# Subarrays with K different Integers

arr = [1 2 1 3 4]  K = 3

# Subarrays with K different Integers

arr = [1 2 1 3 4]  K = 3

1, 2

# Subarrays with K different Integers

arr = [1 2 1 3 4]  K = 3

# Subarrays with K different Integers

arr = [1 2 1 3 4]  K = 3

for ( i = 0 → n)
{

mpp

for ( j = i → n )

```
for (i = 0 → n) {
    mpp
    for (j = i → n) {
        mpp[nums[j]]++;
        if (mpp.size() == k) cnt = cnt + 1
        else if (mpp.size() > k) break;
```

arr = [1 2 1 3 4]  K = 3

```
cnt = 0
for ( i = 0 → n)
{
    mpp
    for ( j = i → n )
    {
        mpp[nums[j]]++;
        if (mpp.size() == k) cnt = cnt + 1
        else if (mpp.size() > k) break;
    }
}
return cnt;
```

arr = [1 2 1 3 4]  K = 3

```
cnt=0                                   TC → O(N^2)
for( i = 0 → n)                         SC → O(N)
{
    mpp
    for( j = i → n )
    {
        mpp[nums[j]]++;

        if (mpp.size() == k) cnt = cnt + 1

        else if (mpp.size() > k) break;
    }
}
return cnt;
```

# Subarrays with K different Integers

arr = [2 1 1 1 3 4 3 2]  K = 3

$N^2 \rightarrow N$.

# Subarrays with K different Integers

arr = [2 1 1 1 3 4 3 2]  k = 3

l  r

3 → 1
1 → 3
2 → 1

(num, freq)

# Subarrays with K different Integers

arr = [2 1 1 1 3 4 3 2]  k = 3

l  r

cnt = 1

4 → 1
3 → 1
1 → 3

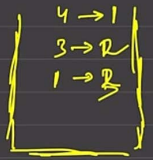
(num, freq)

# Subarrays with K different Integers

arr = [2 1 1 1 3 4 3 2]  k = 3

l  r

cnt = 1 + 2 + 3




(num, freq)

# Subarrays with K different Integers

arr = [2 1 1 1 3 4 3 2]  k = 3

l = 1, r = 5

4 → 1
3 → 2
1 → 3

(num, freq)

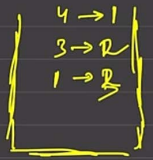
cnt = 1 + 2 = 3

# Subarrays with K different Integers

arr = [2 1 1 1 3 4 3 2]  k = 3

l = 1, r = 5

cnt = 1 + 2 = 3




(num, freq)

# Subarrays with K different Integers

no. of subarrays where different Integers <= K

arr = [2 1 1 1 3 4 3 2]  k = 3

Subarrays with K different Integers

no. of subarrays where different Integers <= K

arr = [2 1 1 1 3 4 3 2]  K = 3

l r

cnt = 1


(num, freq)

Subarrays with K different Integers

no. of subarrays where different Integers <= K

arr = [2 1 1 1 3 4 3 2]  K = 3

l  r

cnt = 2 + 3

| 1, 2 |
| --- |
| 2, 1 |

(num, freq)

Subarrays with K different Integers

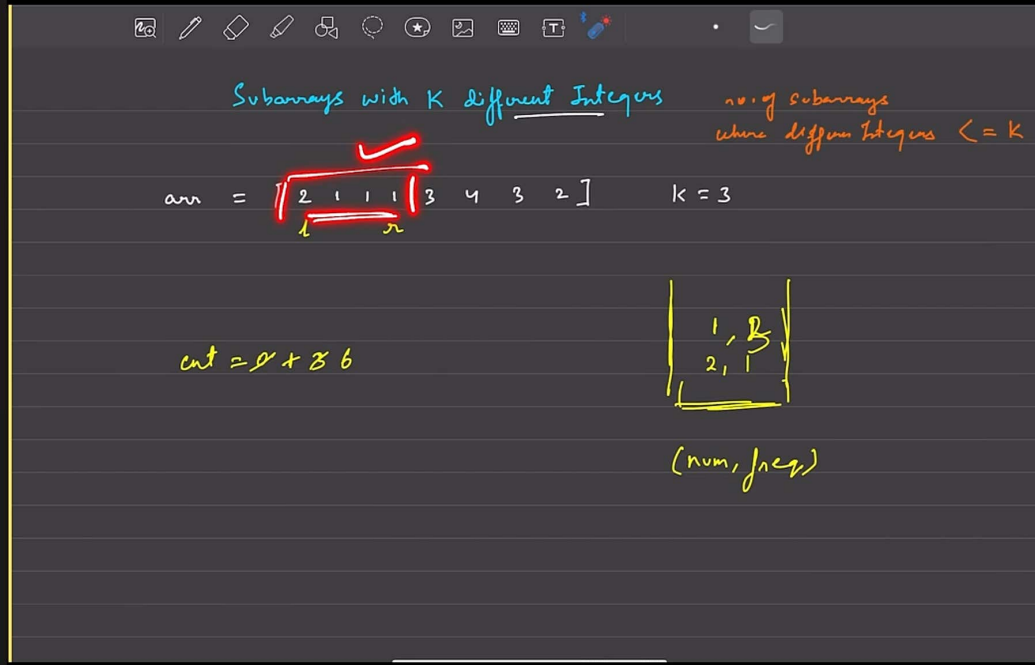
no. of subarrays where different Integers <= K

arr = [2 1 1 1 3 4 3 2]  k = 3

cnt = x + 3

| 1, 2 |
| --- |
| 2, 1 |

(num, freq)


Subarrays with K different Integers
no. of subarrays where different Integers <= K
arr = [2 1 1 1 3 4 3 2]
k = 3
l
r
cnt = x + 3 6
1, 3
2, 1
(num, freq)

Subarrays with K different Integers

no. of subarrays where diffrent Integers <= K

arr = [2 1 1 1 3 4 3 2]  k = 3

l  r

cnt = x + 3 + 10


(num, freq)

# Subarrays with K different Integers

no. of subarrays where different Integers <= K

arr = [1 1 1 3 4 3 2]  k = 3

cnt = x + 3 = 10

| 3, 1 |
| --- |
| 1, 3 |
| 2, 1 |

(num, freq)

Subarrays with K different Integers

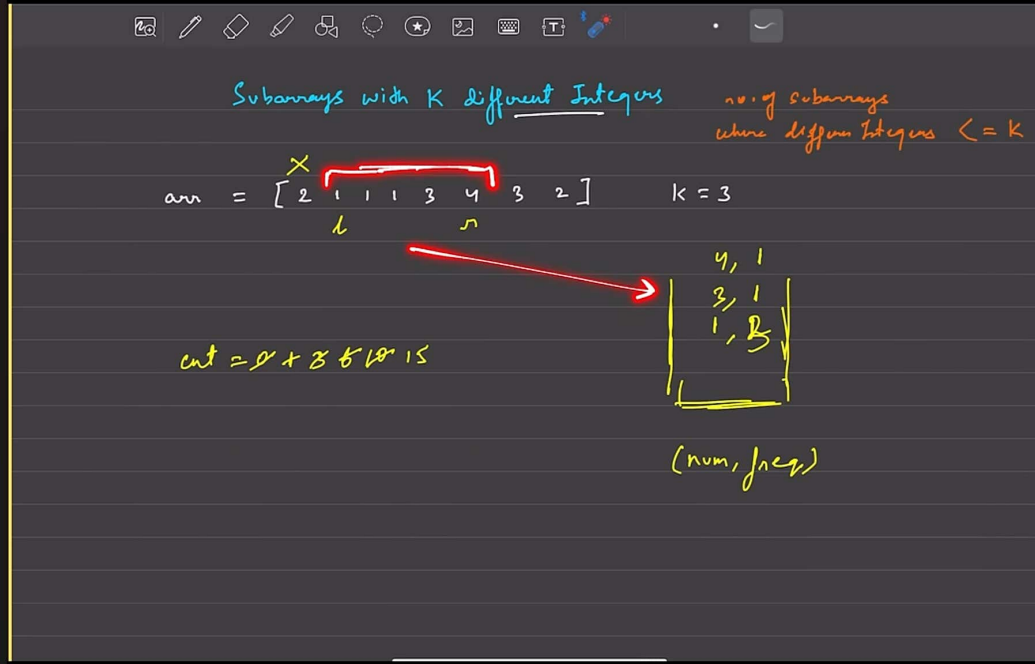
no. of subarrays where different Integers <= K

arr = [2 1 1 1 3 4 3 2]  K = 3

cnt = 4 + 3 + 1 = 8

4, 1
3, 1
1, 3
2, 1

(num, freq)


Subarrays with K different Integers
no. of subarrays where different Integers <= K
arr = [2 1 1 1 3 4 3 2]
k = 3
cnt = x + 3 for loop is
(num, freq)

Subarrays with K different Integers

no. of subarrays where different Integers <= K

arr = [2 1 1 1 3 4 3 2]  K = 3

l  r

cnt = 4 + 3 + 1 + 1 + 1 = 20

4, 1
3, 1
1, 3

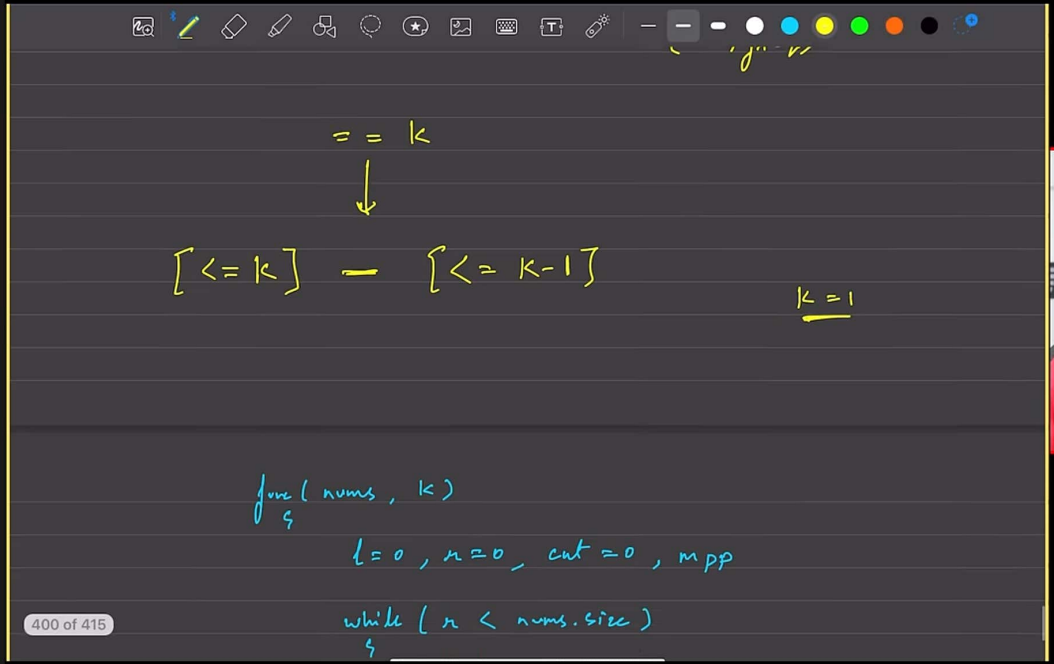
(num, freq)

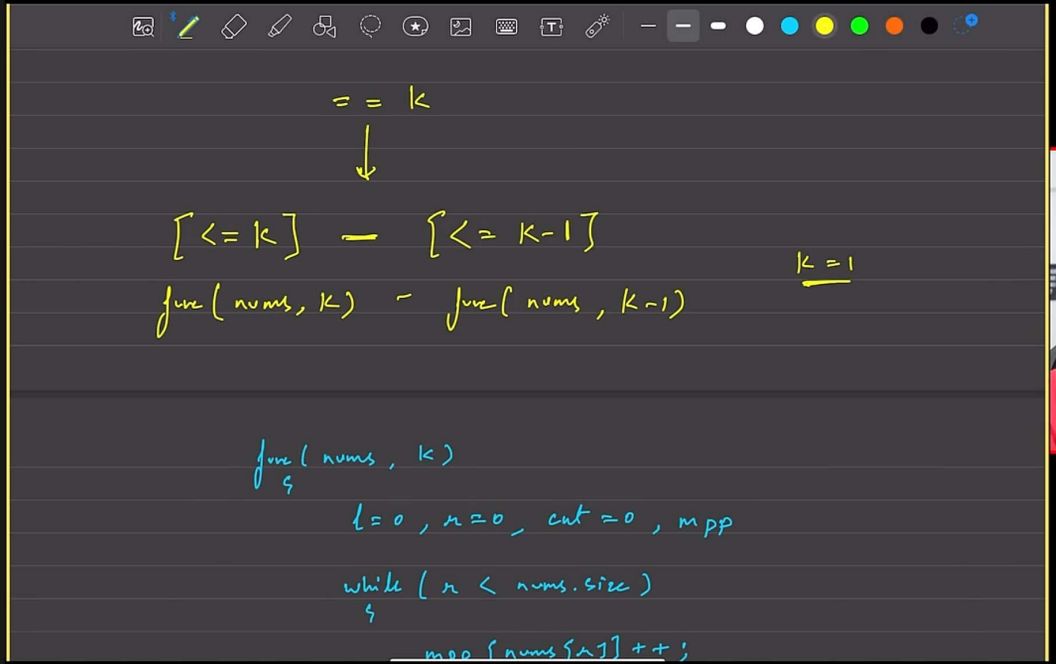
```
fun( nums , k )
{
    l=0 , r=0 , cnt=0 , mpp

    while ( r < n
```

```
mpp[nums[r]]++;

while(mpp.size() <= K)
{
    mpp[nums[l]]--; l=l-1;
```

```
func( nums , k )
{
    l = 0 , r = 0 , cnt = 0 , mpp

    while ( r < nums.size )
    {
        mpp [nums[r]] ++ ;

        while(mpp.size() <= K)
        {
            mpp [nums[l]]-
```

```
                mpp.erase(nums[l]);
                l = l - 1;
            }
            cnt = cnt + (r - l + 1);
            r = r + 1;
        }
        return cnt;
```


== k
[<= k] — [<= k-1]
k = 1
func( nums , k )
{
l = 0 , r = 0 , cnt = 0 , mpp
while ( r < nums.size )
{


== k
[<= k] — [<= k-1]
func(nums, K) - func(nums, k-1)
k = 1
func(nums, k)
{
l = 0, r = 0, cnt = 0, mpp
while (r < nums.size)
{
mpp[nums[r]]++;

```
fun( nums , k )
{
    l = 0 , r = 0 , cnt = 0 , mpp

    while ( r < nums.size )          N
    {
        mpp [nums[r]] ++ ;

        while(mpp.size() <= K) →     N
        {
            mpp[nums[l]] -- ;

            if [mpp[nums[l]] == 0)
                mpp.erase(nums[l]) ;

            l = l - 1 ;
        }
```

```
fun( nums , k )
{
    l = 0 , r = 0 , cnt = 0 , mpp

    while ( r < nums.size )                    N
    {
        mpp [nums[r]] ++ ;

        while(mpp.size() <= K) →               N
        {
            mpp [nums[l]] -- ;

            if [mpp [nums[l]] == 0)
                mpp.erase(nums[l]) ;

            l = l - 1 ;
        }
```

TC → O(2N)
SC → O(N)

func(nums, K) - func(nums, K-1)

func(nums, k)
{
l=0, r=0, cnt=0, mpp

2 X
TC → O(2N)
SC → O(N)

while(r < nums.size) N
{
mpp[nums[r]]++;

while(mpp.size() <= K) → N
{
mpp[nums[l]]--;

if[mpp[nums[l]] == 0)