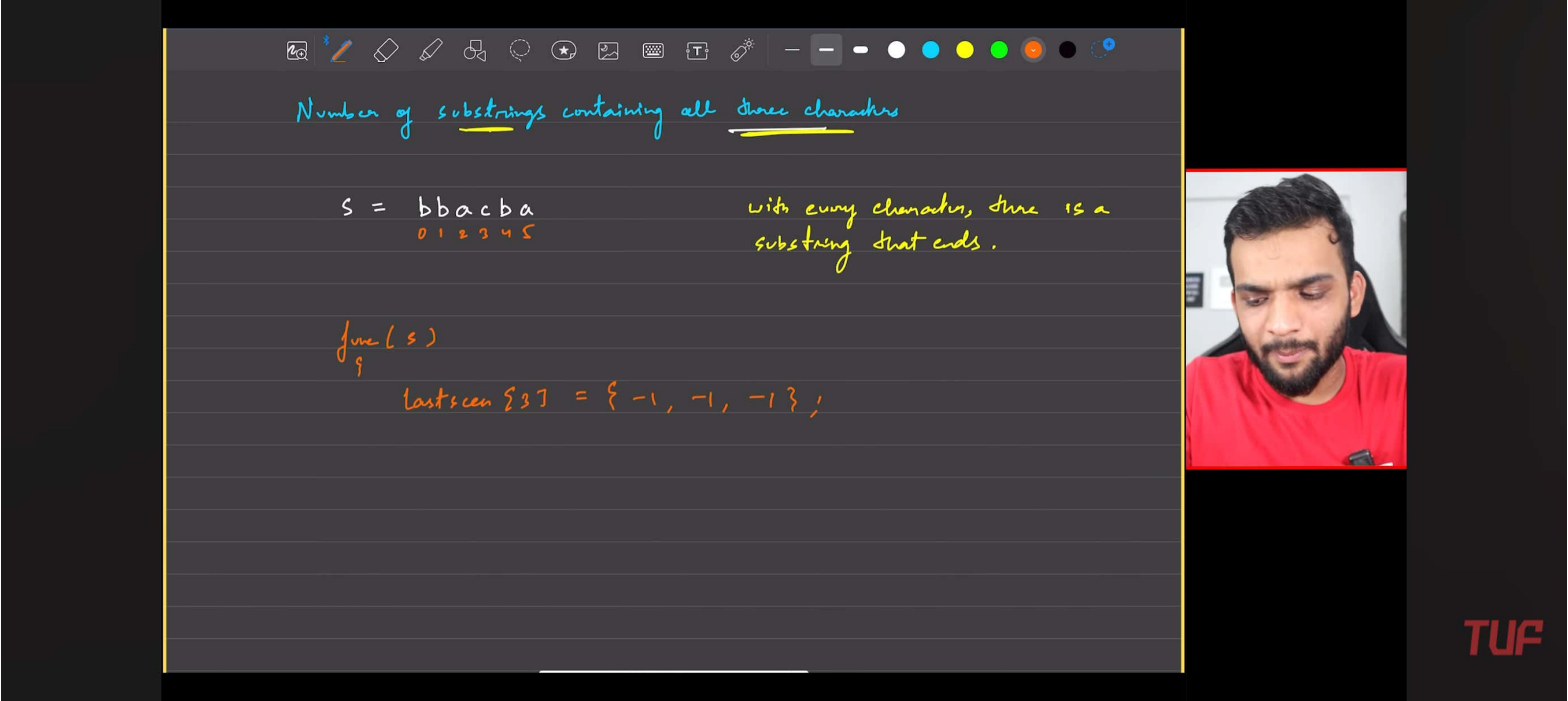


Number of substrings containing all three characters

$s = bbacba$

$0\ 1\ 2\ 3\ 4\ 5$

with every character, there is a substring that ends.



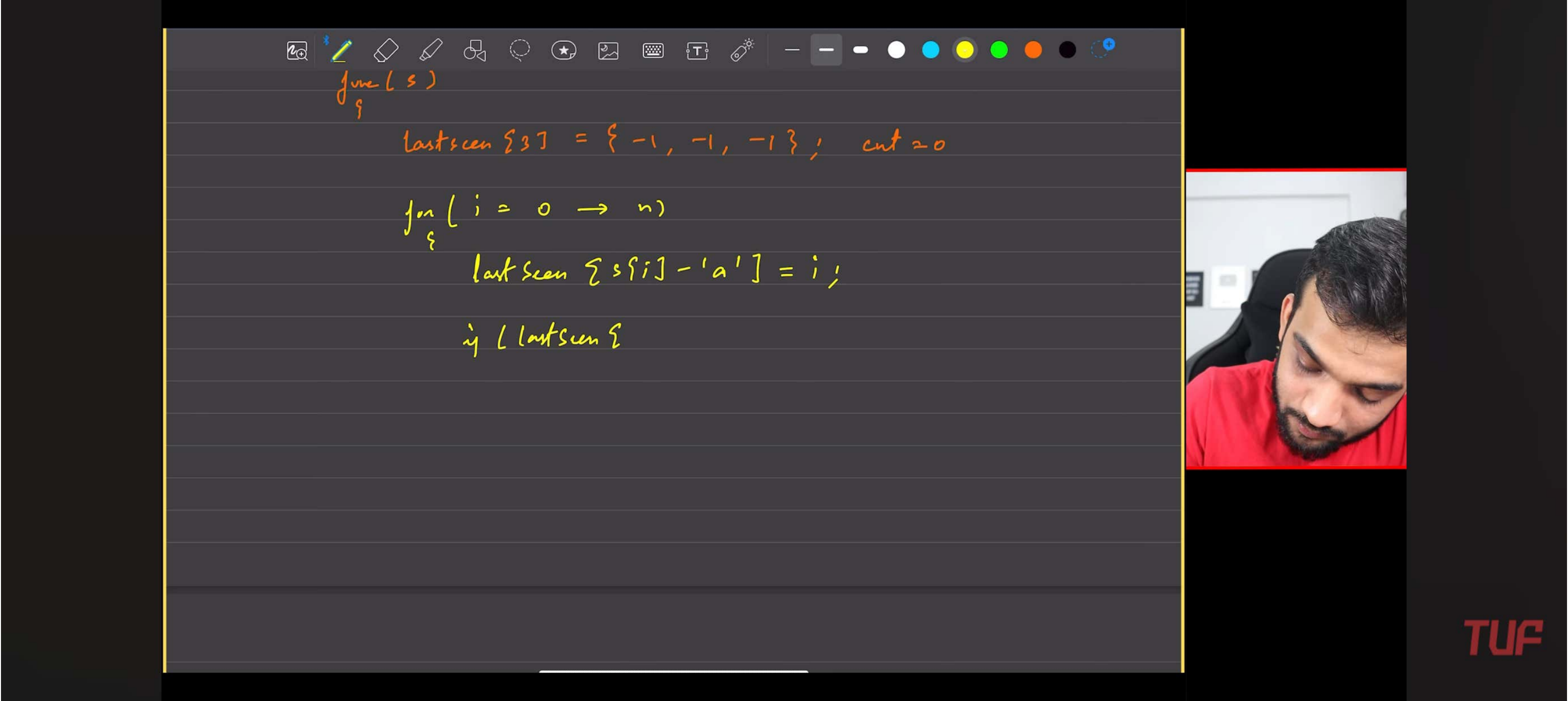
Number of substrings containing all three characters

S = bbacba

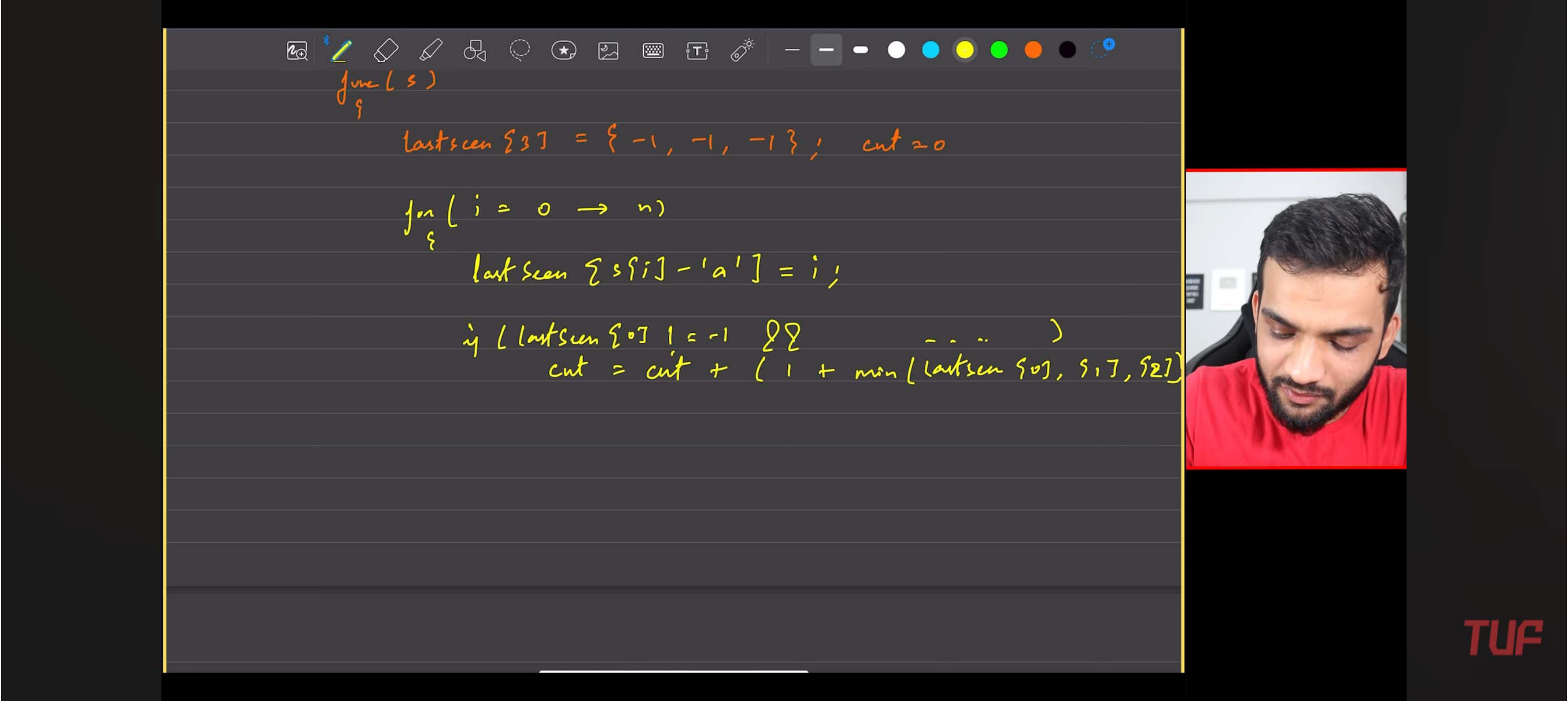
0 1 2 3 4 5

with every character, there is a substring that ends.

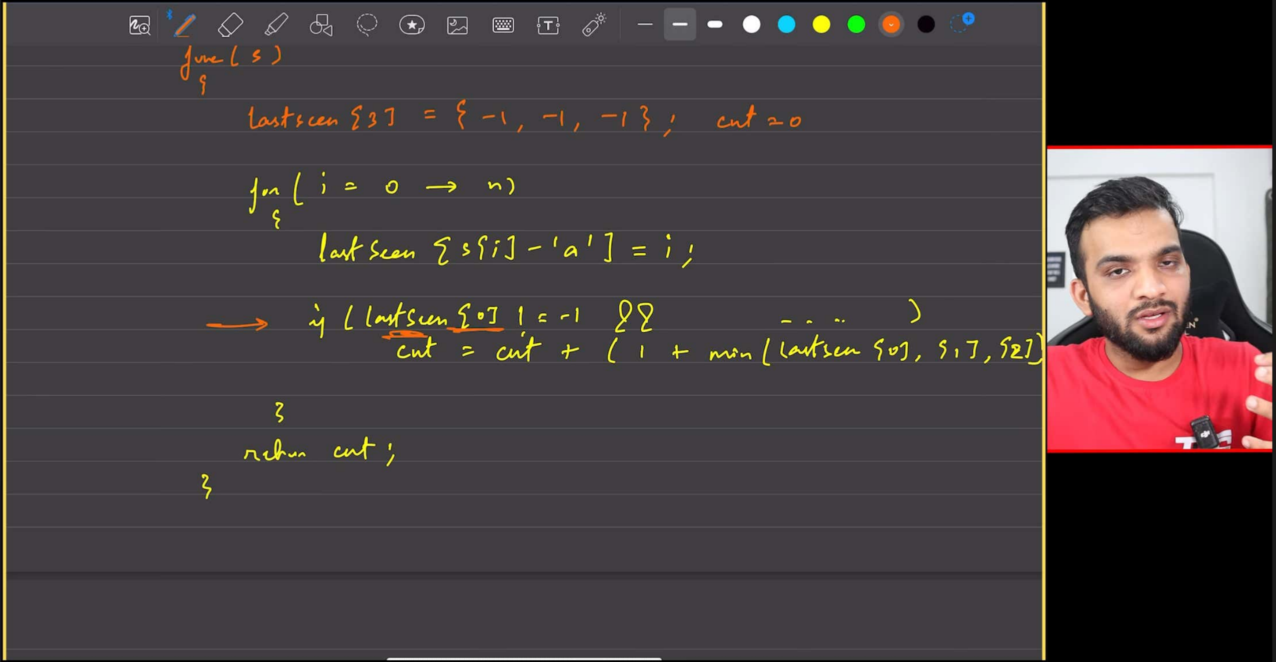
```
func(s)
{
    lastseen[3] = {-1, -1, -1};
```



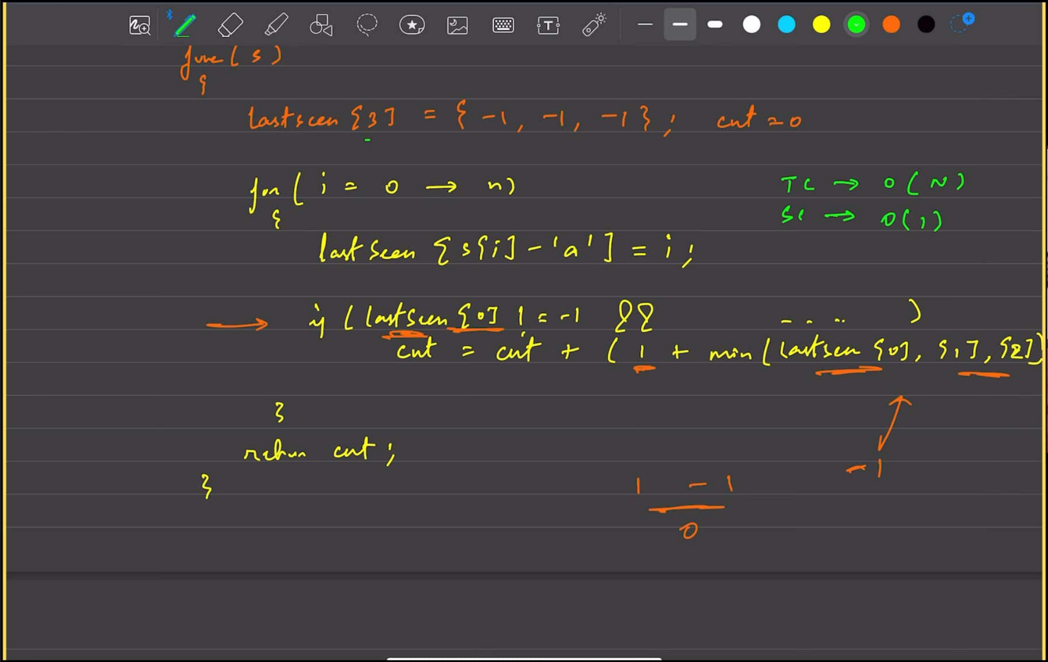
```
func ( s )
{
        lastseen [3] = { -1, -1, -1 };   cnt = 0

        for ( i = 0 → n)
        {
            lastSeen [ s[i] - 'a' ] = i;

            if ( lastSeen [
```



```
func ( s )
{
    lastseen [3] = { -1, -1, -1 };    cnt = 0

    for ( i = 0 → n )
    {
        lastSeen [ s[i] - 'a' ] = i;

        if ( lastSeen [0] != -1  &&   . . . . )
            cnt = cnt + ( 1 + min ( lastsen [0], [1], [2] )
```



```
func(s)
{
    lastseen[3] = {-1, -1, -1};   cnt = 0

    for(i = 0 → n)
    {
        lastSeen[s[i]-'a'] = i;

        if(lastsen[0] != -1  &&        . . . .     )
            cnt = cnt + (1 + min(lastsen[0], [1], [2])

    }
    return cnt;
}
```



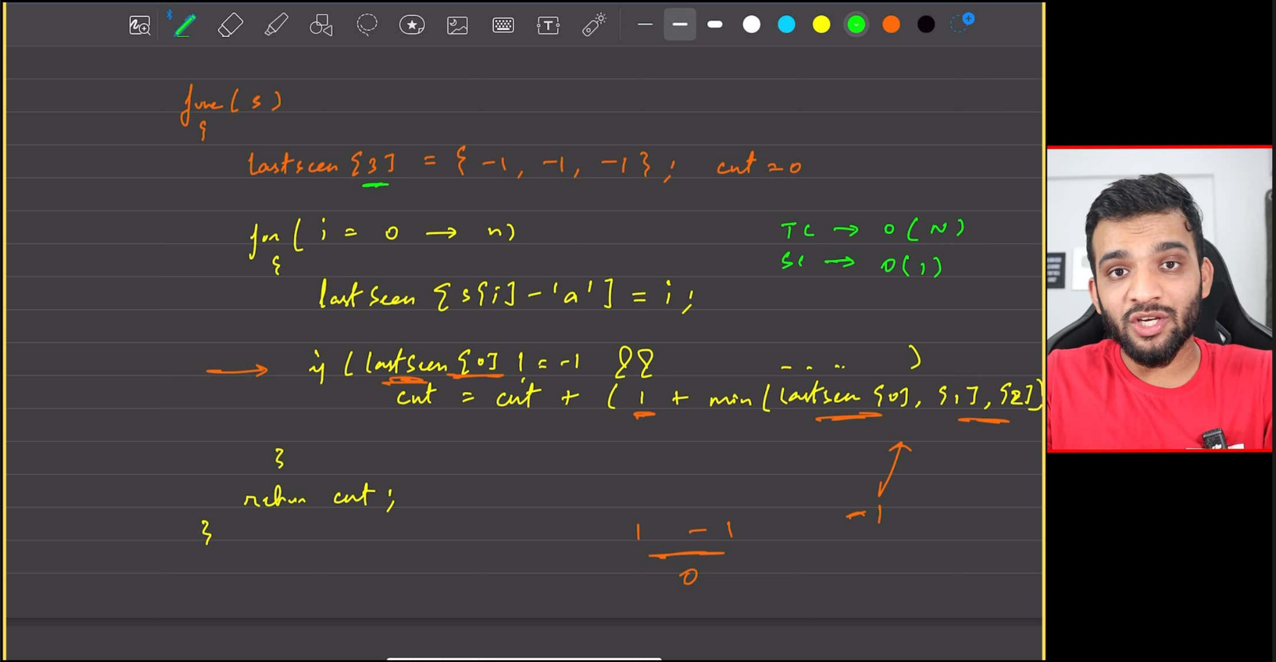
```
func(s)
{
    lastseen[3] = {-1, -1, -1};   cnt = 0

    for(i = 0 → n)
    {
        lastSeen[s[i]-'a'] = i;

        if(lastSeen[0] != -1  &&  ......  )
            cnt = cnt + (1 + min(lastseen[0], [1], [2])
    }
    return cnt;
}
```

TC → O(N)
SC → O(1)

1 -1
0

-1

TUF



```
func(s)
{
    lastseen[3] = {-1, -1, -1};   cnt = 0

    for(i = 0 -> n)
    {
        last Seen[s[i] - 'a'] = i;

        if(lastseen[0] != -1  &&   . . ..   )
            cnt = cnt + (1 + min(lastseen[0], [1], [2])

    }
    return cnt;
}
```

TC -> O(N)
SC -> O(1)

-1

1 - 1
0