Longest substring without repeating characters

s = c a d b z a b c d

Longest substring without repeating characters

s = c a d b z a b c d

- c
- ca
- cad
- cadb
- cadbz

Longest substring without repeating characters

s = c a d b z a b c d

a
a d
a b b
a d b z

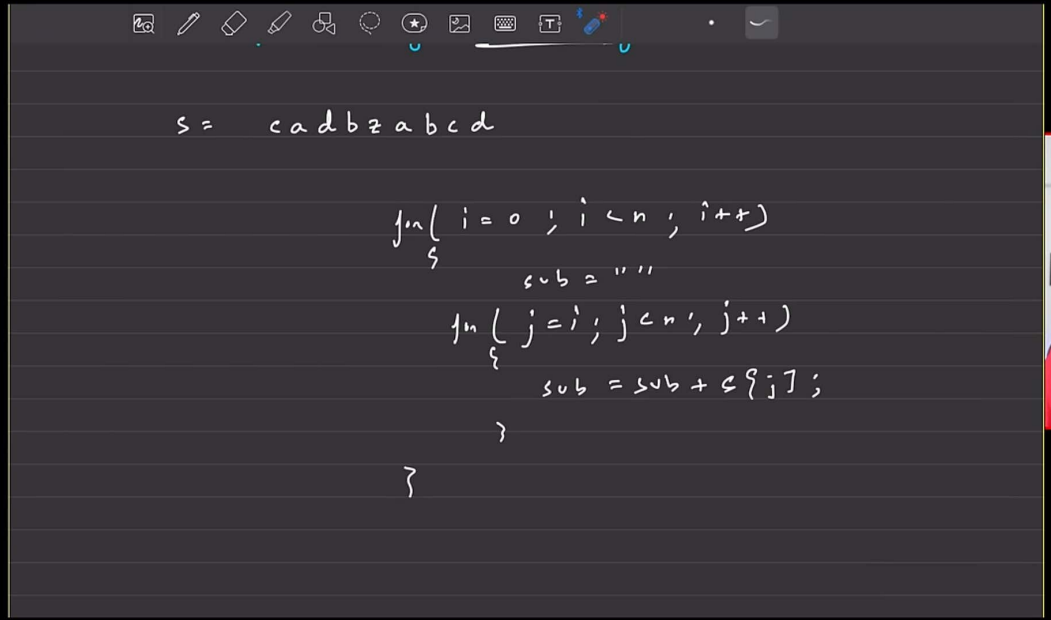
Longest substring without repeating characters

s = c a d b z a b c d

d

d b

d b z

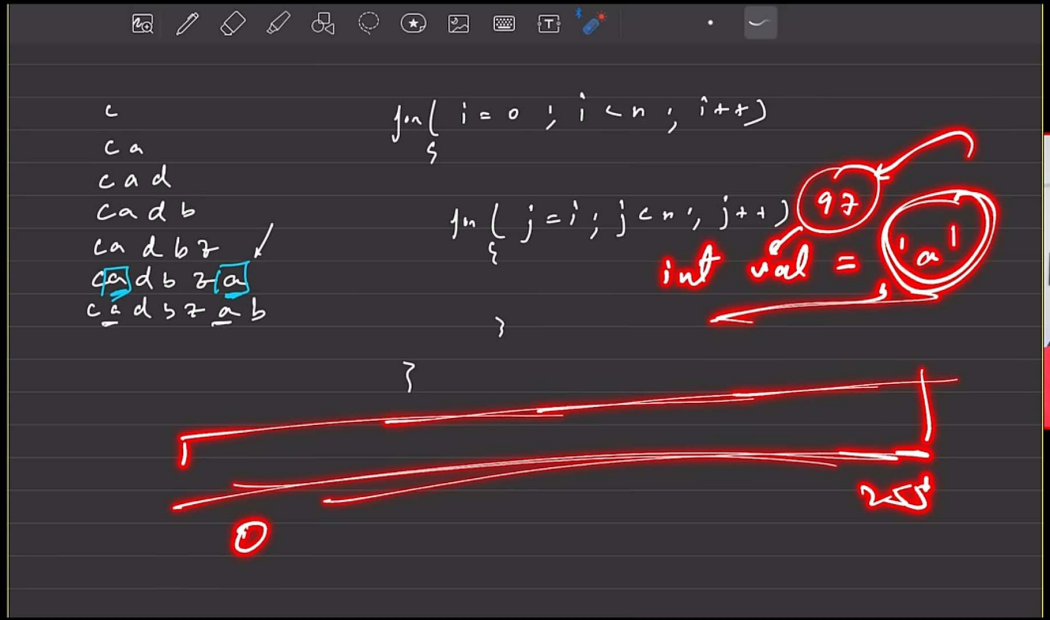

s = cadbzabcd
for(i = 0 ; i < n ; i++)
{
sub = ""
for(j = i ; j < n ; j++)
{
sub = sub + s[j] ;
}
}

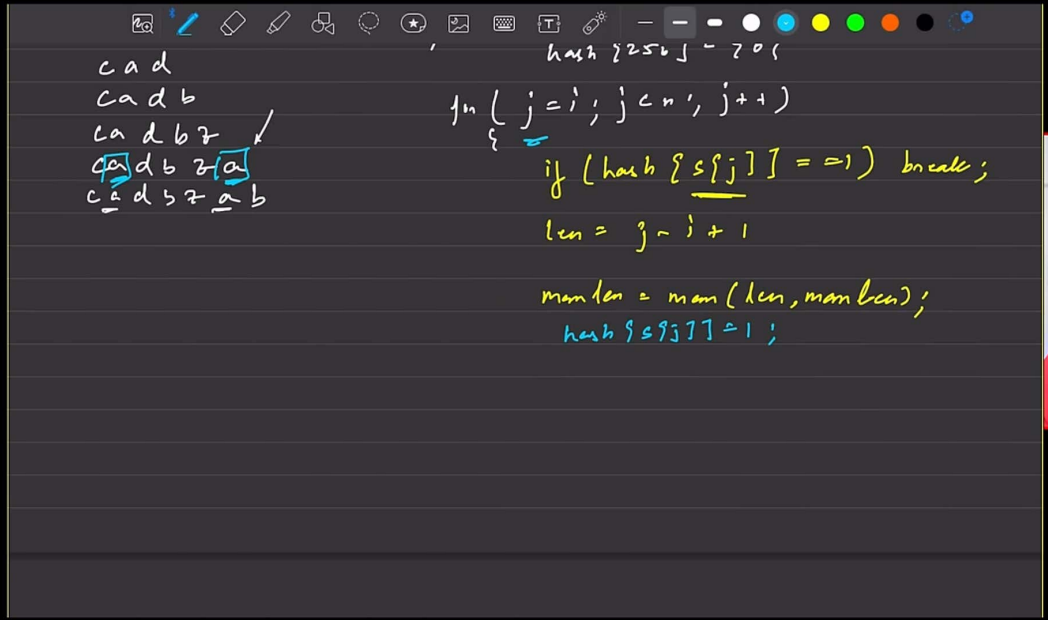
Longest substring without repeating characters

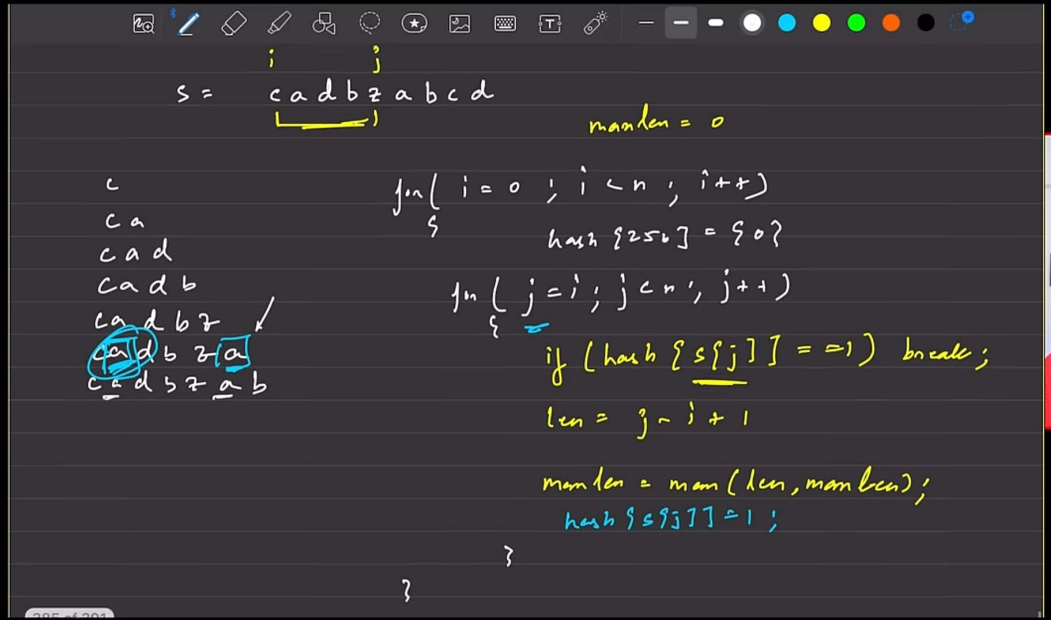
s = c a d b z a b c d

c
ca
cad
cadb
cadbz
cadbza

```
for( i = 0 ; i < n ; i++)
{
    sub = ""
    for( j = i ; j < n ; j++)
    {
        sub = sub + s[j] ;
    }
}
```

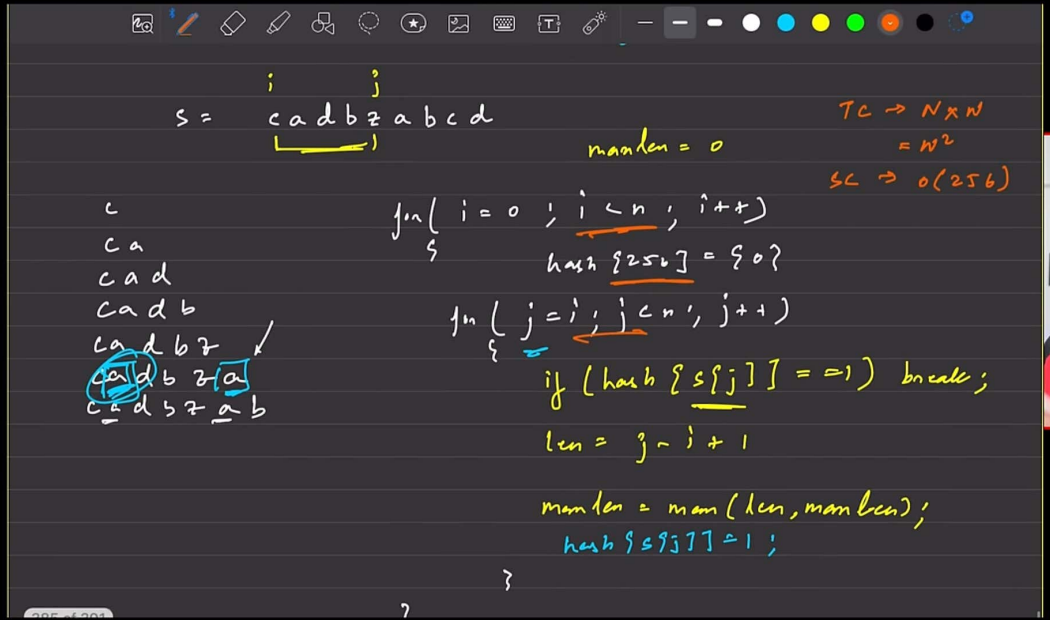

c
ca
cad
cadb
cadbz
cadbza
cadbzab
for(i=0 ; i<n ; i++)
{
for(j=i ; j<n ; j++)
{
}
}
97
'a'
int val =
2nd
0

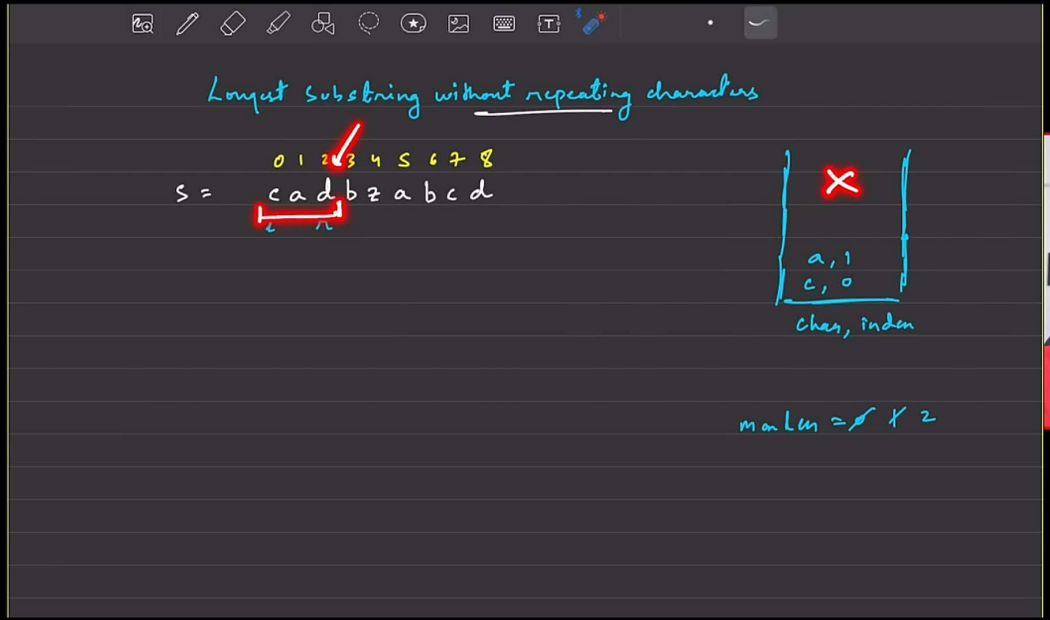

cad
cadb
cadbz
cadbza
cadbzab
hash[256] = {0};
for (j=i; j<n; j++)
{
if (hash[s[j]] == 1) break;
len = j - i + 1
maxlen = max(len, maxlen);
hash[s[j]] = 1;


i j
s = c a d b z a b c d
maxlen = 0
for(i = 0 ; i < n ; i++)
{
hash[256] = {0}?
for(j = i ; j < n ; j++)
{
if (hash[s[j]] == 1) break;
len = j - i + 1
maxlen = max(len, maxlen);
hash[s[j]] = 1 ;
}
}
c
ca
cad
cadb
cadbz
cadbza
cdbzab

c
ca
cad
cadb
cadbz
cadbza
cdzab

```
for( i = 0 ; i < n ; i++)
{
    hash[256] = {0}
    for( j = i ; j < n ; j++)
    {
        if (hash[s[j]] == 1) break;
        len = j - i + 1
        maxlen = max(len, maxlen);
        hash[s[j]] = 1 ;
    }
}
print (maxlen)
```


i j
s = c a d b z a b c d
TC → N×N = N²
SC → O(256)
maxlen = 0
for(i = 0 ; i < n ; i++)
hash[256] = {0}?
for(j = i ; j < n ; j++)
if (hash[s[j]] == 1) break;
len = j - i + 1
maxlen = max(len, maxlen);
hash[s[j]] = 1 ;
c
ca
cad
cadb
cadbz
cadbza
cdbzab


Longest substring without repeating characters
0 1 2 3 4 5 6 7 8
s = c a d b z a b c d
i j
a, 1
c, 0
char, index
maxLen = 4 X 2

Longest substring without repeating characters

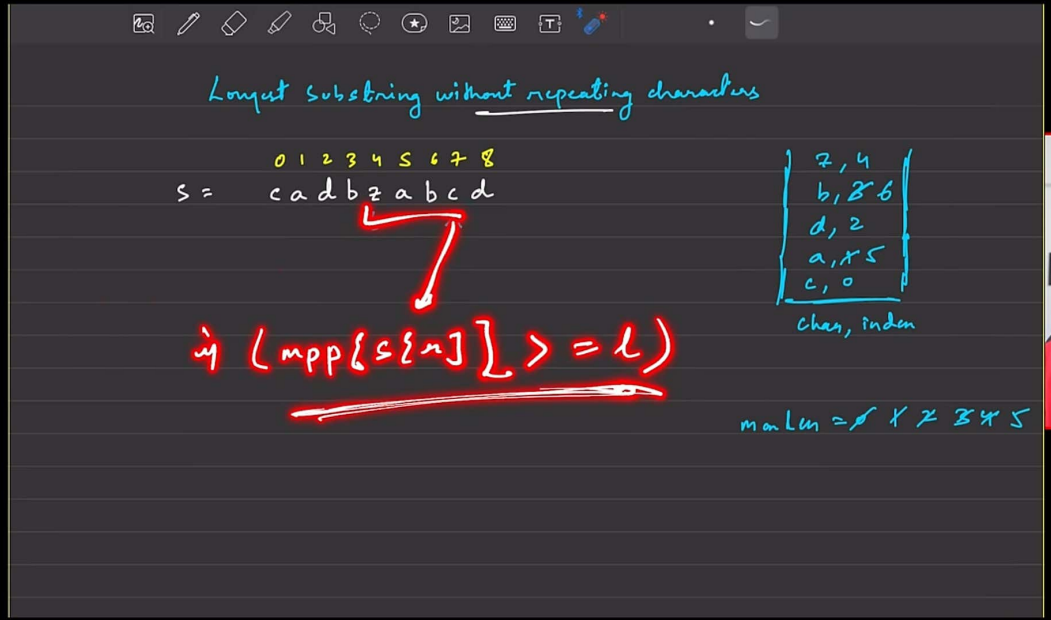
	0	1	2	3	4	5	6	7	8
s =	c	a	d	b	z	a	b	c	d

l r

char, index
z, 4
b, 3
d, 2
a, 1
c, 0

l = mpp['a'] + 1

maxLen = 0 1 2 3 4 5


Longest substring without repeating characters
0 1 2 3 4 5 6 7 8
s = c a d b z a b c d
if (mpp[s[r]] >= l)
z, 4
b, 3
d, 2
a, 1
c, 0
char, index
maxLen = 0 1 2 3 4 5

Longest substring without repeating characters

	0	1	2	3	4	5	6	7	8
s =	c	a	d	b	z	a	b	c	d
		l				r			

char	index
z	4
b	6
d	2
a	5
c	7

maxLen = 5

```
func ( string s)
{
    hash[256] ⟶ [-1]
        n = s.size()
    l = 0   r = 0   maxLen = 0

    while (r < n)
    {
        if ( hash[s[r]] != -
```

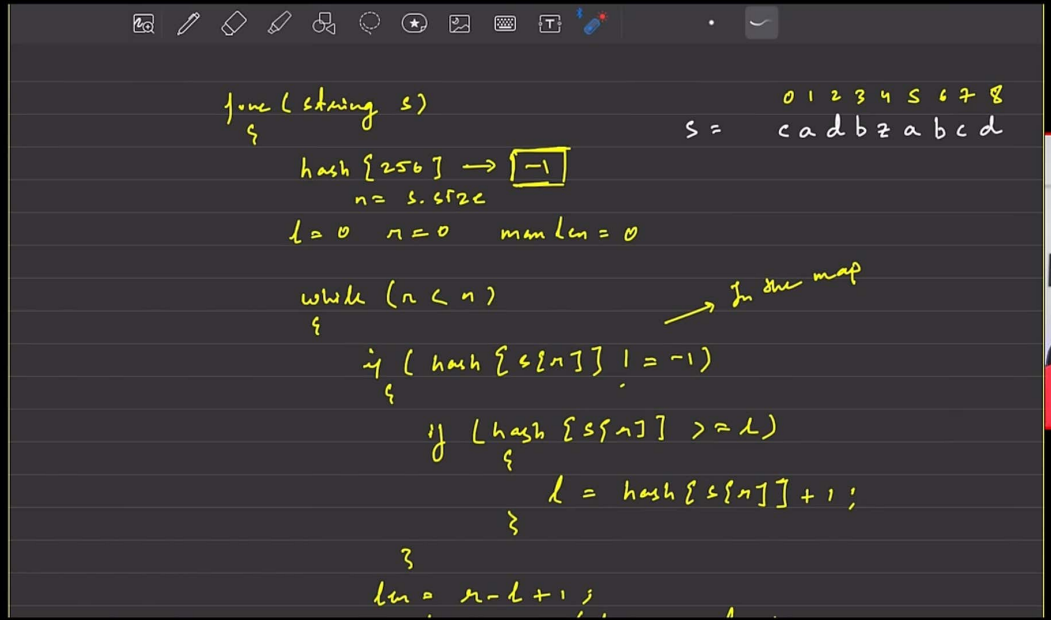
```
        0 1 2 3 4 5 6 7 8
s =     c a d b z a b c d
```

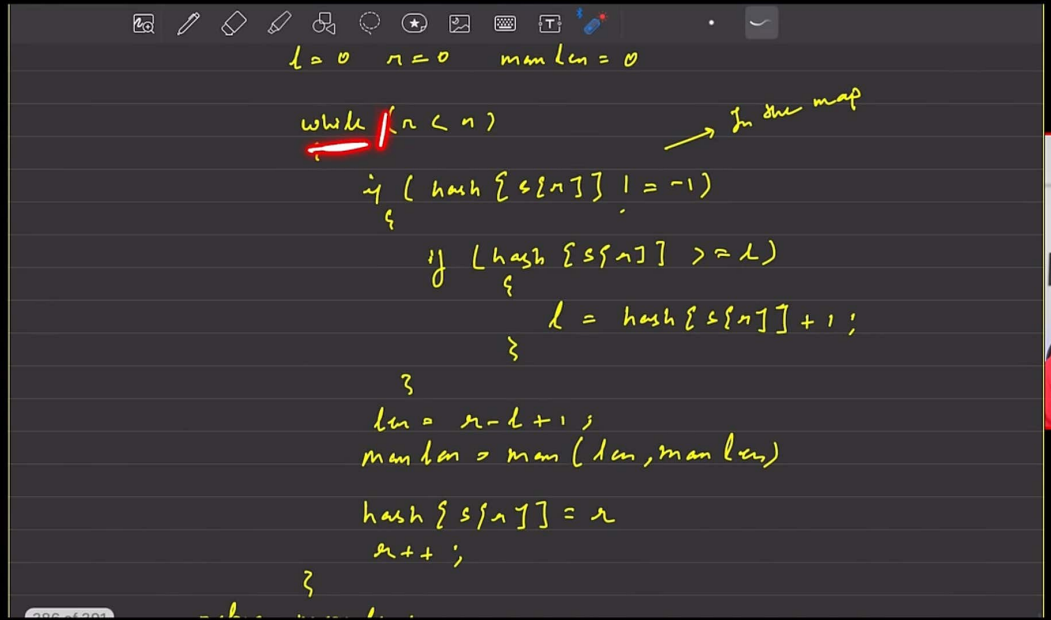
```
while (r < n)                          → In the map
{
    if ( hash[s[r]] != -1)
    {
        if (hash[s[r]] >= l)
        {
            l = hash[s[r]] + 1;
        }
    }
    len = r - l + 1;
    maxlen = max(len, maxlen)
```

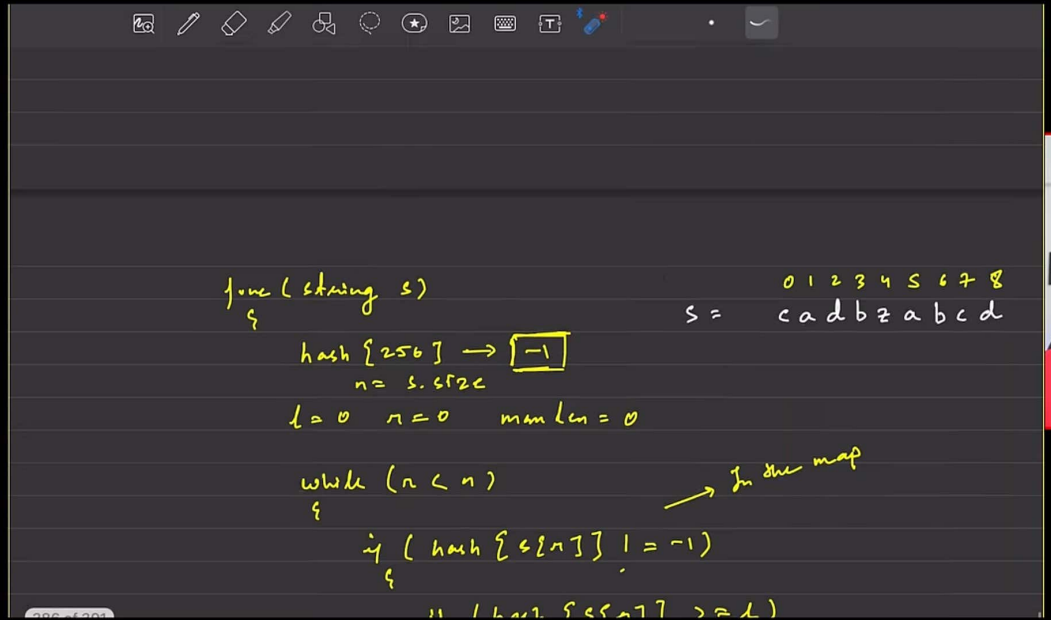
```
l = 0   r = 0   maxLen = 0

while (r < n)                          → In the map
{
    if ( hash[s[r]] != -1)
    {
        if (hash[s[r]] >= l)
        {
            l = hash[s[r]] + 1;
        }
    }
    len = r - l + 1;
    maxLen = max(len, maxLen)

    hash[s[r]] = r
    r++;
}
```


func (string s)
{
hash [256] → [-1]
n = s.size()
l = 0 r = 0 maxLen = 0
while (r < n)
{
if (hash [s[r]] != -1)
{
if (hash [s[r]] >= l)
{
l = hash [s[r]] + 1 ;
}
}
len = r - l + 1 ;
0 1 2 3 4 5 6 7 8
s = c a d b z a b c d
→ In the map


l = 0 r = 0 maxLen = 0
while (r < n)
in the map
if (hash[s[r]] != -1)
if (hash[s[r]] >= l)
l = hash[s[r]] + 1;
len = r - l + 1;
maxLen = max(len, maxLen)
hash[s[r]] = r
r++;


func (string s)
{
hash [256] → -1
n = s.size
l = 0 r = 0 mxlen = 0
while (r < n)
{
if (hash [s[r]] != -1)
0 1 2 3 4 5 6 7 8
s = c a d b z a b c d
→ In the map

```
l = 0   r = 0   maxLen = 0

while (r < n)                          → In the map
{
    if (hash[s[r]] != -1)
    {
        if (hash[s[r]] >= l)
        {
            l = hash[s[r]] + 1;
        }
    }
    len = r - l + 1;
    maxLen = max(len, maxLen)

    hash[s[r]] = r
    r++;
}
```

View key concept

TC → O(N)
SC → O(256)

func(string s)
{
hash[256] → [-1]
n = s.size()
l = 0 r = 0 maxLen = 0

0 1 2 3 4 5 6 7 8
s = c a d b z a b c d

while (r < n)
{
→ In the map