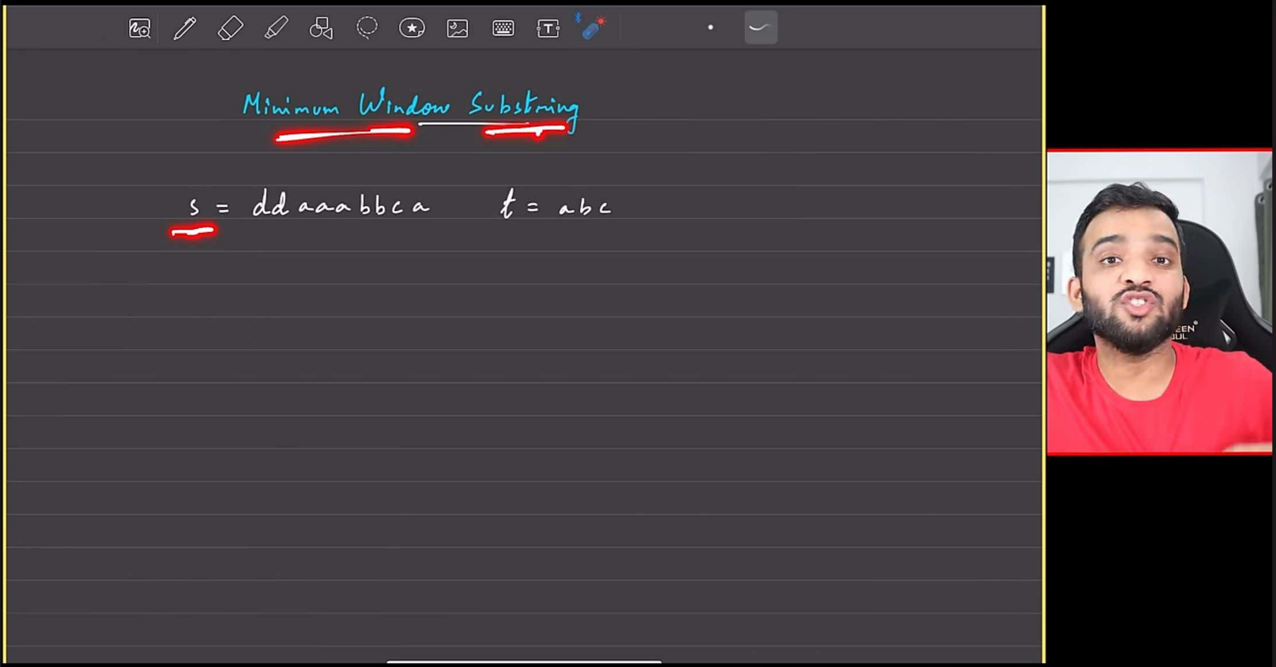
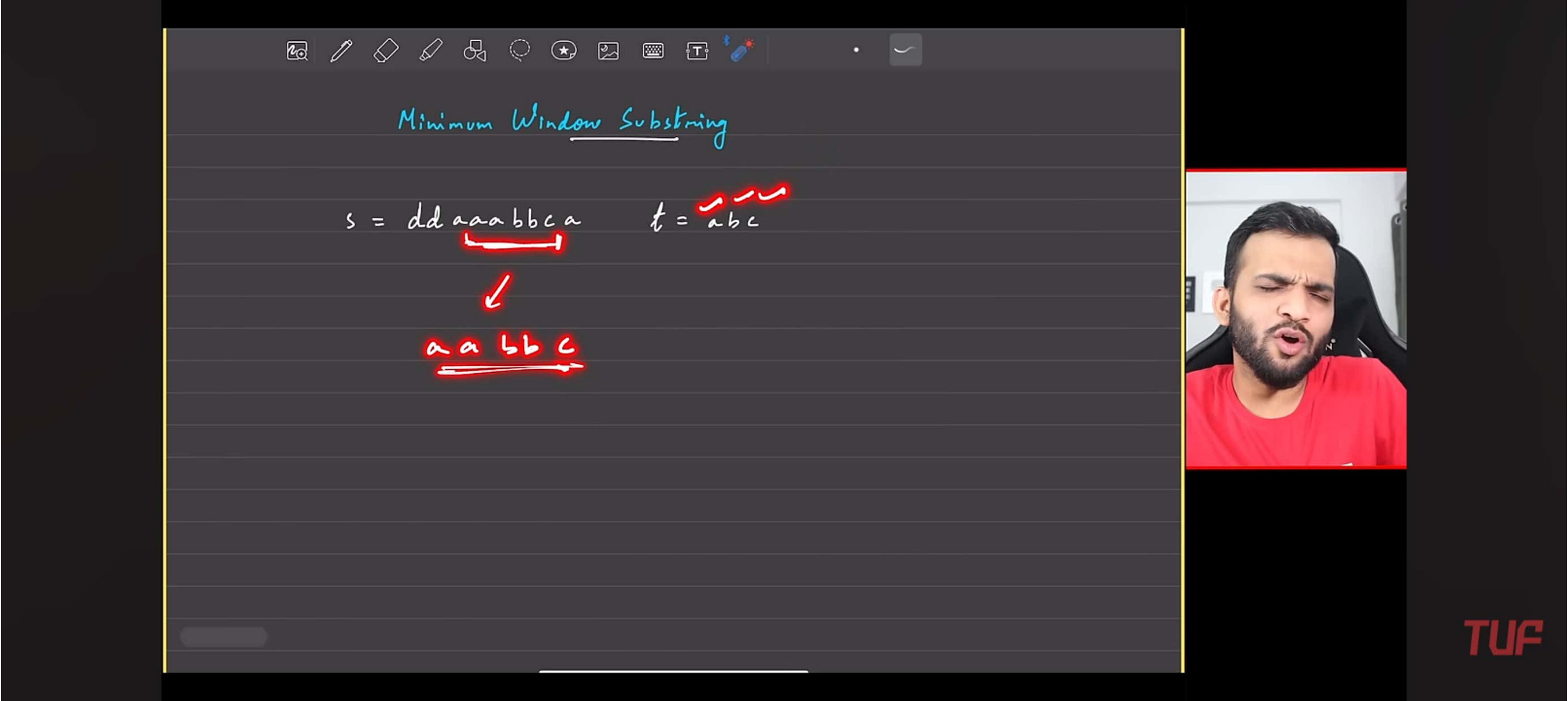


Minimum Window Substring

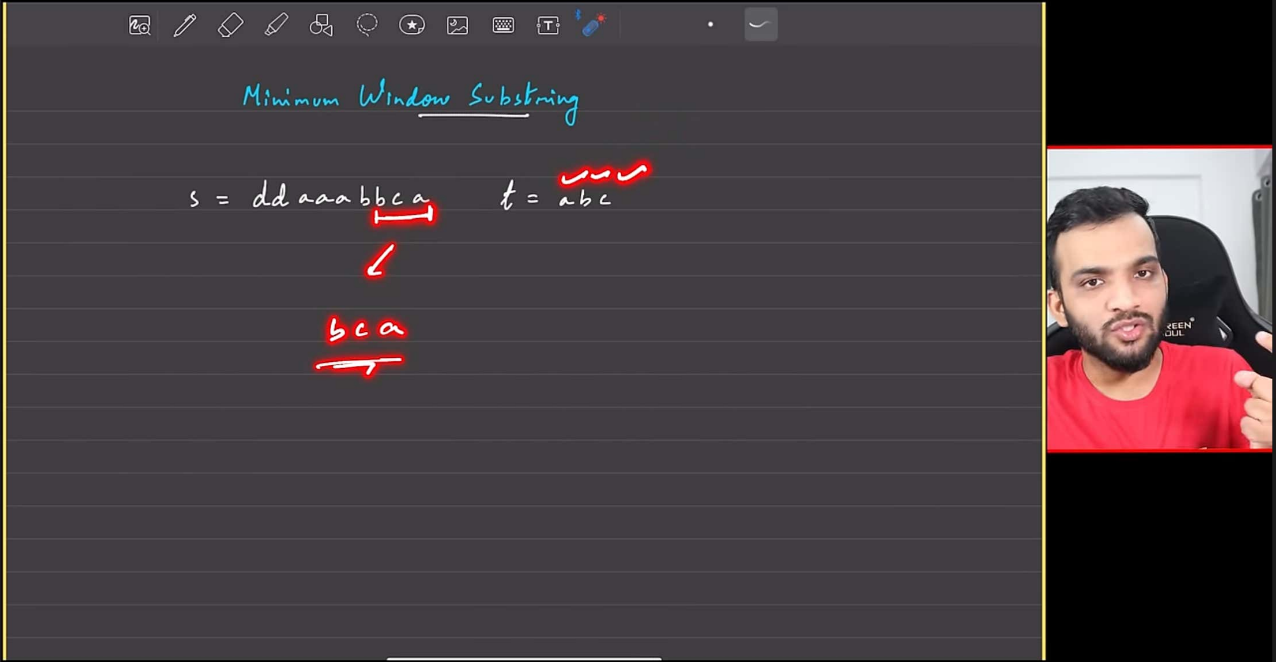
s = ddaaabbca t = abc



Minimum Window Substring

s = ddaaabbca   t = abc

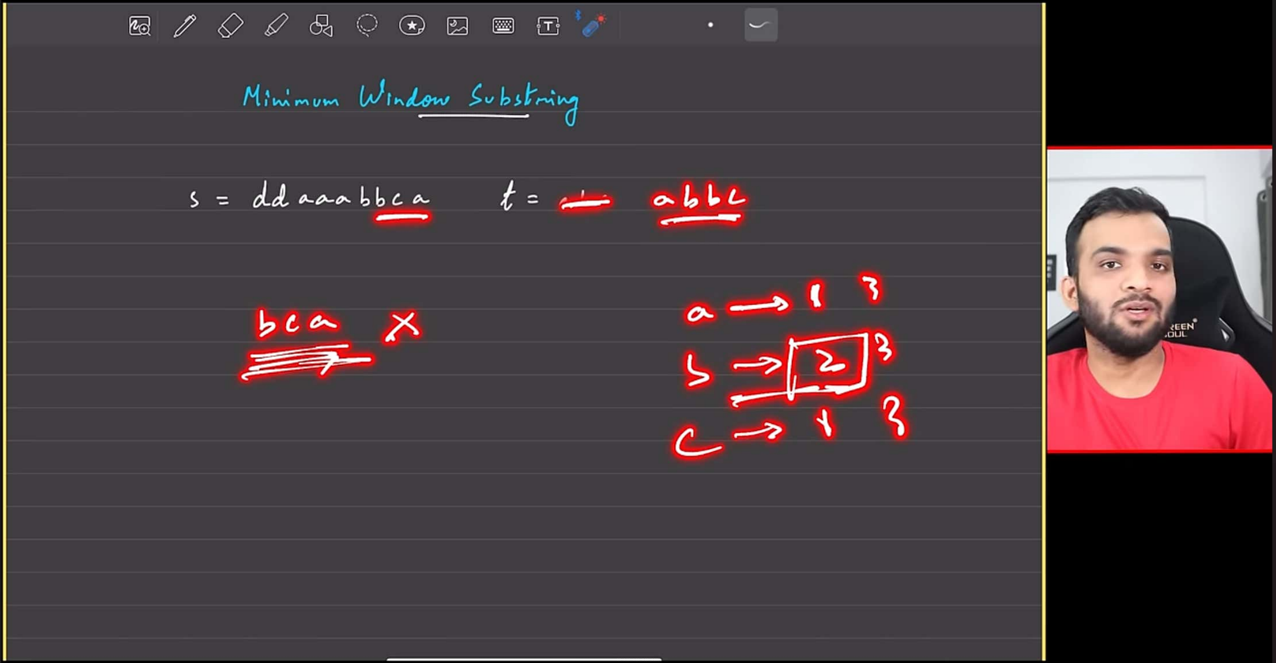
aabbc



Minimum Window Substring

s = ddaaabbca t = abc

bca



Minimum Window Substring

s = ddaaabbca   t = ~~a~~ abbc

bca ✗

a → 1 3

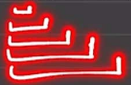
b → 2 3

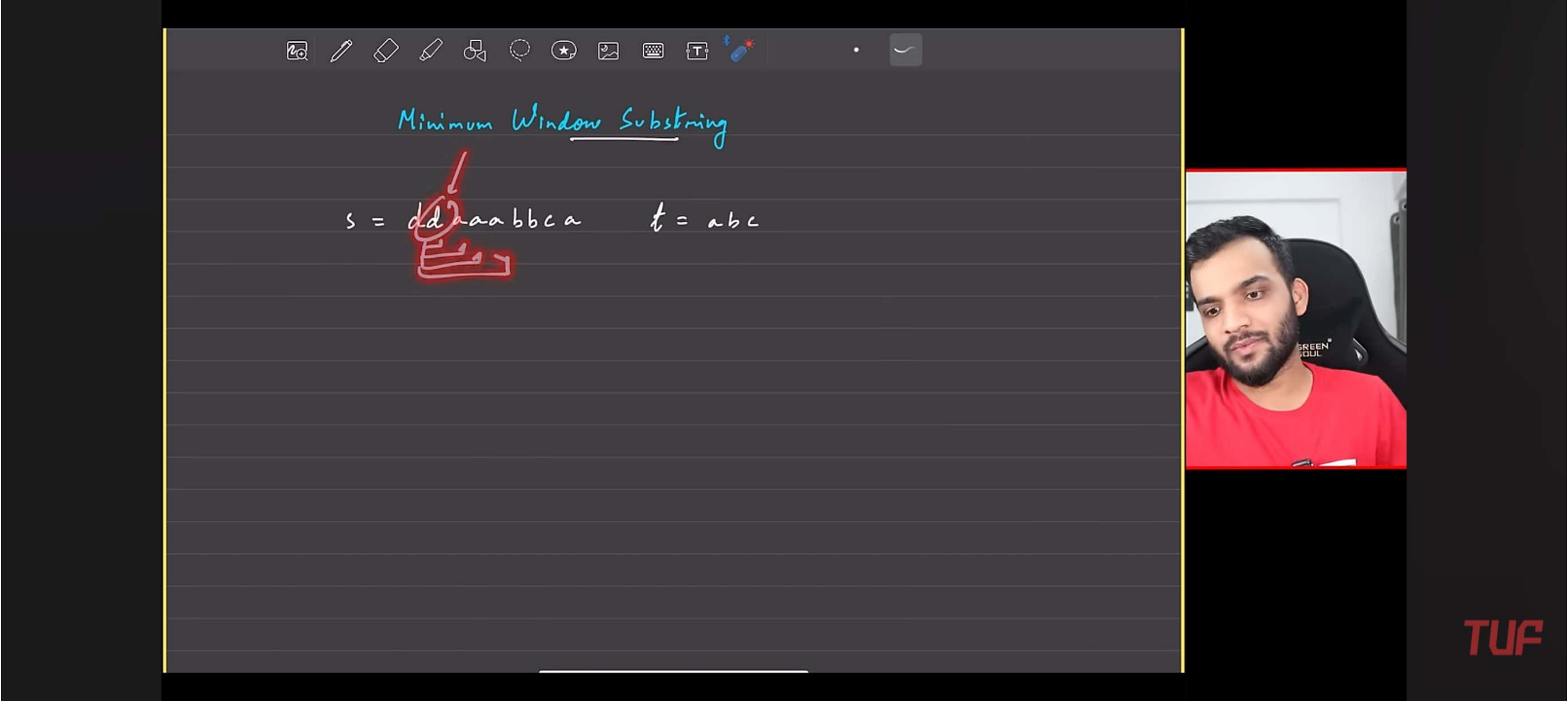
c → 1 3

Minimum Window Substring

↓

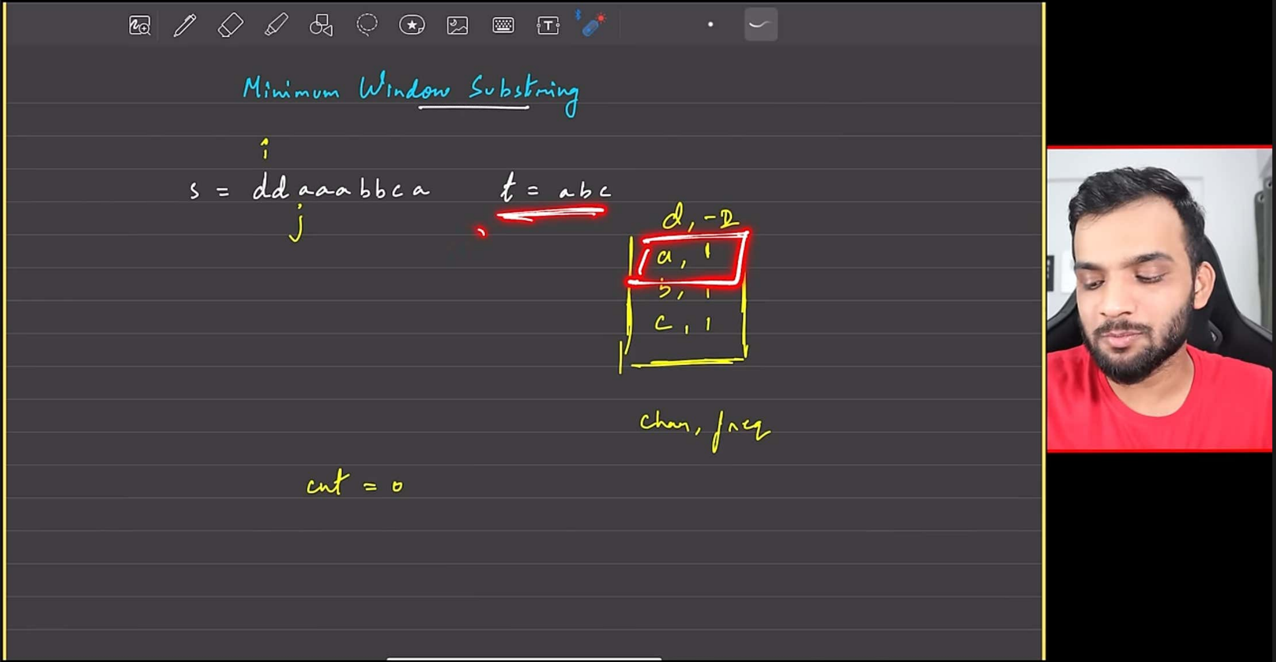
s = ddaaabbca   t = abc





Minimum Window Substring

s = ddaaabbca   t = abc



Minimum Window Substring

i

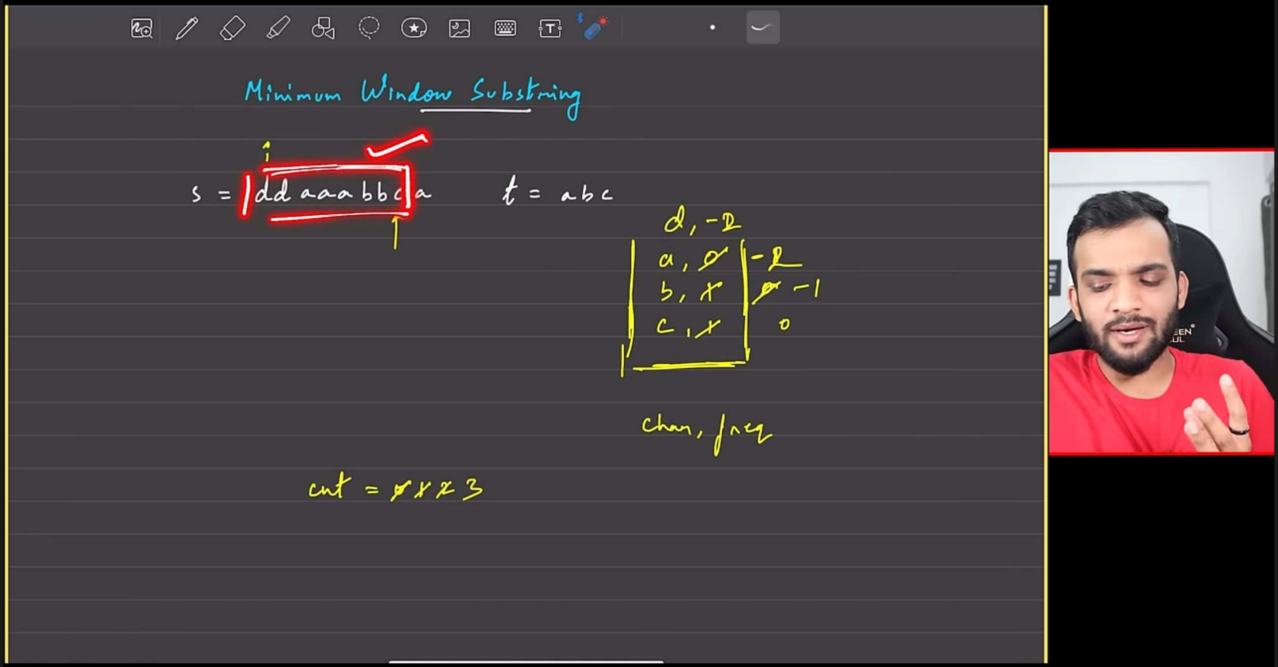
s = dd aaabbca   t = abc

j

d, -2
a, 1
b, 1
c, 1

char, freq

cnt = 0



Minimum Window Substring

s = dd aaa bbc a t = abc

d, -2

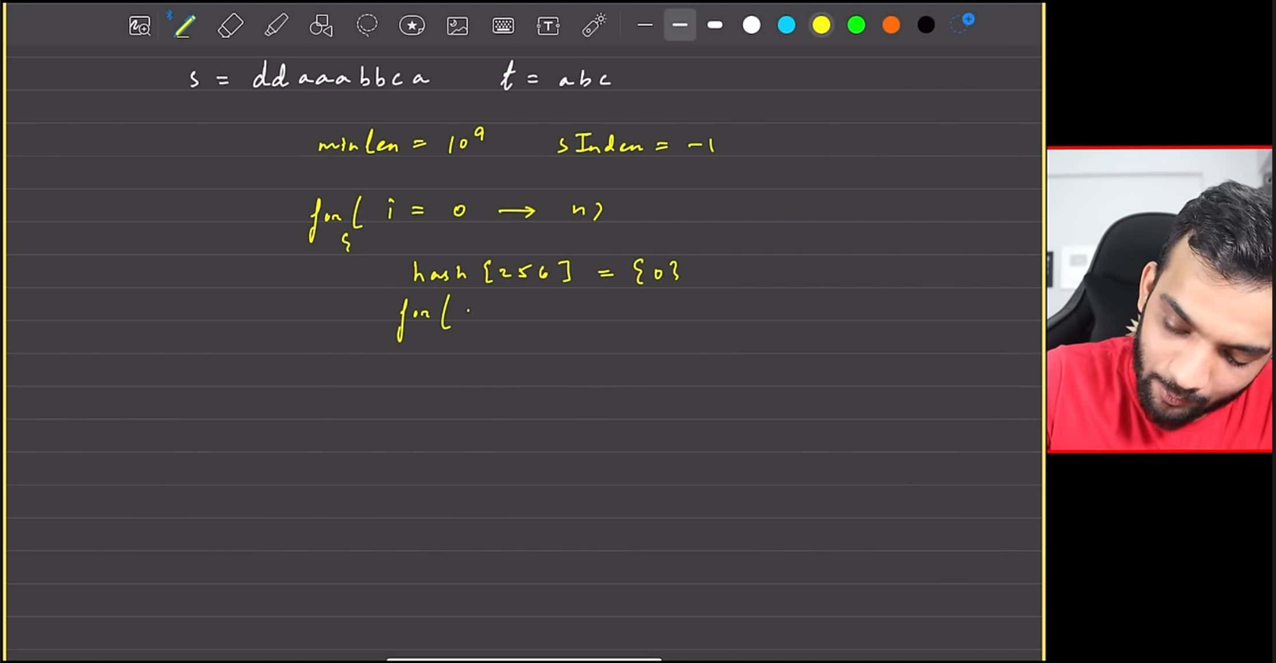
a, ~~0~~ -2

b, ~~1~~ ~~0~~ -1

c, ~~1~~ 0

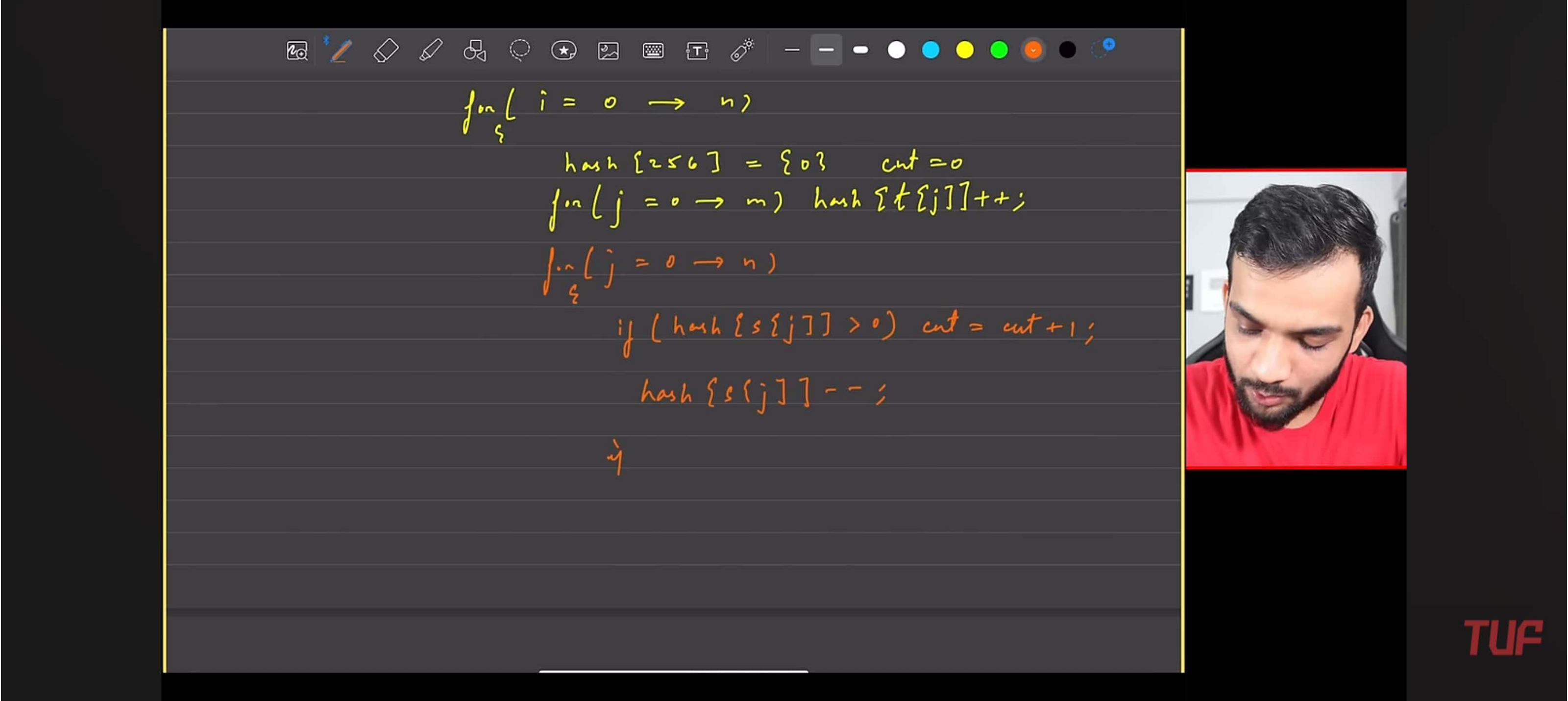
char, freq

cnt = ~~0~~ ~~1~~ ~~2~~ 3

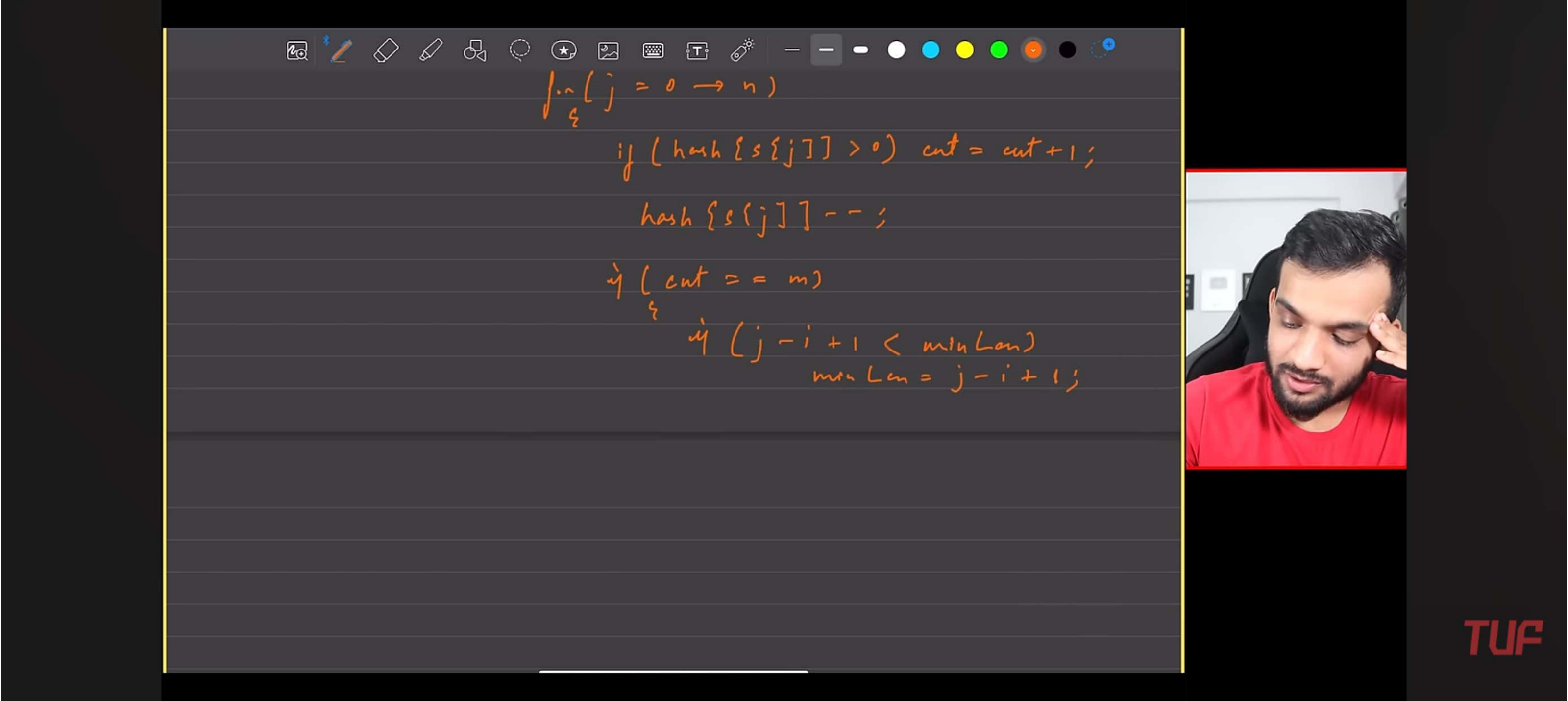


s = ddaaabbca   t = abc

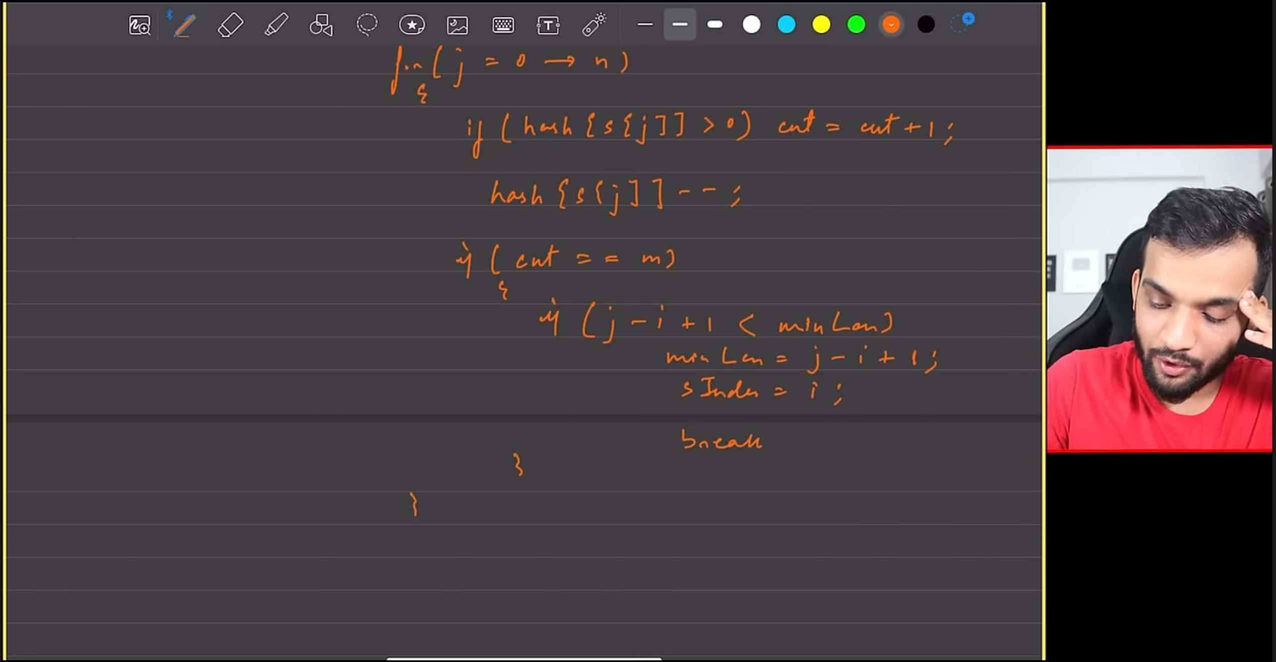
```
minLen = 10^9   sIndex = -1

for( i = 0 → n)
{
    hash[256] = {0}
    for( .
```



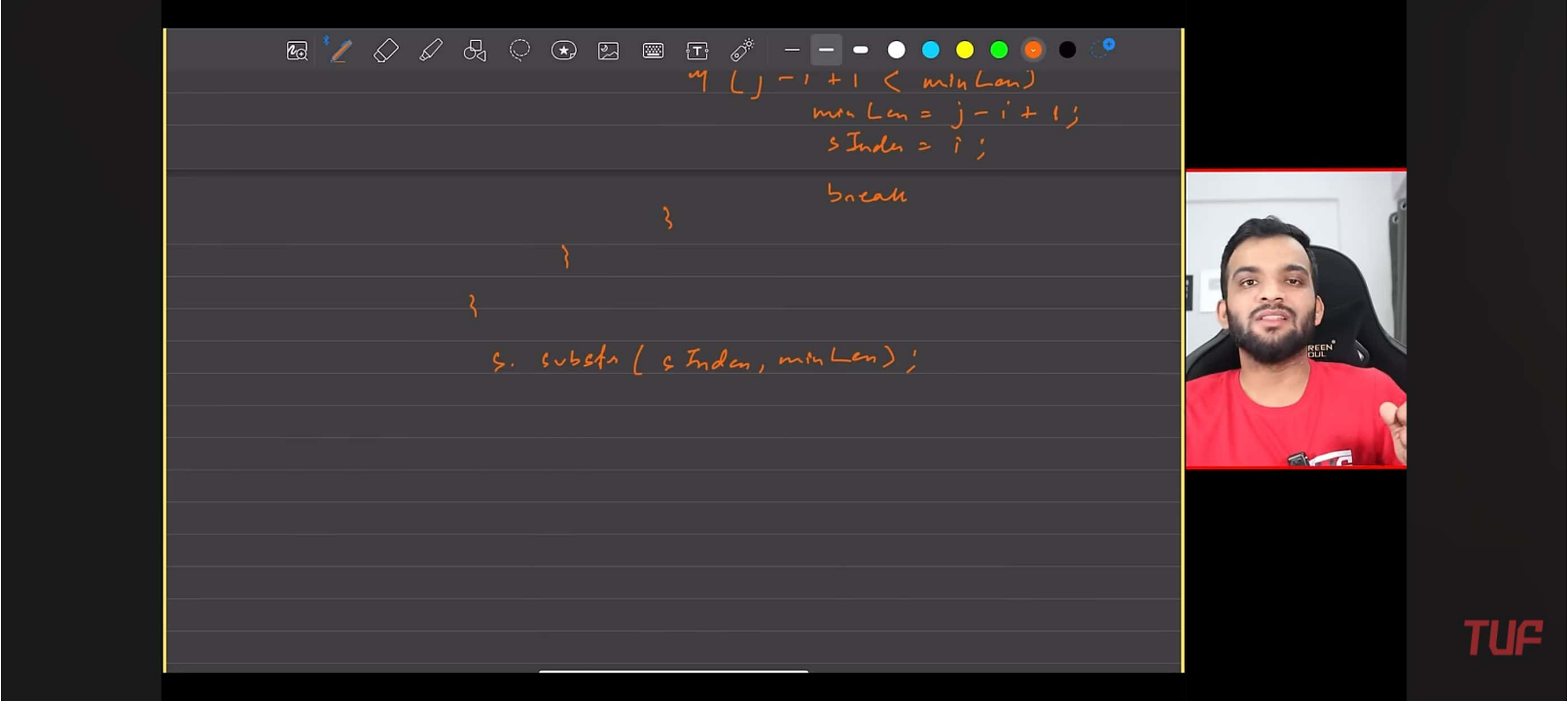
```
for( i = 0 → n)
{
    hash[256] = {0}    cnt = 0
    for(j = 0 → m) hash[t[j]]++;
    for(j = 0 → n)
    {
        if (hash[s[j]] > 0) cnt = cnt+1;
        hash[s[j]]--;
        if
```



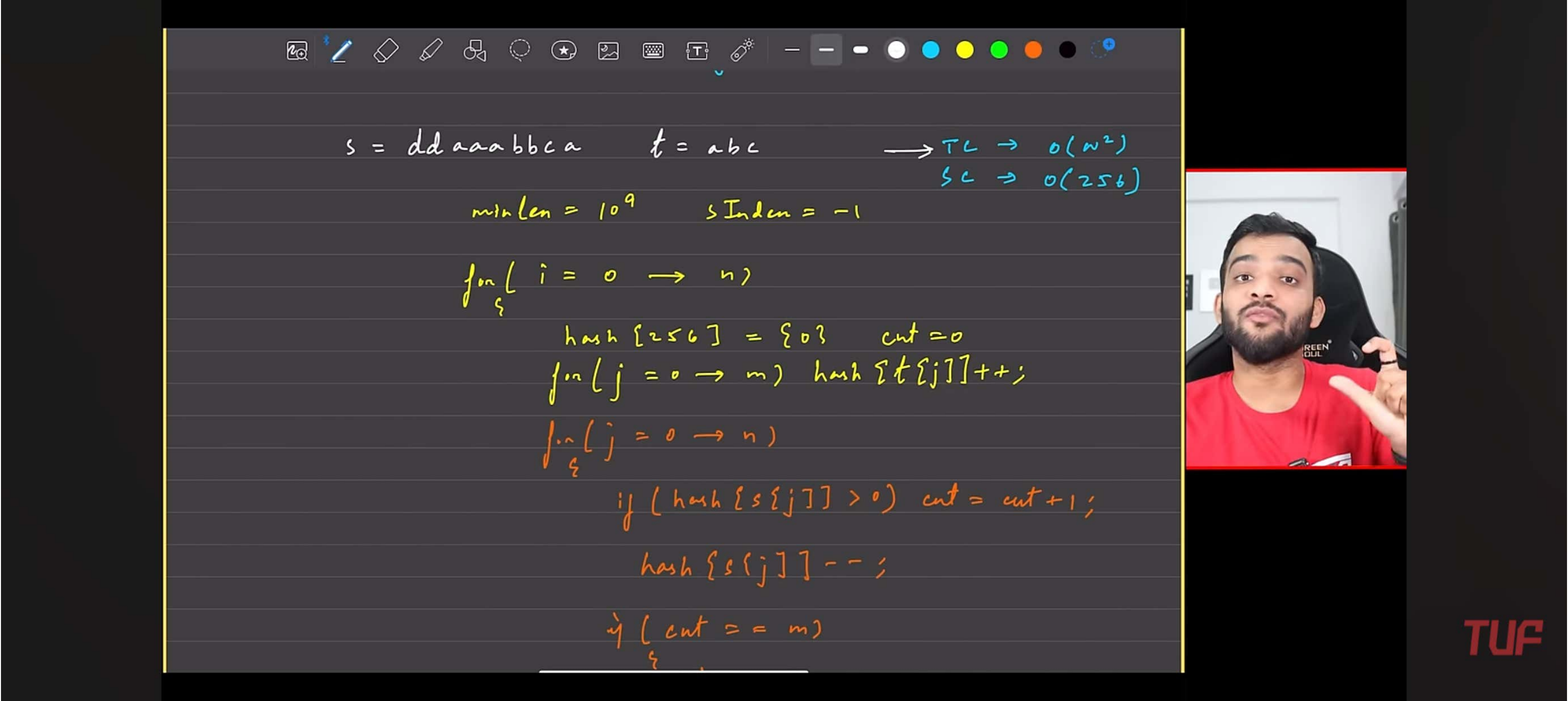
```
for (j = 0 → n)
{
        if (hash[s[j]] > 0) cnt = cnt + 1;

        hash[s[j]] --;

    if (cnt == m)
    {
            if (j - i + 1 < minLen)
                    minLen = j - i + 1;
```



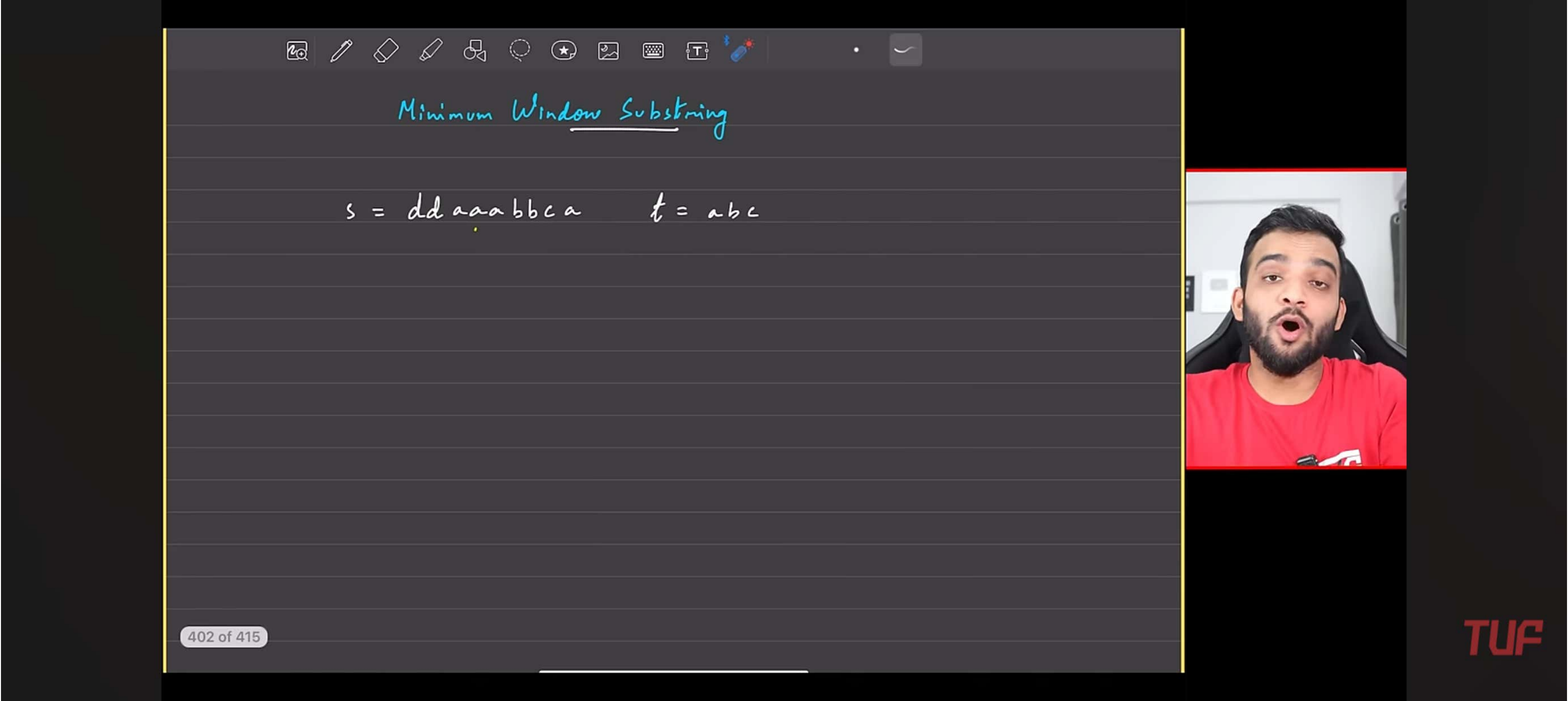
```
for (j = 0 → n)
{
        if (hash[s[j]] > 0) cnt = cnt + 1;

         hash[s[j]]--;

        if (cnt == m)
        {
              if (j - i + 1 < minLen)
                        minLen = j - i + 1;
                         sIndex = i;

                        break
        }
}
```



```
if (j - i + 1 < minLen)
        minLen = j - i + 1;
        sIndex = i;

        break
    }
  }
}

s. substr ( sIndex, minLen);
```

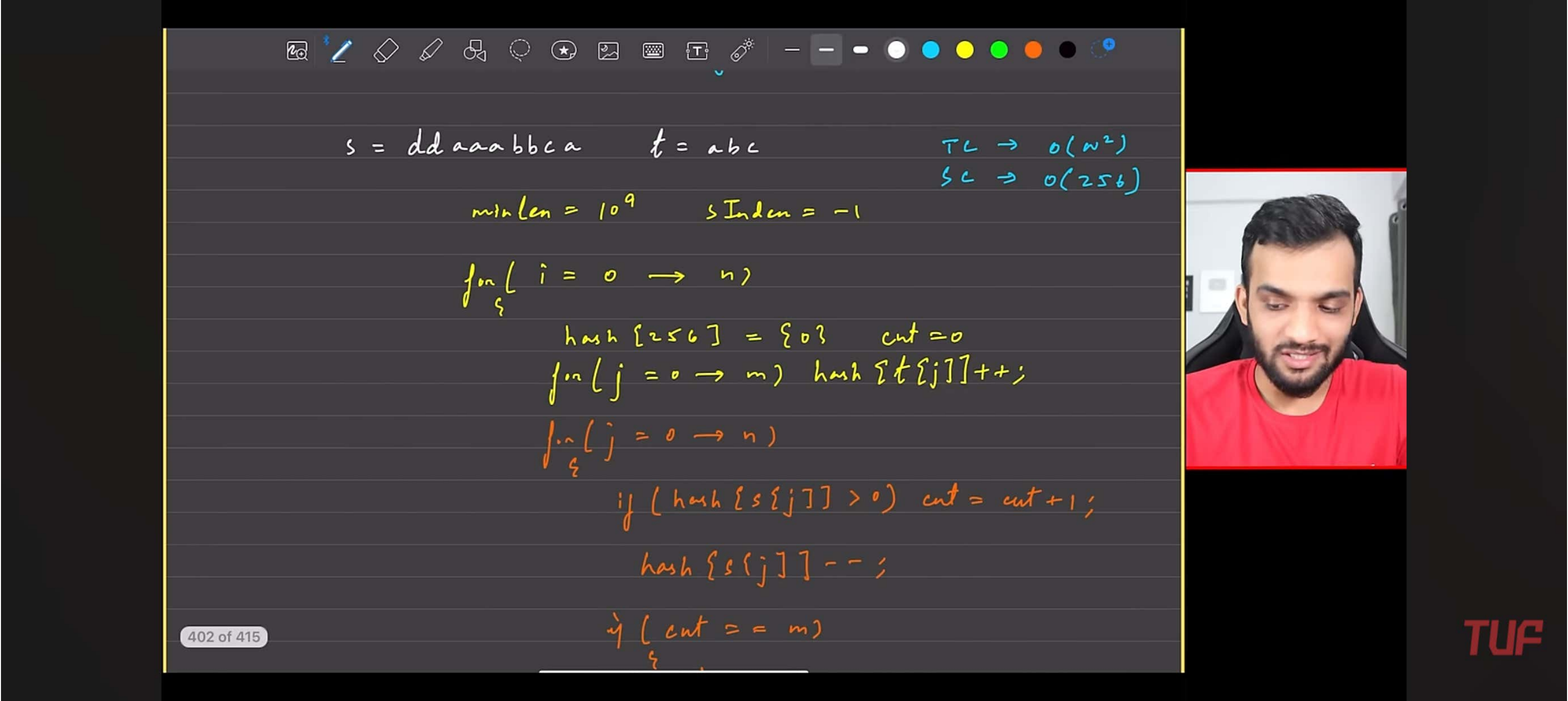


```
s = ddaaabbca      t = abc          → TC → O(N^2)
                                      SC → O(256)

minLen = 10^9      sIndex = -1

for( i = 0 → n)
{
    hash[256] = {0}     cnt = 0
    for(j = 0 → m) hash[t[j]]++;
    for(j = 0 → n)
    {
        if (hash[s[j]] > 0) cnt = cnt+1;
        hash[s[j]]--;
        if (cnt == m)
        {
```

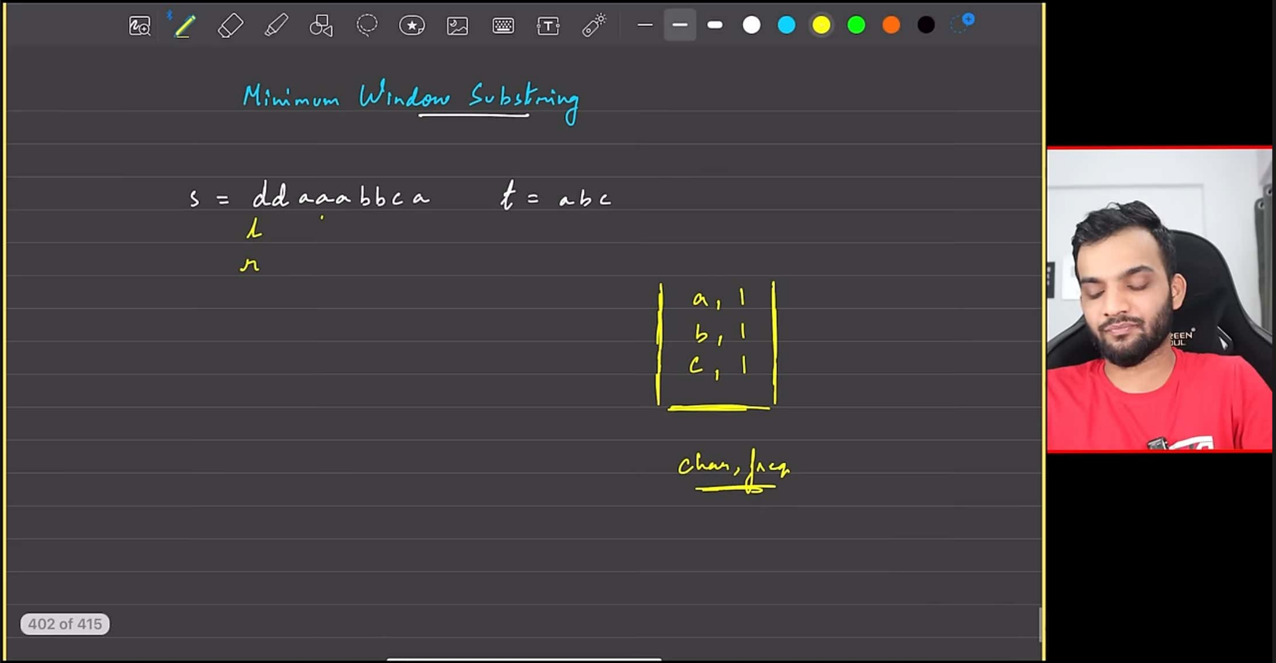


Minimum Window Substring

s = dd aaa bbc a   t = abc



```
s = ddaaabbca        t = abc                 TC → O(N²)
                                             SC → O(256)
        minLen = 10^9        sIndex = -1

        for( i = 0 → n)
        {
            hash[256] = {0}      cnt = 0
            for(j = 0 → m) hash[t[j]]++;
            for(j = 0 → n)
            {
                if (hash[s[j]] > 0) cnt = cnt + 1;
                hash[s[j]]--;
                if (cnt == m)
                {
```



Minimum Window Substring

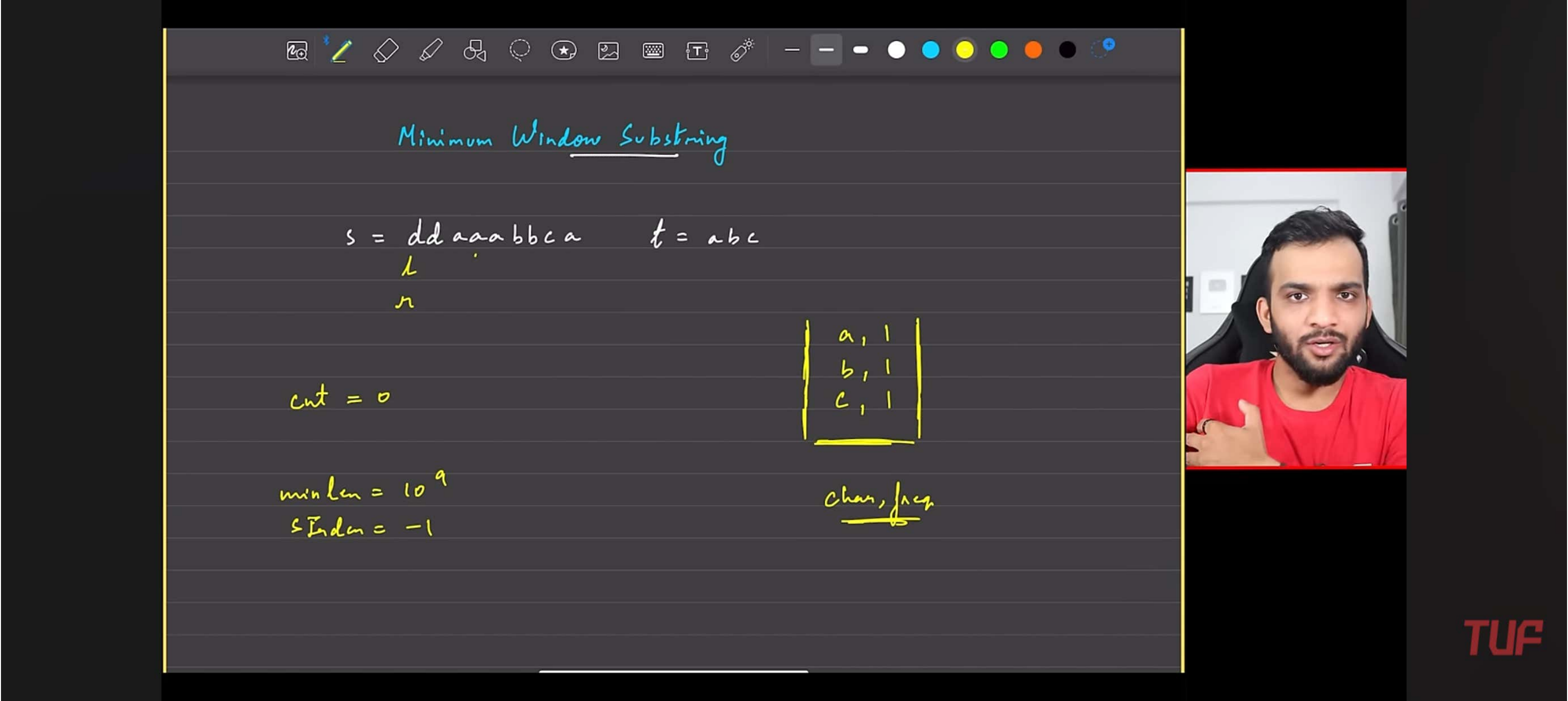
s = ddaaabbca   t = abc

l

r

a	1
b	1
c	1

char, freq



Minimum Window Substring

s = ddaaabbca   t = abc

l

r

cnt = 0

minLen = 10^9

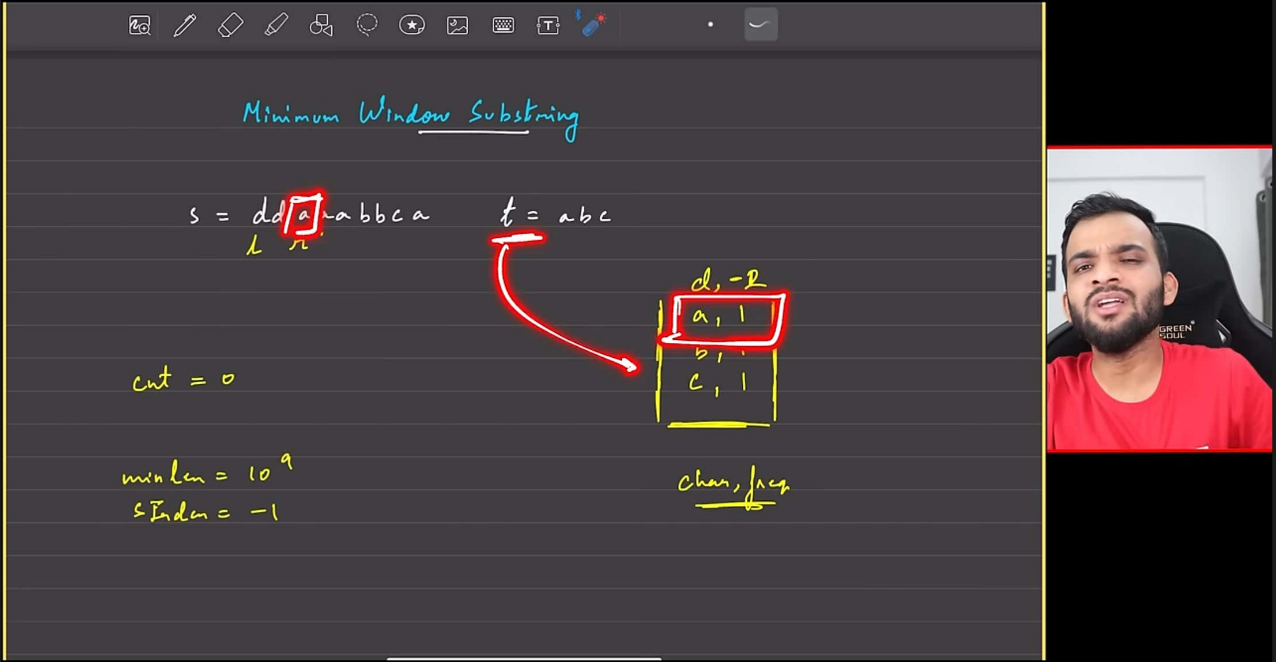
sIndex = −1

a, 1

b, 1

c, 1

char, freq



Minimum Window Substring

s = ddaaabbca   t = abc

l r

d, -2

a, 1

b, 1

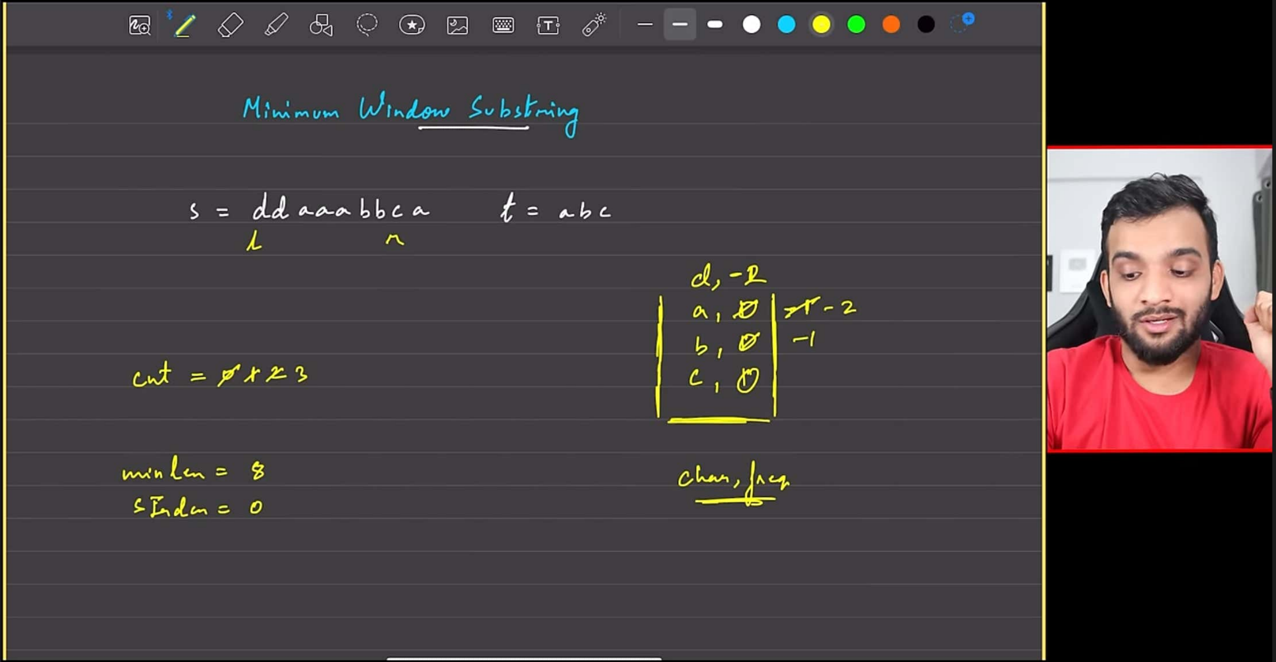
c, 1

char, freq

cnt = 0

minlen = 10^9

sIndex = -1



Minimum Window Substring

s = ddaaabbca   t = abc

l   r

d, -2

a, ~~0~~ ~~-1~~ -2

b, ~~0~~ -1

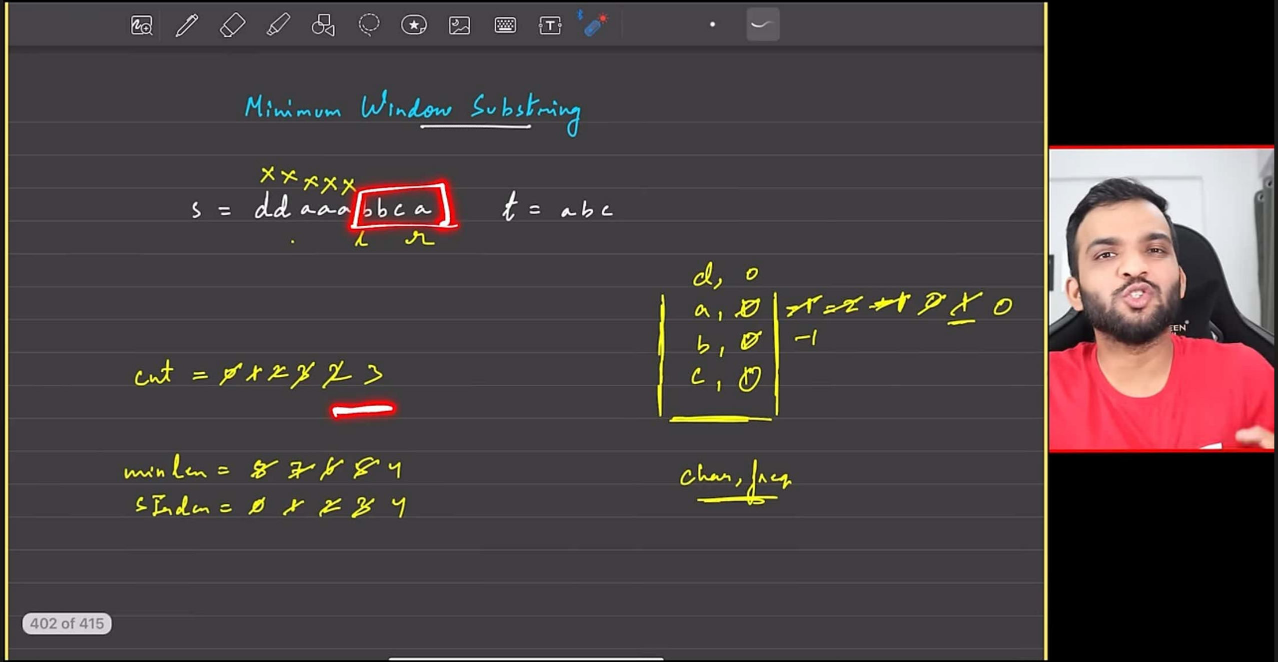
c, ~~1~~ 0

char, freq

cnt = ~~0~~ ~~1~~ ~~2~~ 3

minLen = 8

sIndex = 0



Minimum Window Substring

s = ddaaabbca   t = abc

l   r

cnt = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~2~~ 3

minLen = ~~8~~ ~~7~~ ~~6~~ ~~5~~ 4

sIndex = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

d, 0

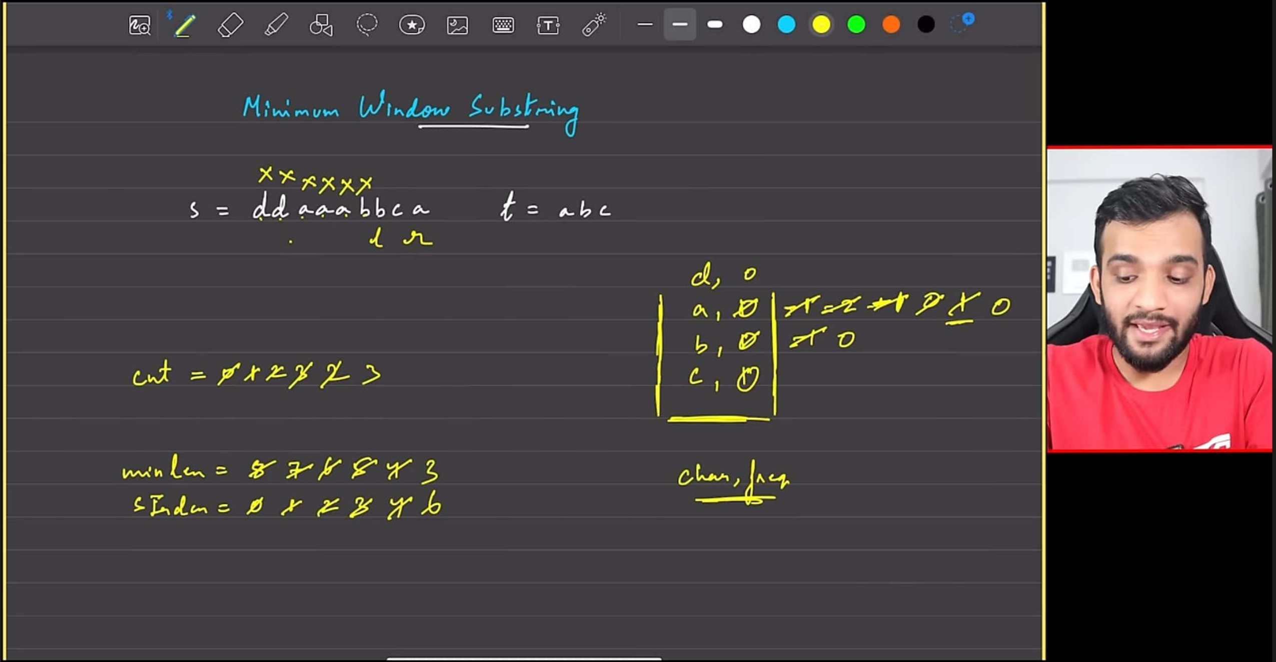
a, ~~0~~   ~~-1~~ ~~-2~~ ~~-1~~ ~~0~~ ~~1~~ 0

b, ~~0~~   -1

c, ~~0~~

char, freq

402 of 415



Minimum Window Substring

s = ddaaabbca   t = abc

l r

d, 0

a, ~~0~~ | ~~-1~~ ~~-2~~ ~~-1~~ ~~0~~ ~~1~~ 0

b, ~~0~~ | ~~-1~~ 0

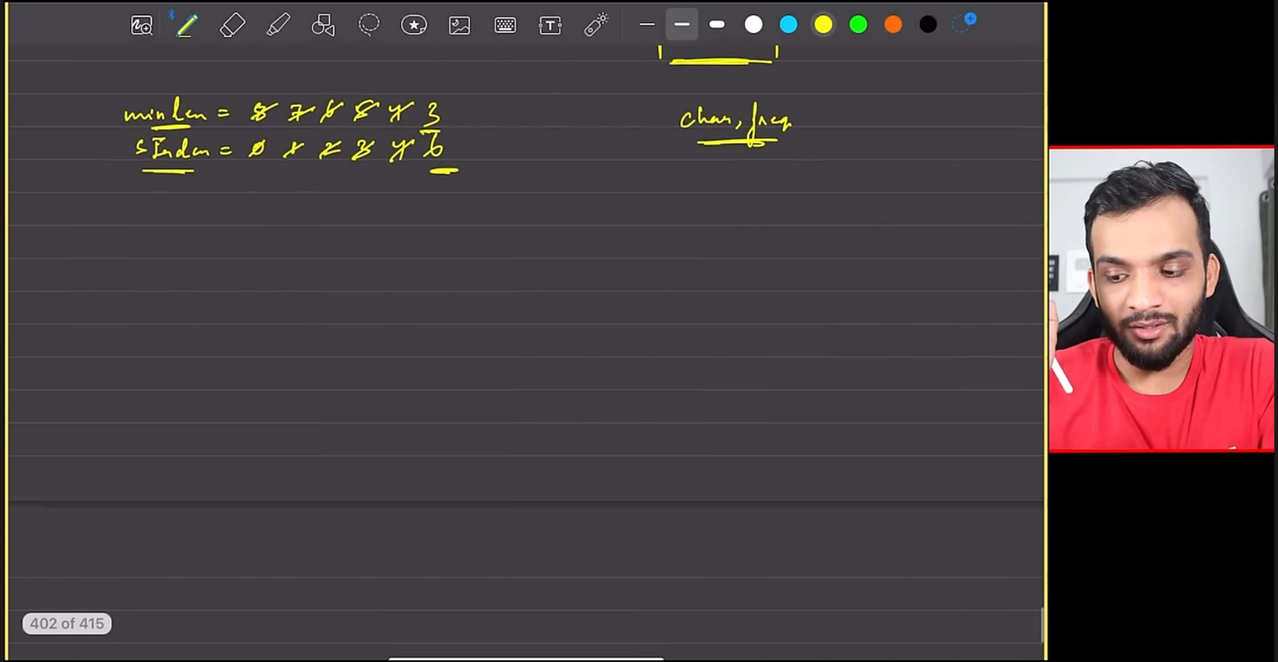
c, 0

char, freq

cnt = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~2~~ 3

minLen = ~~8~~ ~~7~~ ~~6~~ ~~5~~ ~~4~~ 3

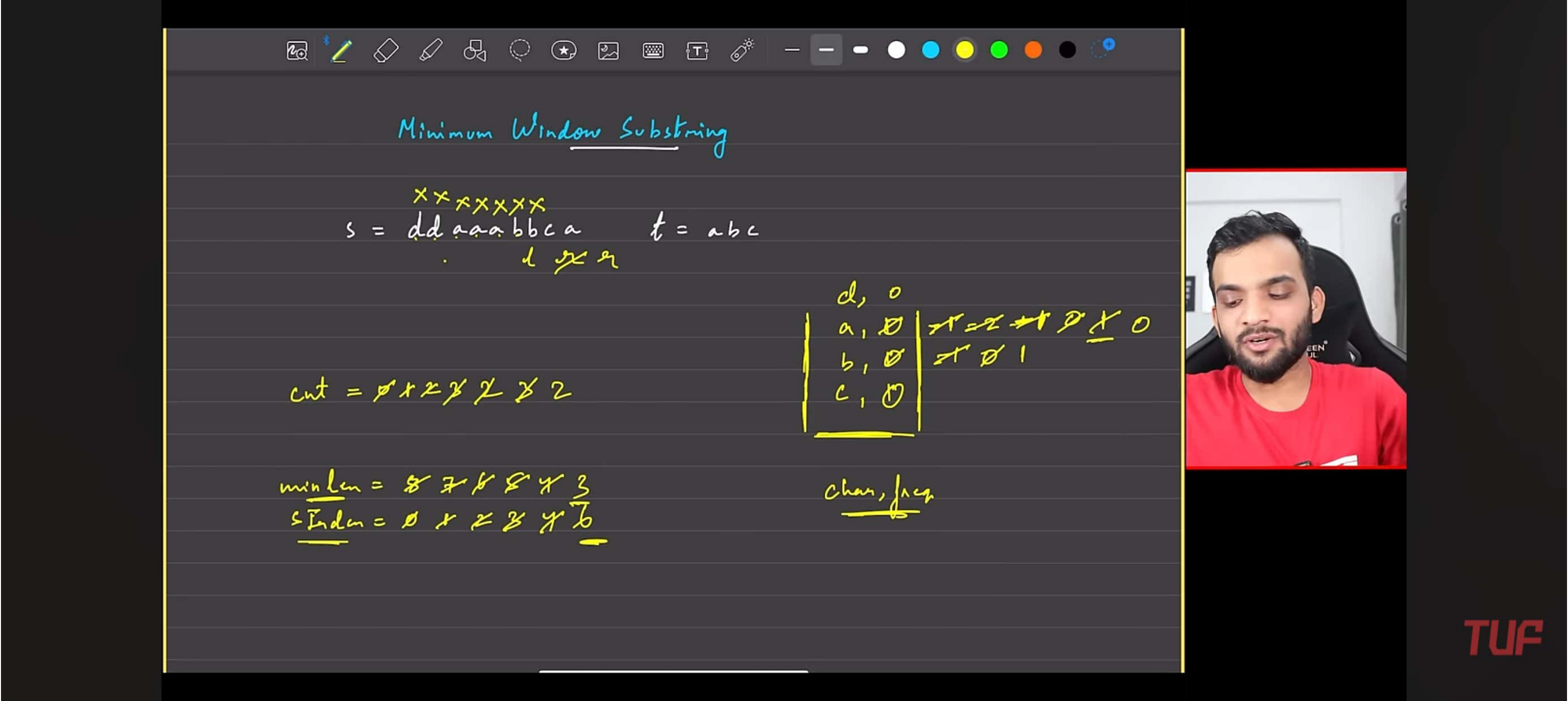
sIndex = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 6



minLen = ~~8~~ ~~7~~ ~~6~~ ~~5~~ ~~4~~ 3

sIndex = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 6

char, freq



Minimum Window Substring

× × × × × × ×

s = dd aaabbca   t = abc

l ~~r~~ r

cnt = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~2~~ ~~3~~ 2

minLen = ~~8~~ ~~7~~ ~~6~~ ~~5~~ ~~4~~ 3

sIndex = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 6

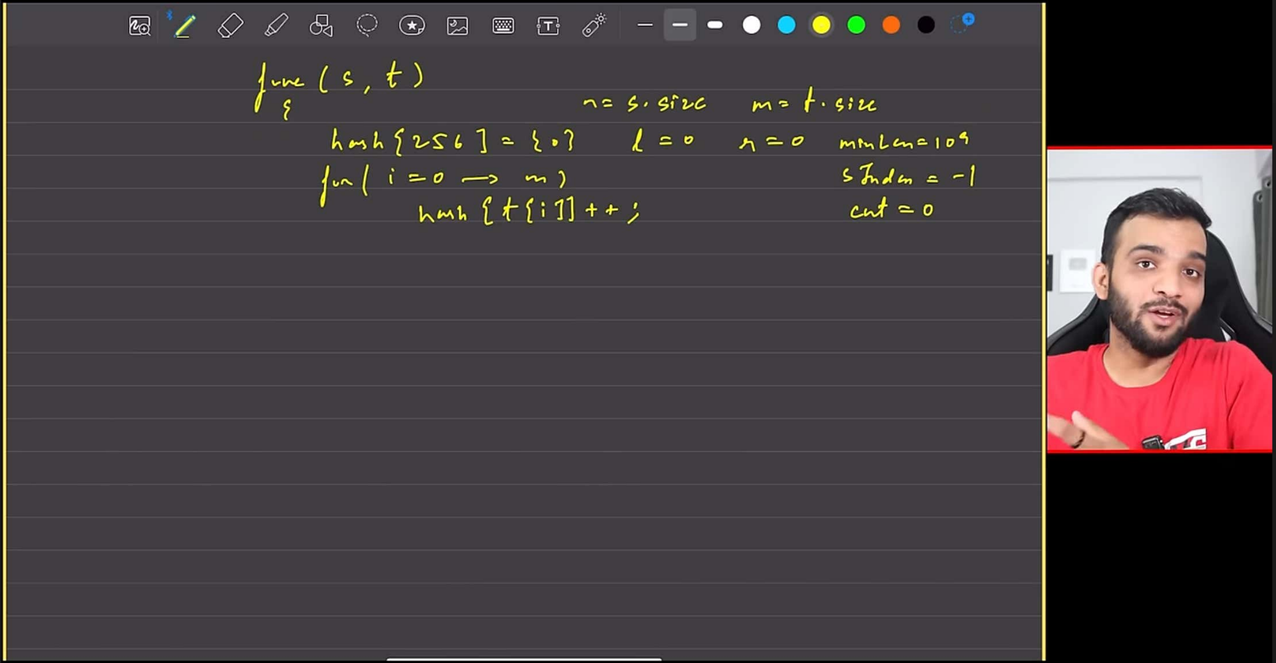
d, 0

a, ~~0~~ ~~-1~~ ~~-2~~ ~~-1~~ ~~0~~ ~~1~~ 0

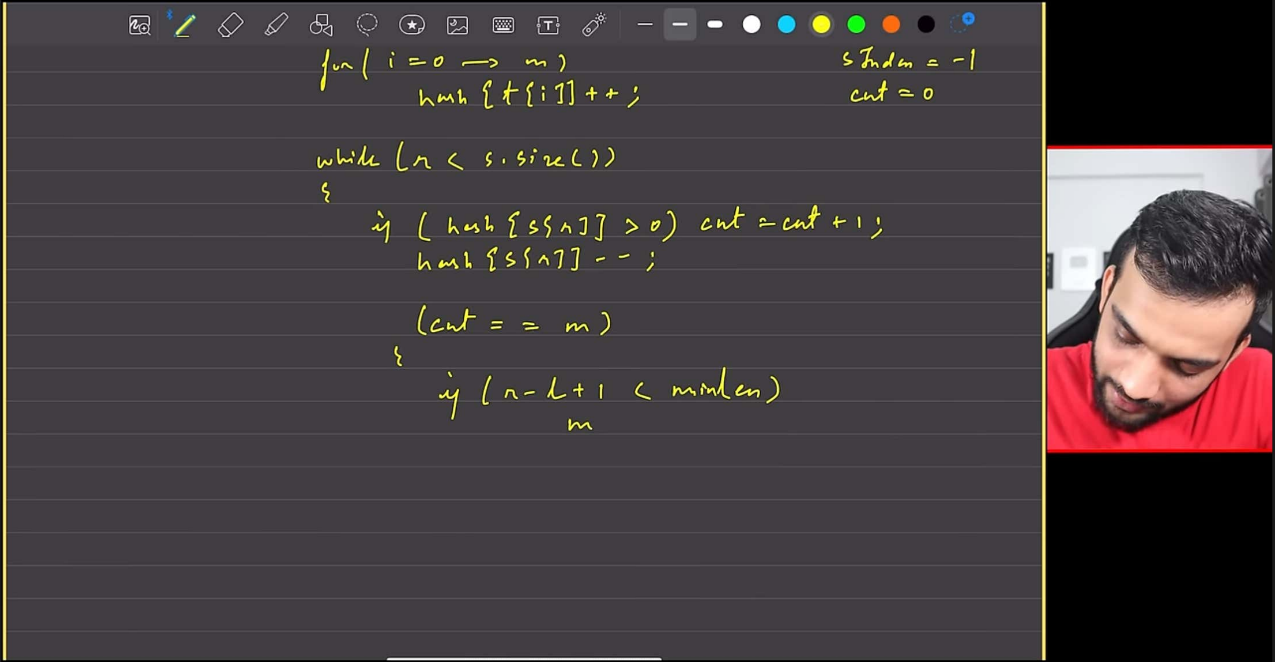
b, ~~0~~ ~~-1~~ ~~0~~ 1

c, ~~0~~

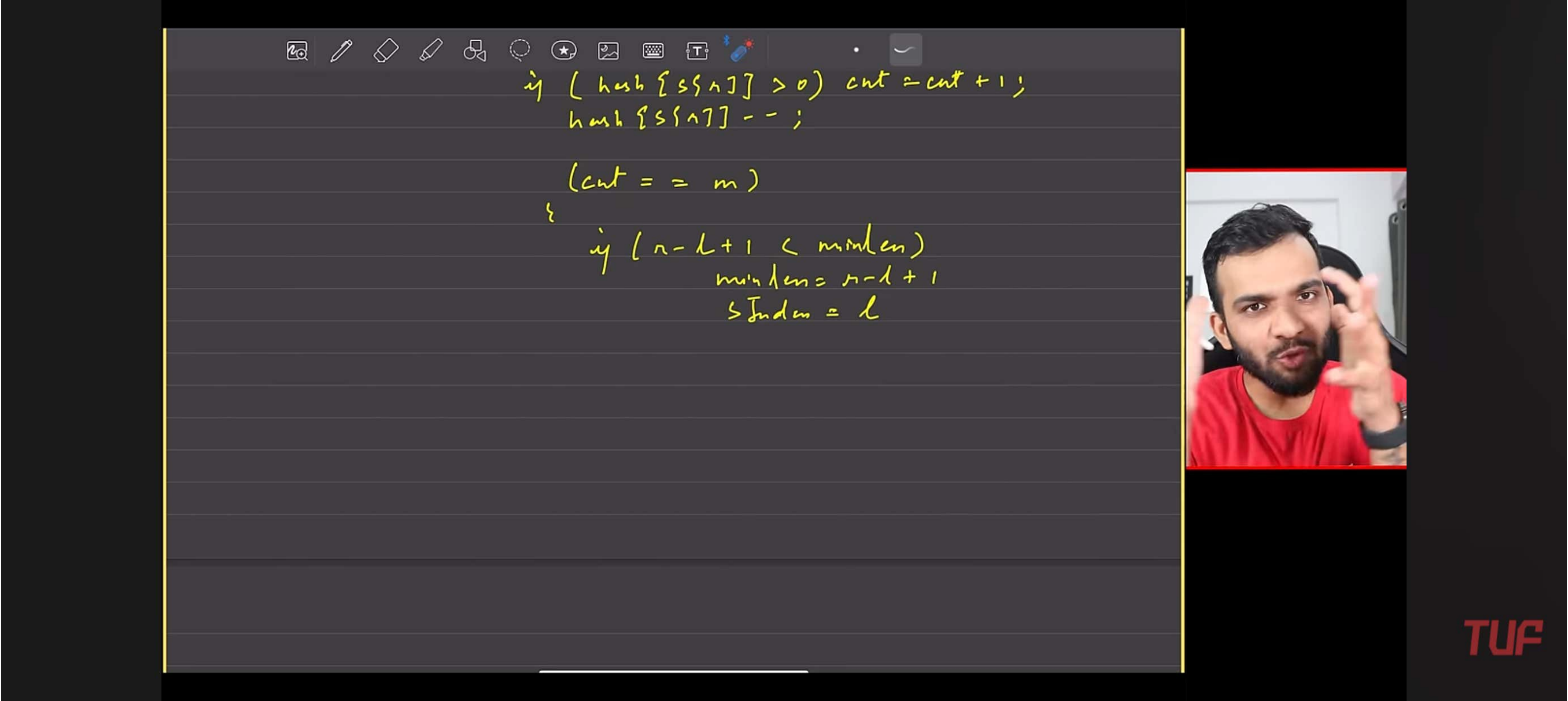
char, freq



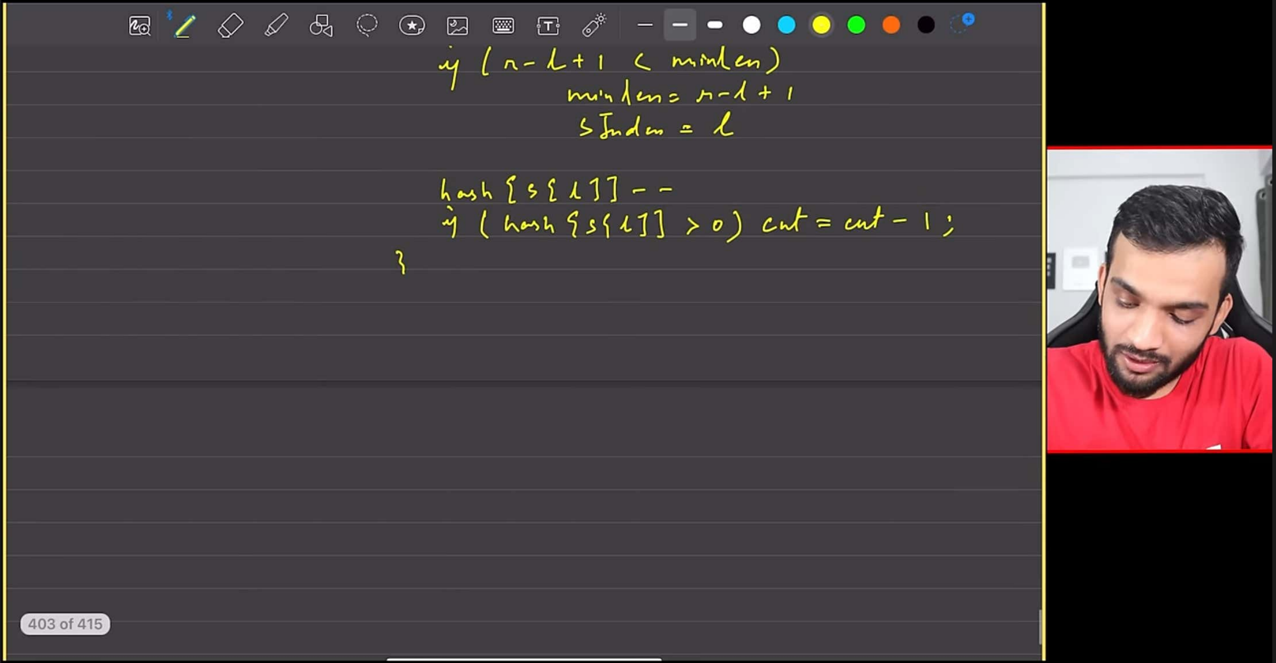
```
func ( s , t )
{
                          n = s.size      m = t.size
      hash[256] = {0}     l = 0     r = 0     minLen = 10^9
   for( i = 0 → m )                          sIndex = -1
          hash[t[i]]++;                      cnt = 0
```



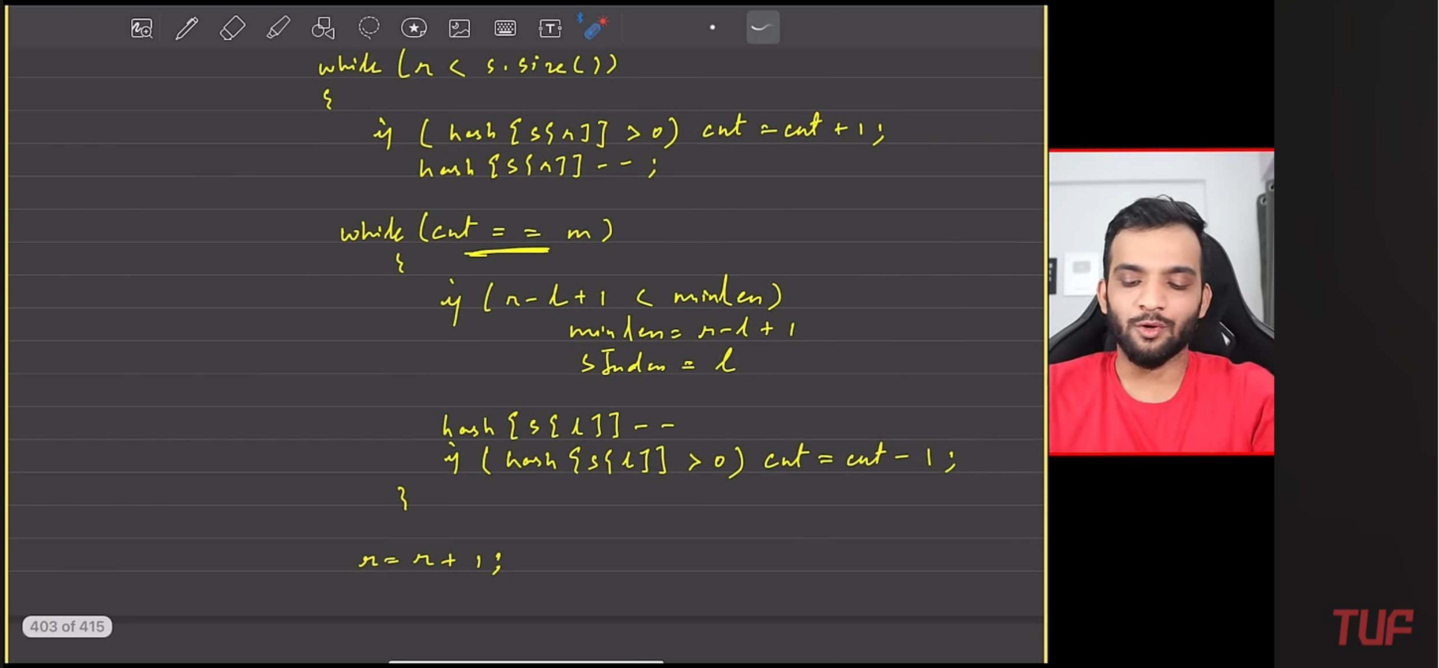
```
for (i=0 → m)                         sIndex = -1
        hash[t[i]]++;                  cnt = 0

while (r < s.size())
{
    if (hash[s[r]] > 0) cnt = cnt + 1;
        hash[s[r]]--;

        (cnt == m)
    {
        if (r-l+1 < minLen)
                m
```



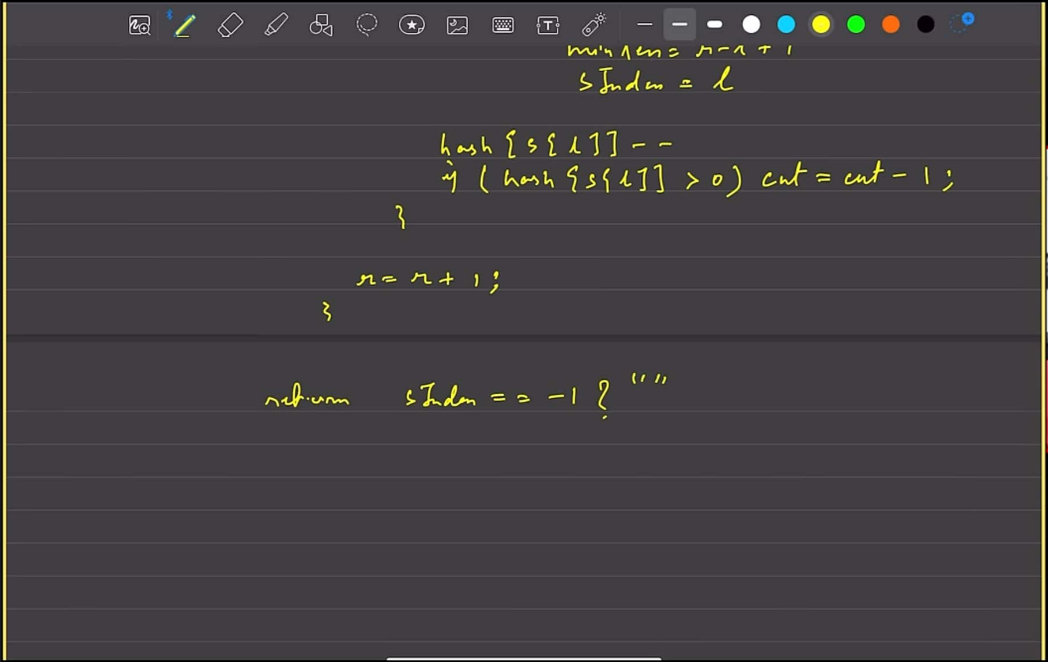
```
if ( hash[s[r]] > 0) cnt = cnt + 1;
    hash[s[r]] - - ;

    (cnt = = m)
  {
      if (r-l+1 < minLen)
              minLen = r-l + 1
              sIndex = l
```



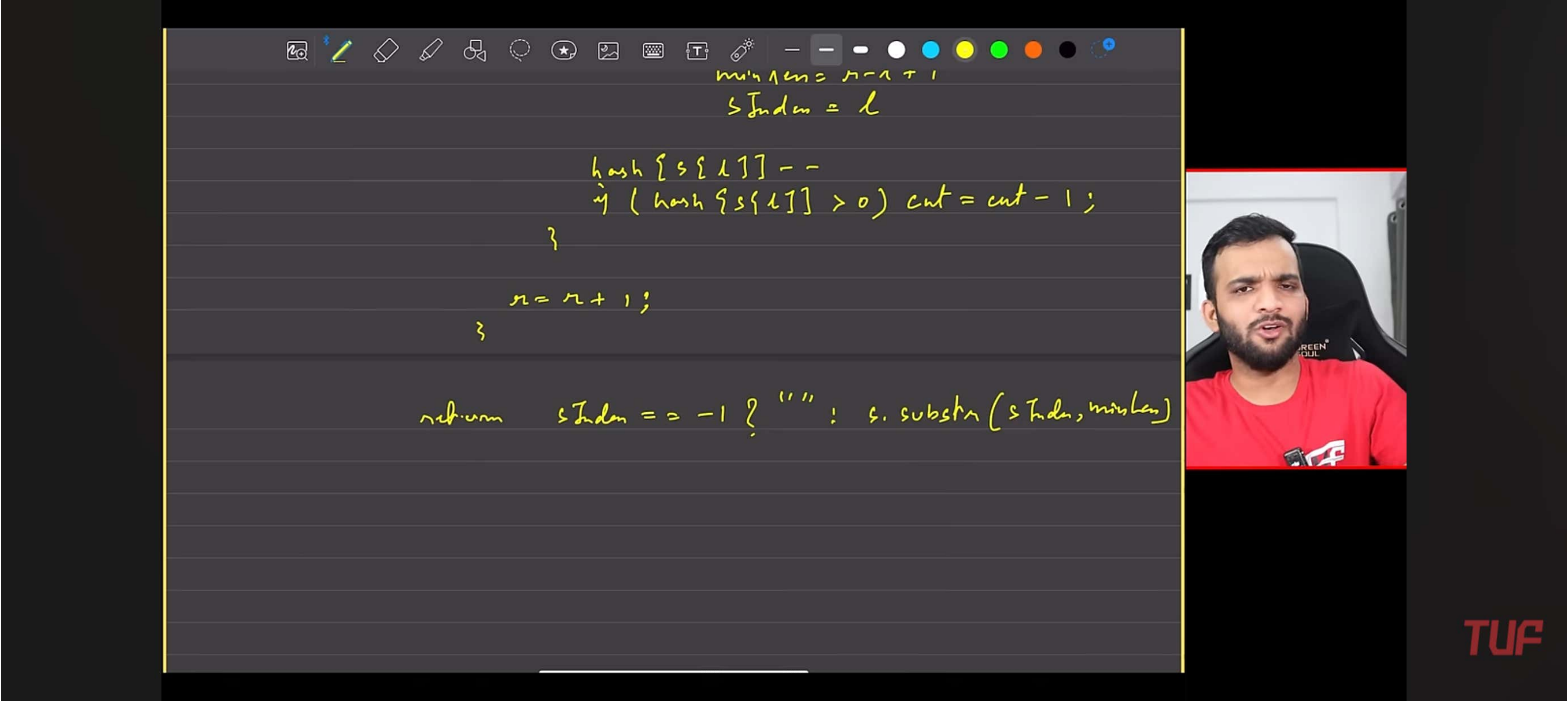
```
        if (r-l+1 < minlen)
                minlen= r-l+1
                sIndex = l

        hash[s[l]]--
        if (hash[s[l]] > 0) cnt = cnt - 1;
    }
```



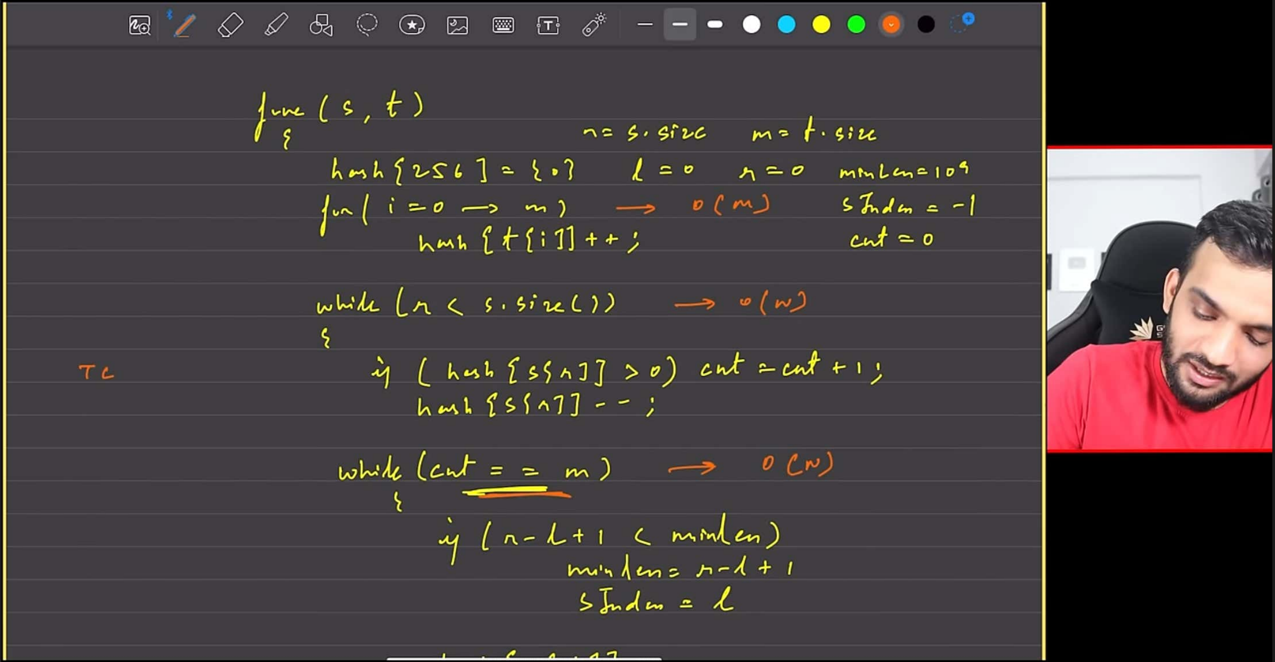
```
while (r < s.size())
{
    if (hash[s[r]] > 0) cnt = cnt + 1;
    hash[s[r]] - - ;

  while (cnt = = m)
    {
        if (r-l+1 < minlen)
                minlen = r-l+1
                sIndex = l

        hash[s[l]] - -
        if (hash[s[l]] > 0) cnt = cnt - 1;
    }

  r = r + 1;
```



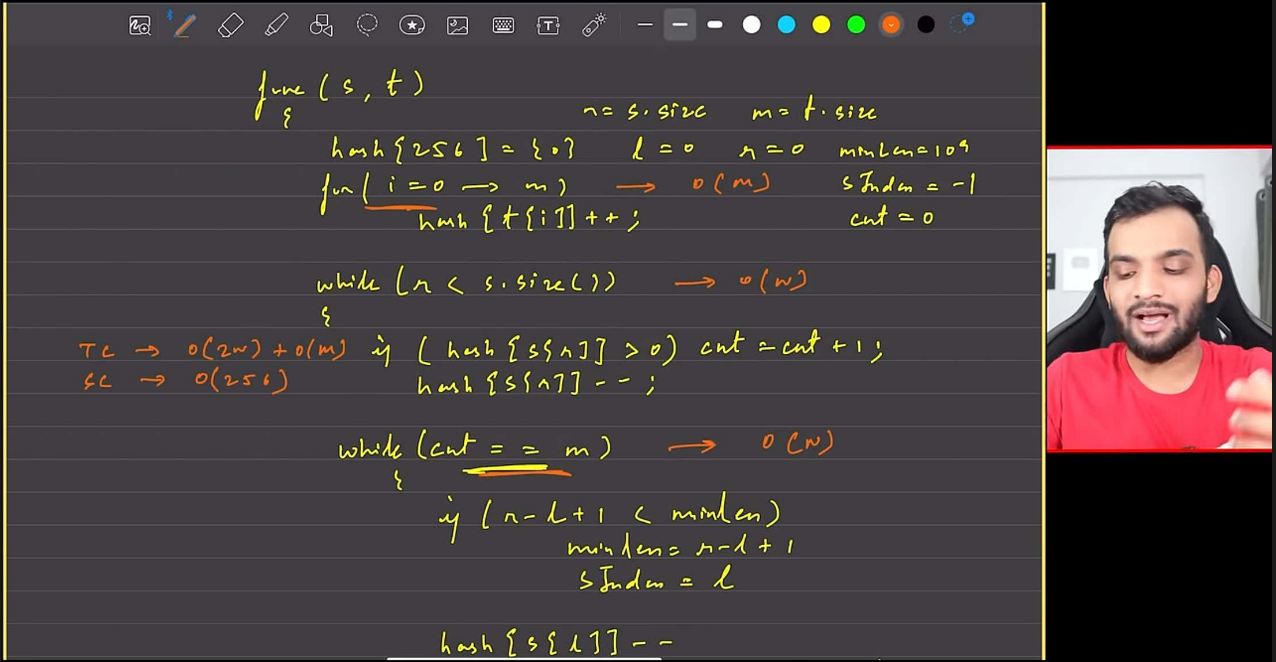
```
                         minlen = r-l+1
                         sIndex = l

                    hash[s[l]]--
                    if (hash[s[l]] > 0) cnt = cnt - 1;
                 }

              r = r + 1;
           }

     return   sIndex == -1 ? ""
```



```
                    minlen = r-l+1
                    sIndex = l

                hash[s[l]]--
                if (hash[s[l]] > 0) cnt = cnt - 1;
            }

        r = r+1;
    }

return   sIndex == -1 ? "" : s.substr(sIndex, minLen)
```

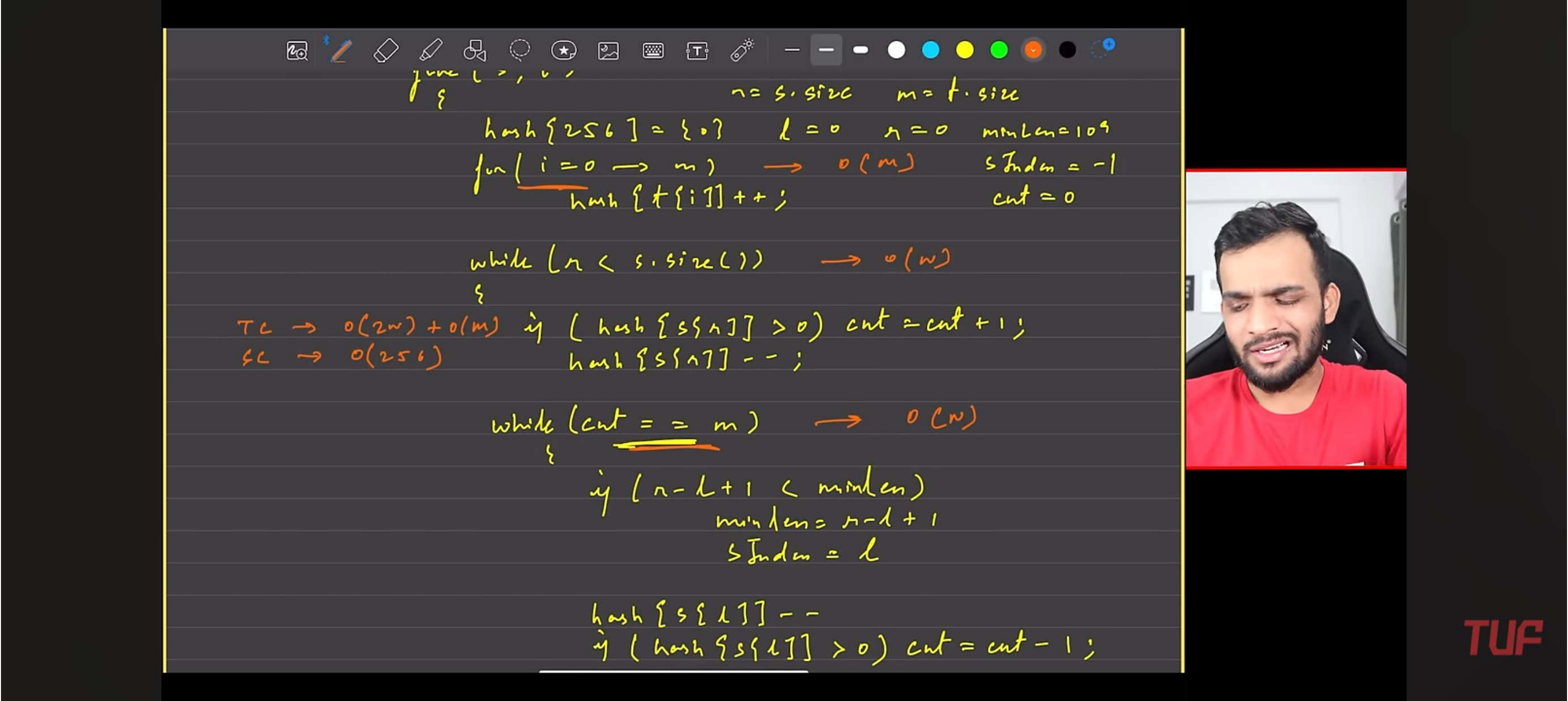


```
func ( s , t )
{                          n = s.size    m = t.size
    hash[256] = {0}    l = 0    r = 0    minLen = 10^9
    for( i = 0 -> m )   ->  O(m)         sIndex = -1
        hash[t[i]]++;                    cnt = 0

    while (r < s.size())   ->  O(n)
    {
TC      if ( hash[s[r]] > 0) cnt = cnt + 1;
        hash[s[r]]--;

        while (cnt == m)   ->  O(n)
        {
            if (r-l+1 < minlen)
                minlen = r-l+1
                sIndex = l
```



```
func ( s , t )
{
                         n = s.size    m = t.size
    hash[256] = {0}    l = 0    r = 0    minLen = 10^9
    for ( i = 0 -> m )   ->  O(m)        sIndex = -1
        hash[t[i]]++;                    cnt = 0

    while ( r < s.size() )   ->  O(N)
    {
TC -> O(2N) + O(m)   if ( hash[s[r]] > 0 ) cnt = cnt + 1;
SC -> O(256)         hash[s[r]]--;

        while ( cnt = = m )   ->  O(N)
        {
            if ( r-l+1 < minLen )
                minLen = r-l+1
                sIndex = l

            hash[s[l]]--
```

TUF



```
func(s, t)
{
                         n = s.size      m = t.size
      hash[256] = {0}     l = 0     r = 0     minLen = 10^9
      for(i = 0 → m)    → O(m)               sIndex = -1
            hash[t[i]]++;                    cnt = 0

      while(r < s.size())     → O(n)
      {
TC → O(2n) + O(m)   if(hash[s[r]] > 0) cnt = cnt + 1;
SC → O(256)         hash[s[r]]--;

            while(cnt == m)     → O(n)
            {
                  if(r - l + 1 < minLen)
                        minLen = r - l + 1
                        sIndex = l

                  hash[s[l]]--
                  if(hash[s[l]] > 0) cnt = cnt - 1;
```