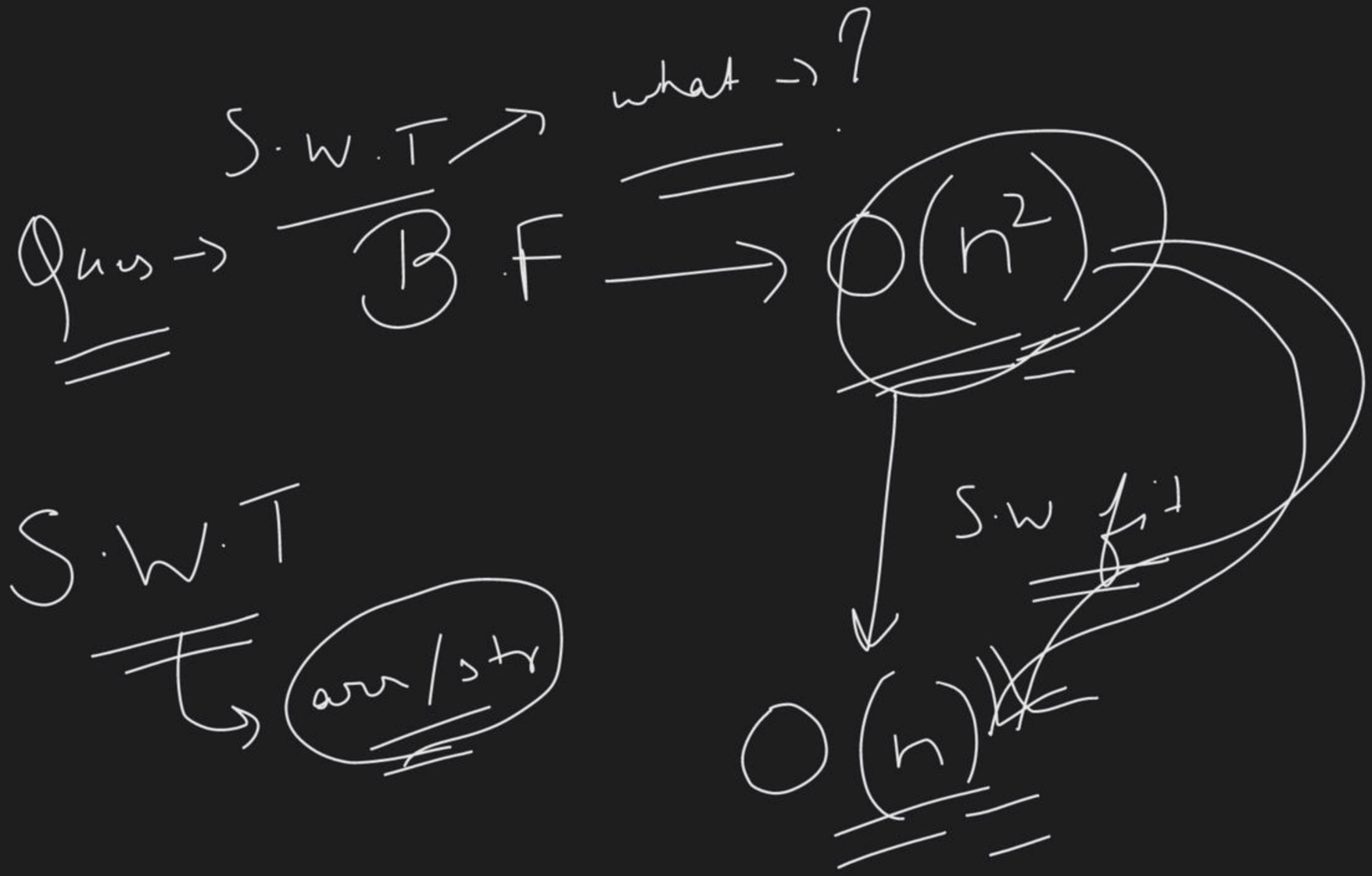
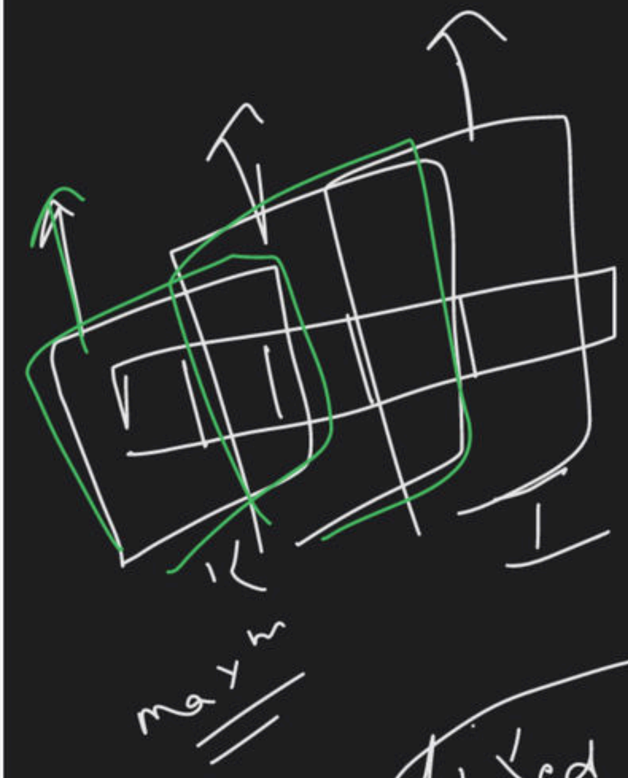
Sliding Window Problems

Special class

Love Babbar • Mar 29, 2022


S.W.T
what →?
Ques →
B.F
O(n²)
S.W.T
arr/str
S.W.T
O(n)



S.W.T

Types

fixed window

size "K"

Not - fixed sized window

2 ptr

find all subarrays whose sum is exactly "K"

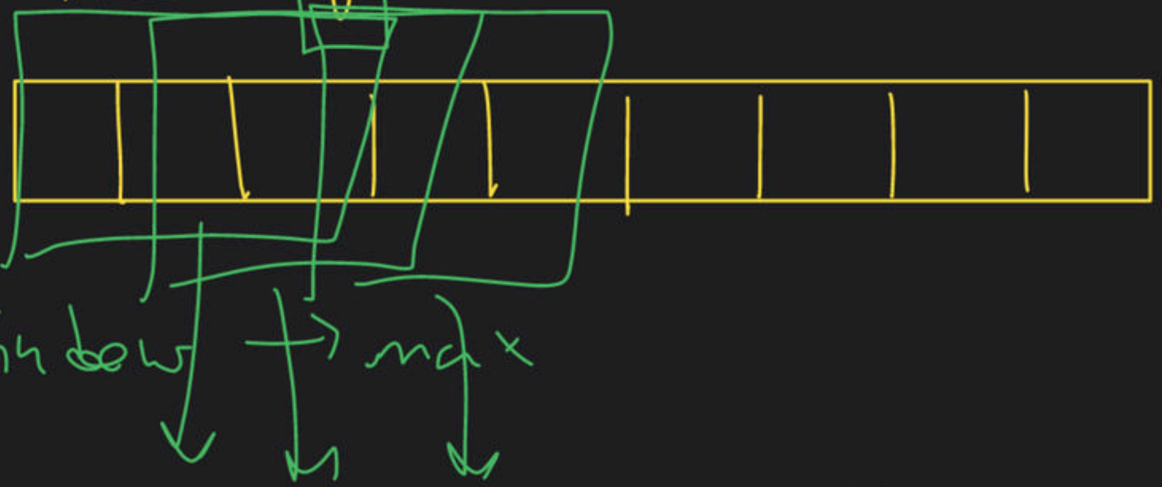
1 → n

Ex → find all subarrays based on xyz criteria

types

1

fixed window



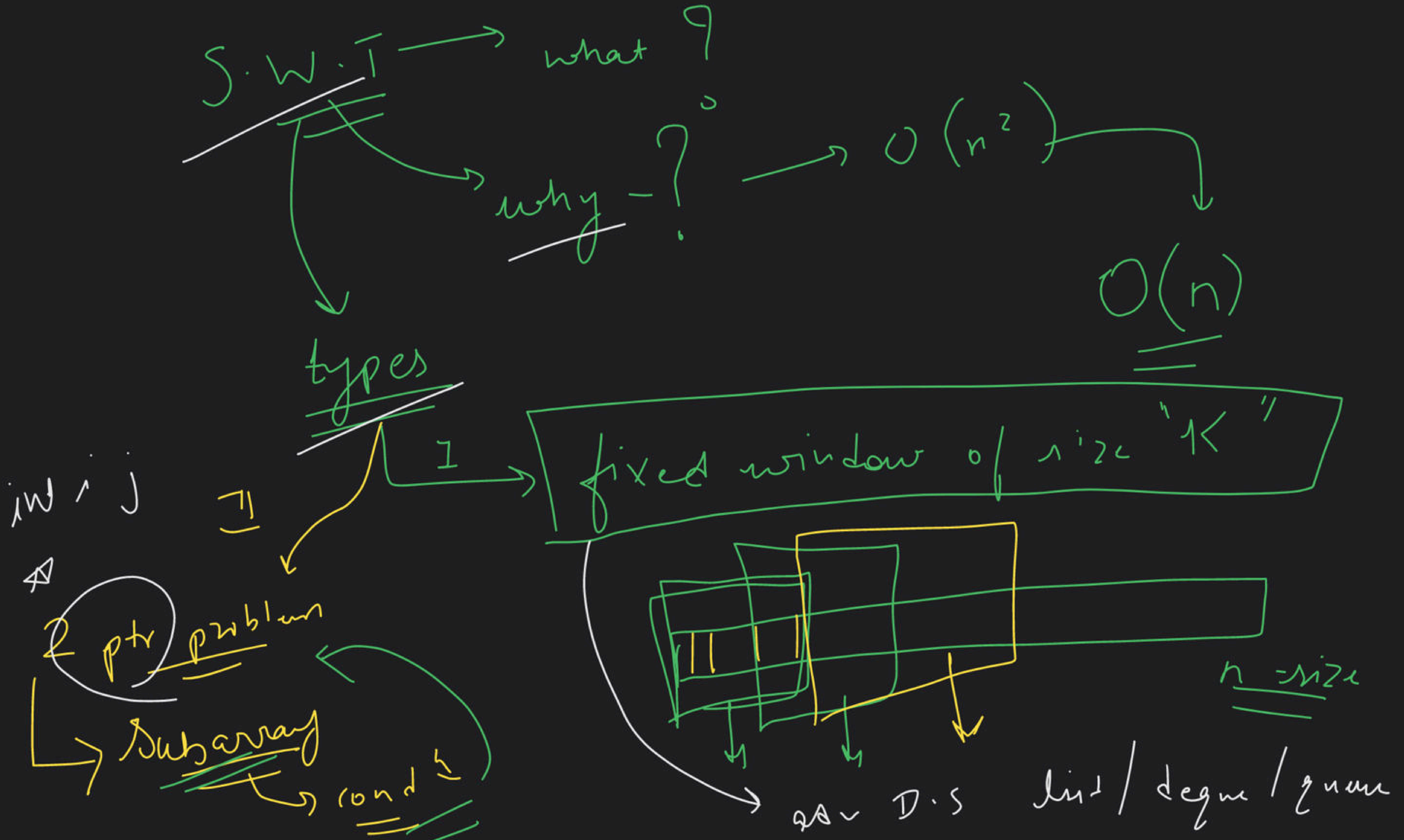
K = 3

K size window → max

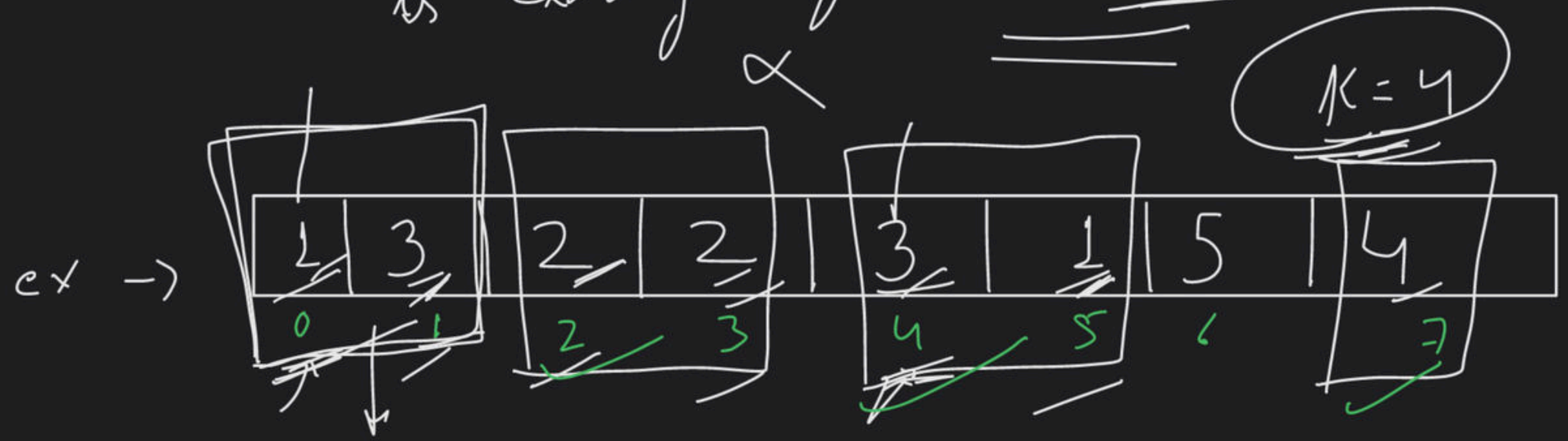
2 ptr problem

find all subarray

basis on criteria


S.W.T
what ?
why - ?
O(n²)
O(n)
types
I
II
fixed window of size "K"
n = size
2 ptr problem
Subarray
cond^n
use D.S
list / deque / queue

Ques :- 1 Find all subarrays whose sum is exactly equal to "K"



0 → 1 ⇒ 4
2 → 3 ⇒ 4
4 → 5 ⇒ 4
7 → 7 ⇒ 4

→ solution

Approach :- #1 Brute force

for( )
for

Random → find all subarrays → $O(n^2)$

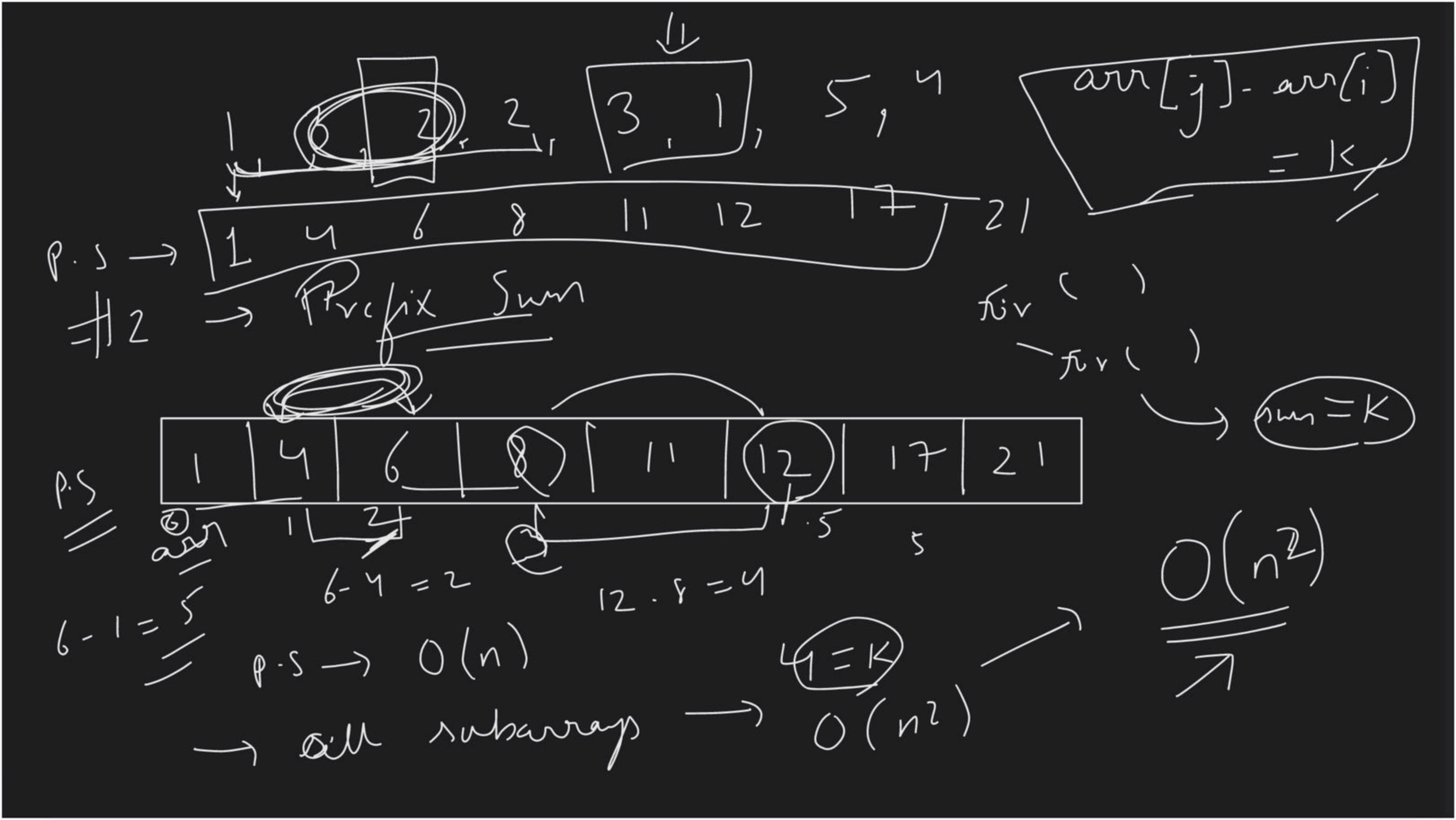
T.C

→ find sum of k subarray → $O(n)$

$O(n^2 * n) \rightarrow O(n^3)$

$O(n^3)$

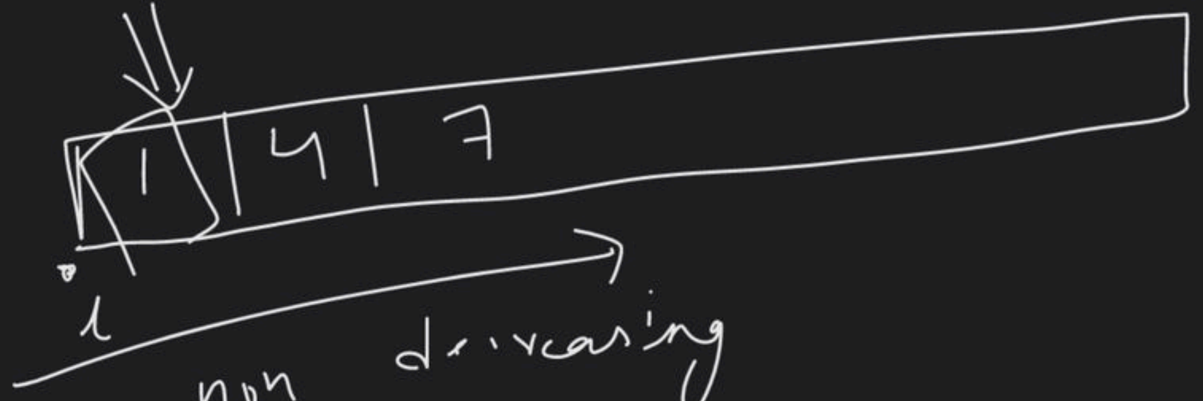
#2


3, 1, 5, 4
arr[j] - arr[i] = k
1 2 2
1 4 6 8 11 12 17 21
P.S →
= 12 → Prefix Sum
fun ( )
fun ( )
sum = K
P.S
1 4 6 8 11 12 17 21
arr
1 2
6 - 4 = 2
12 - 8 = 4
5
6 - 1 = 5
P.S → O(n)
1 = K
→ all subarrays → O(n²)
O(n²)

$O(n^3) \longrightarrow O(n^2)$

Binary Search

#3



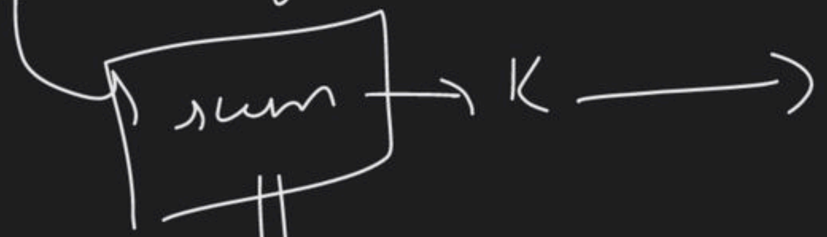
$O(n \log n)$

$arr[j] - arr[i] = K$

$arr[j] - 1 = 4$

$arr[j] = 4 + 1 = 5$

① B.F → all subarray

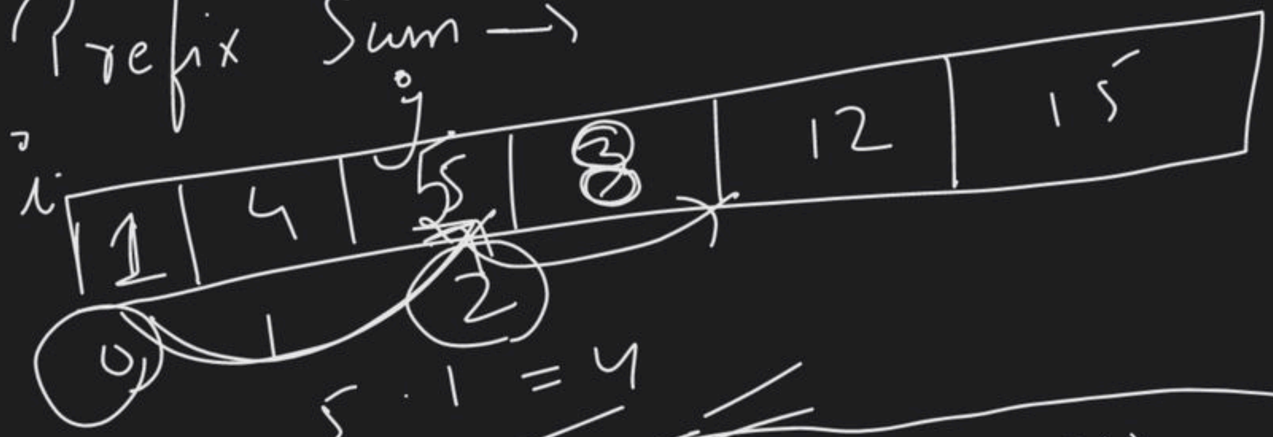


$O(n^3)$

$O(n)$ ~~$O(1)$~~

$O(1)$

② Prefix Sum →

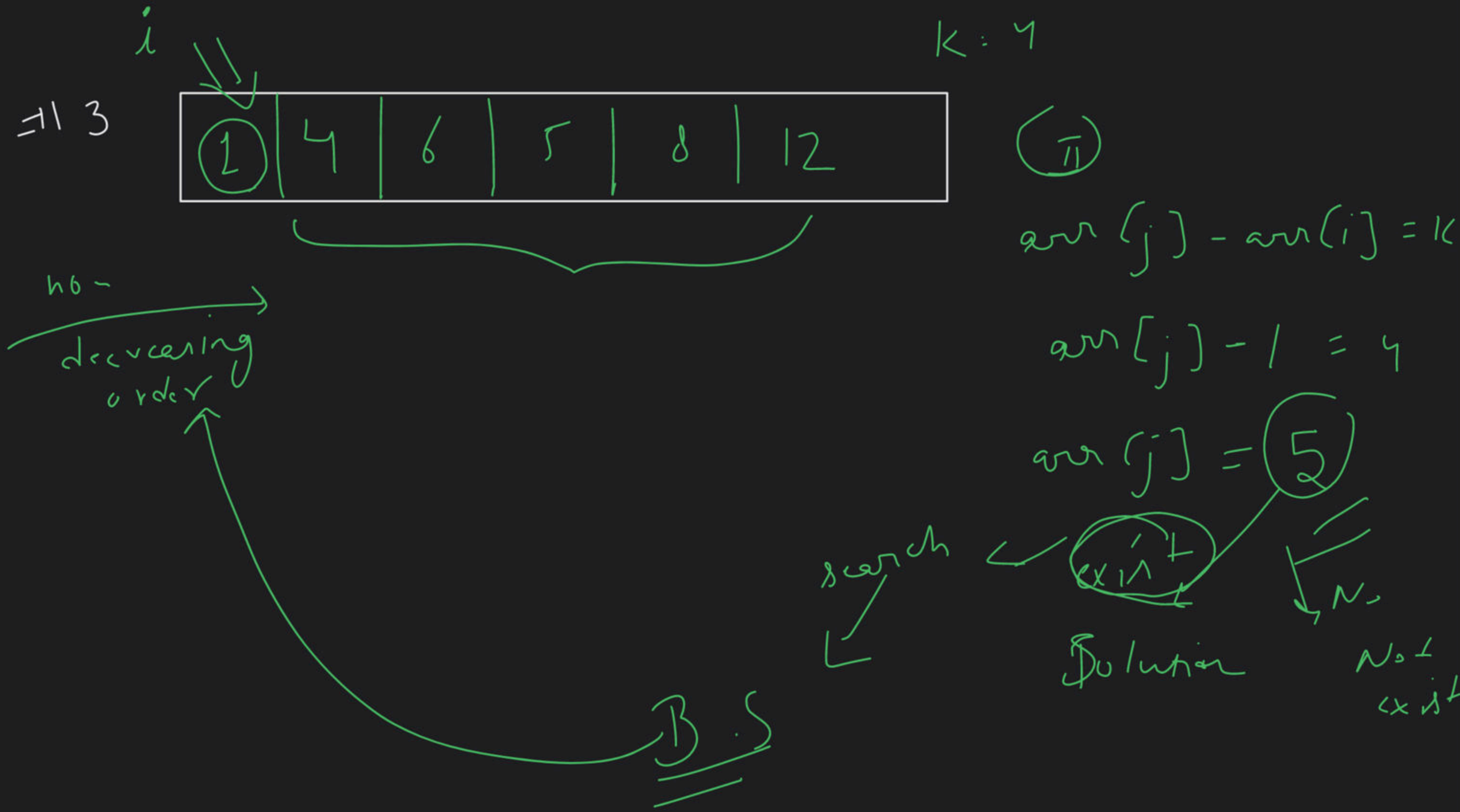


$O(n^2)$

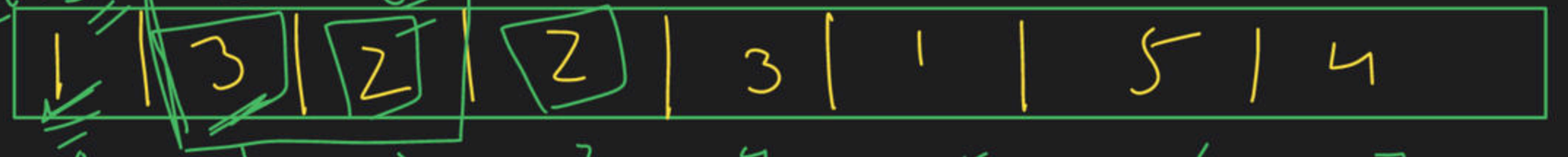
for (i<n)

for (j<n)

if (arr[j] - arr[i] = K)


i
K : 4
n = 3
1
4
6
5
8
12
(II)
arr[j] - arr[i] = K
arr[j] - 1 = 4
arr[j] = 5
no -
decreasing order
search
exist
No
Solution
Not exist
B.S

$O(n^3)$ → $O(n^2)$ → $O(n \log n)$ → S.W.T → $O(n)$



#4

i = 0

j = 0

sum = 0

sum = i → K

K = 4

1 = 4 ✗

4 = 4

6 = 4

5 = 4

4 = 4

i => 0 & j - 1 => 1

2 -> 3

0 -> 1

2

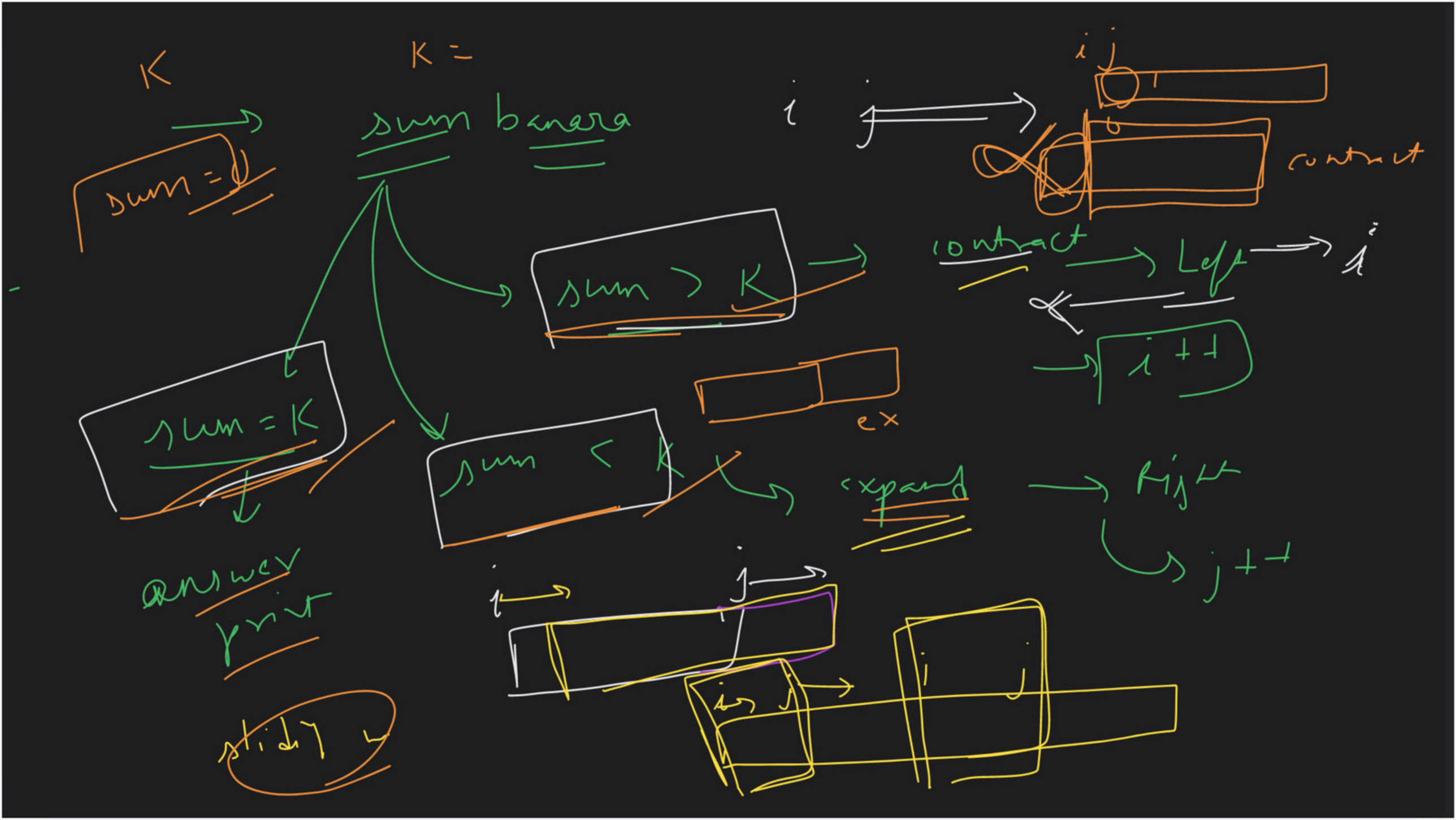
4

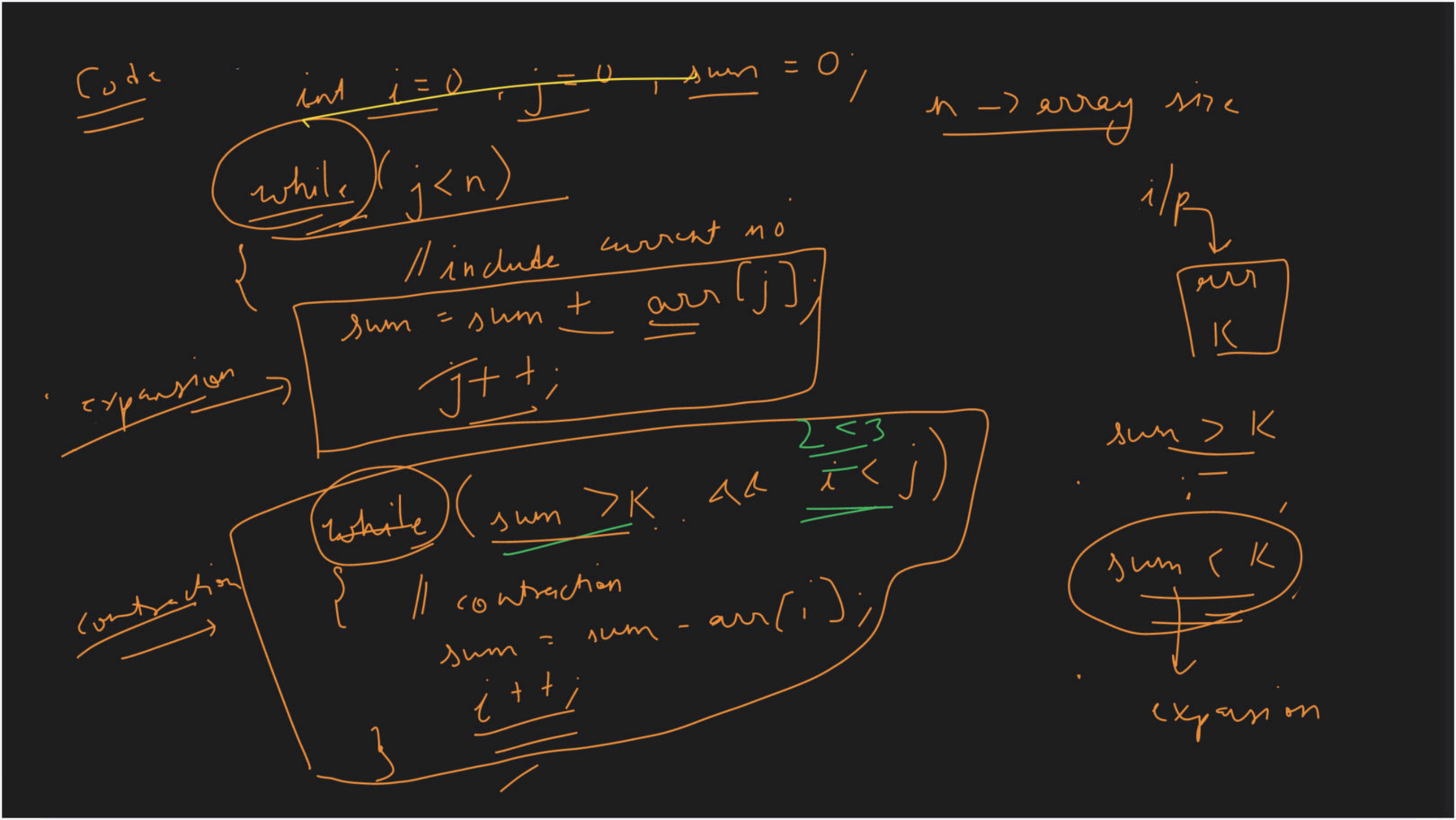
4

5

2

4


K
K =
sum banara
sum = 0
sum > K
sum = K
sum < K
ex
contract
Left
i
i ++
expand
Right
j ++
answer
print
sticky


Code
int i=0, j=0, sum=0;
n -> array size
while (j<n)
{
// include current no
sum = sum + arr[j];
j++;
expansion
i/p
arr
K
sum > K
&&
i<j
2<3
while (sum>K && i<j)
{
// contraction
sum = sum - arr[i];
i++;
}
contraction
sum < K
expansion

if (sum == K)

{ // answer print

cout << i << " " << j-1 << endl;

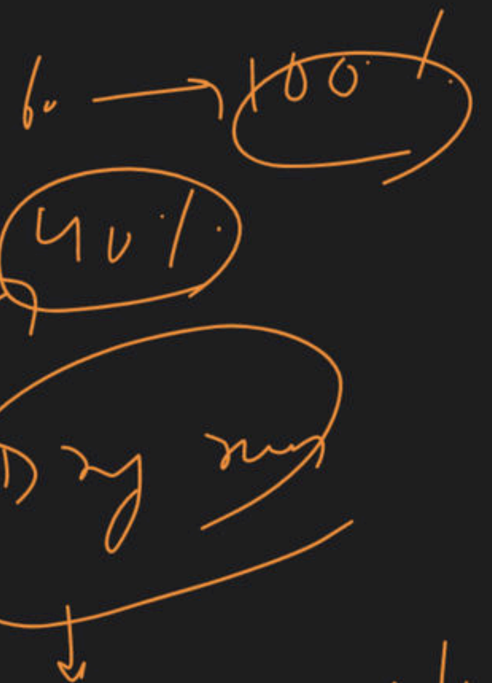
}

<i, j-1>

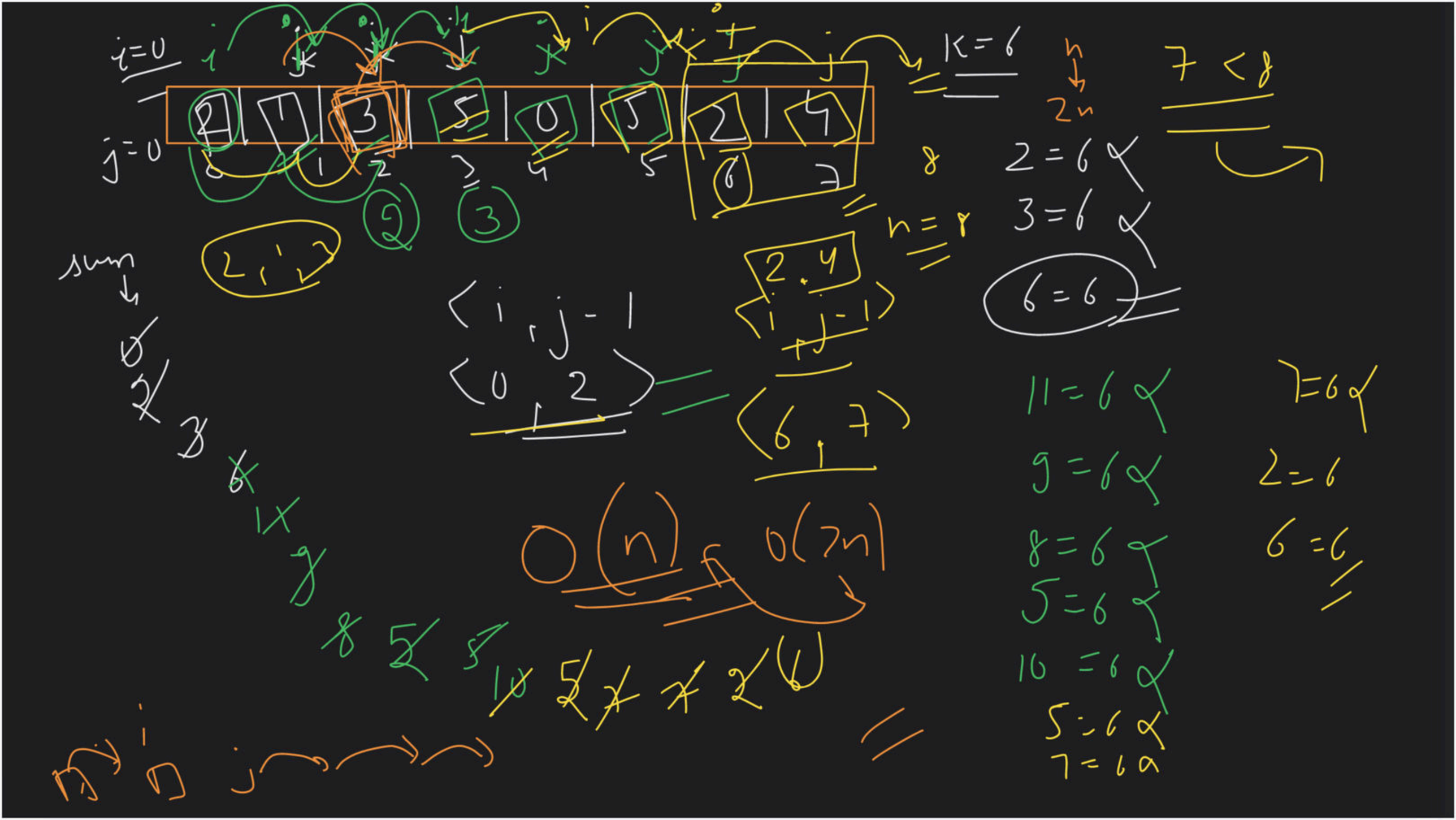
K

S.W.

→ nahi aaya



3-1 example


i=0
j=0
k=6
2 4 3 5 0 5 2 7
0 1 2 3 4 5 6 7
2,4
2,1,2
sum
(i, j-1)
(0, 2)
(6, 7)
O(n)
O(2n)
n=8
2n
7 < 8
2=6 ✗
3=6 ✗
6=6
11=6 ✗
9=6 ✗
8=6 ✗
5=6 ✗
10=6 ✗
5=6 ✗
7=6 ✗
7=6 ✗
2=6
6=6

B.F -> O(n^3)

P.S -> O(n^2)

P.S -> B.S -> O(n log n)

S.W -> O(n)

Interview

Selected

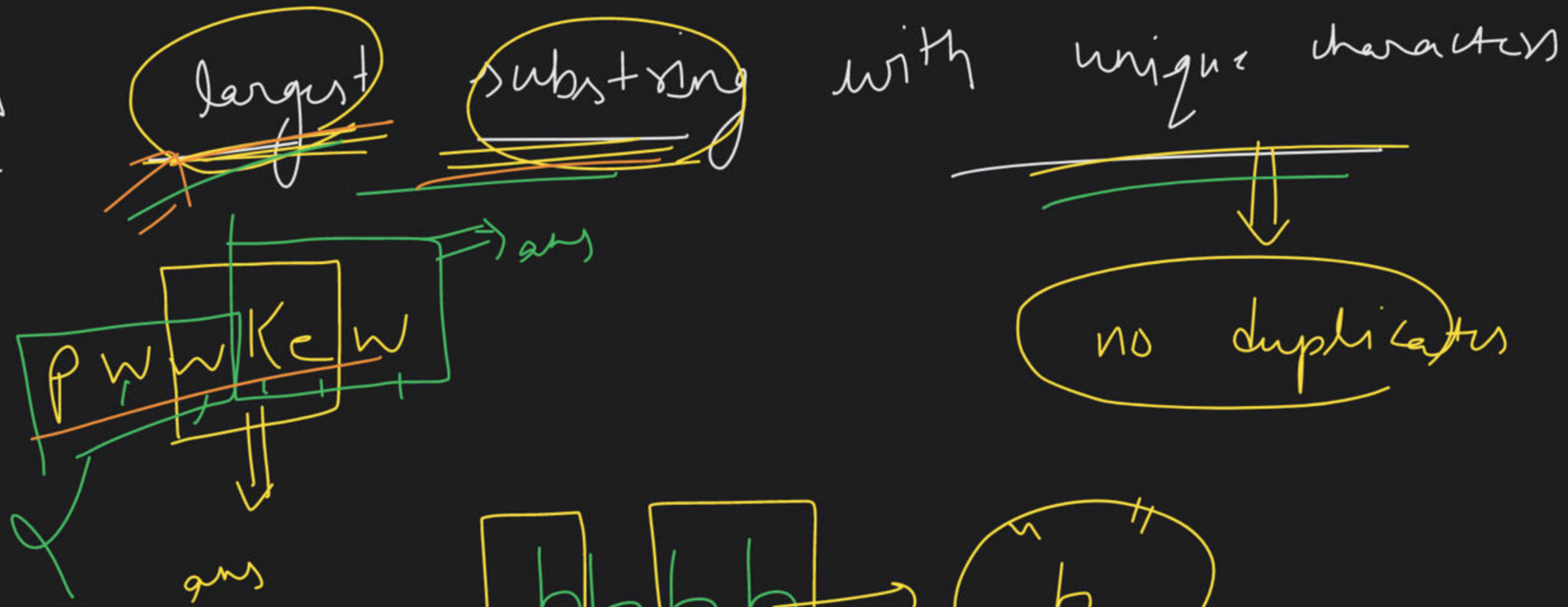
H/w

all subarray -> longest subarray

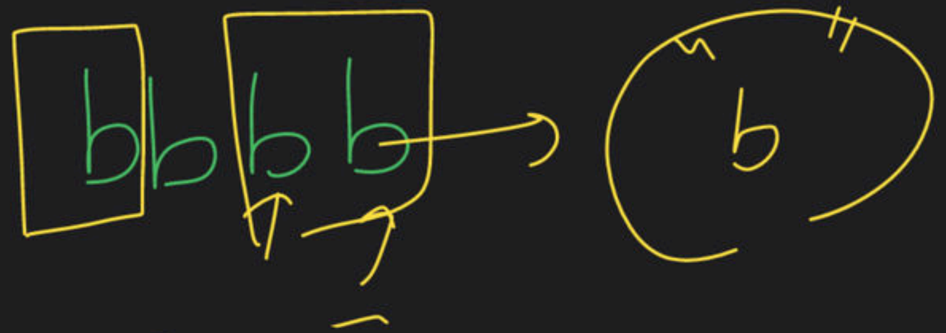
-ve numbers -> smallest subarray -> -ve ? -> H/w

Ques 2

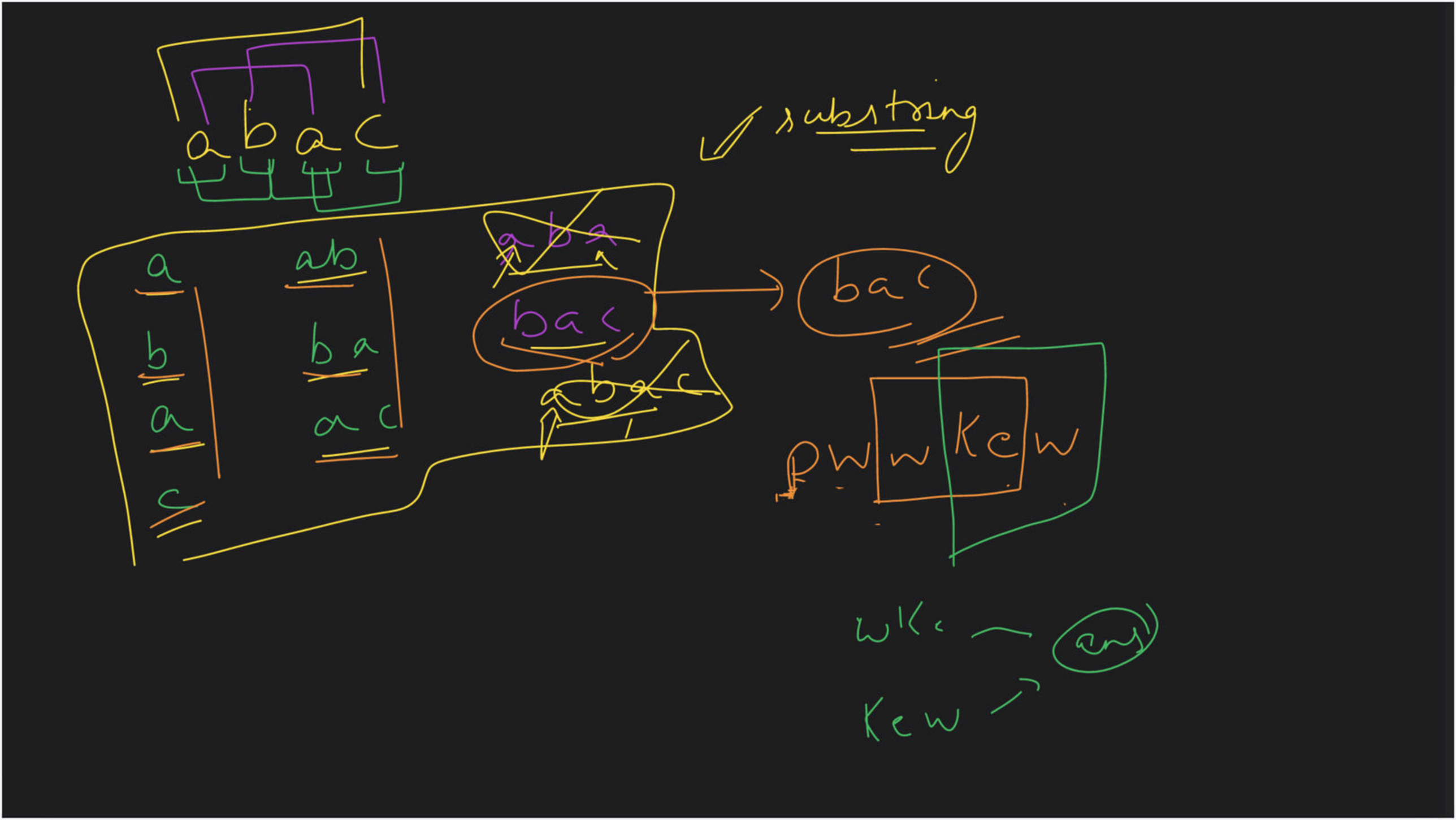
→ find largest substring with unique characters



str =



aabcb


a b a c
substring
a
b
a
c
ab
ba
ac
aba
bac
bac
bac
pwwkew
wke
kew
any

Bhaag J^9

bbbb

bb

→ b

Approach

B.F → $O(n^2 * n) = O(n^3)$ → sab kaata sakta hai

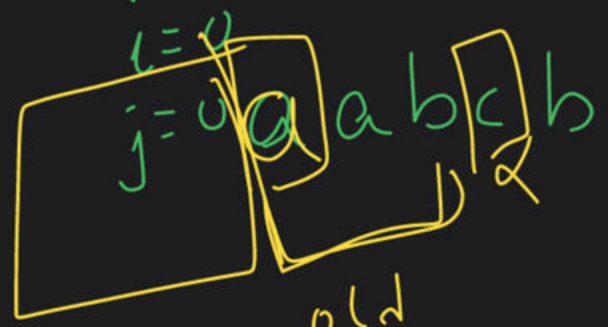
all substring

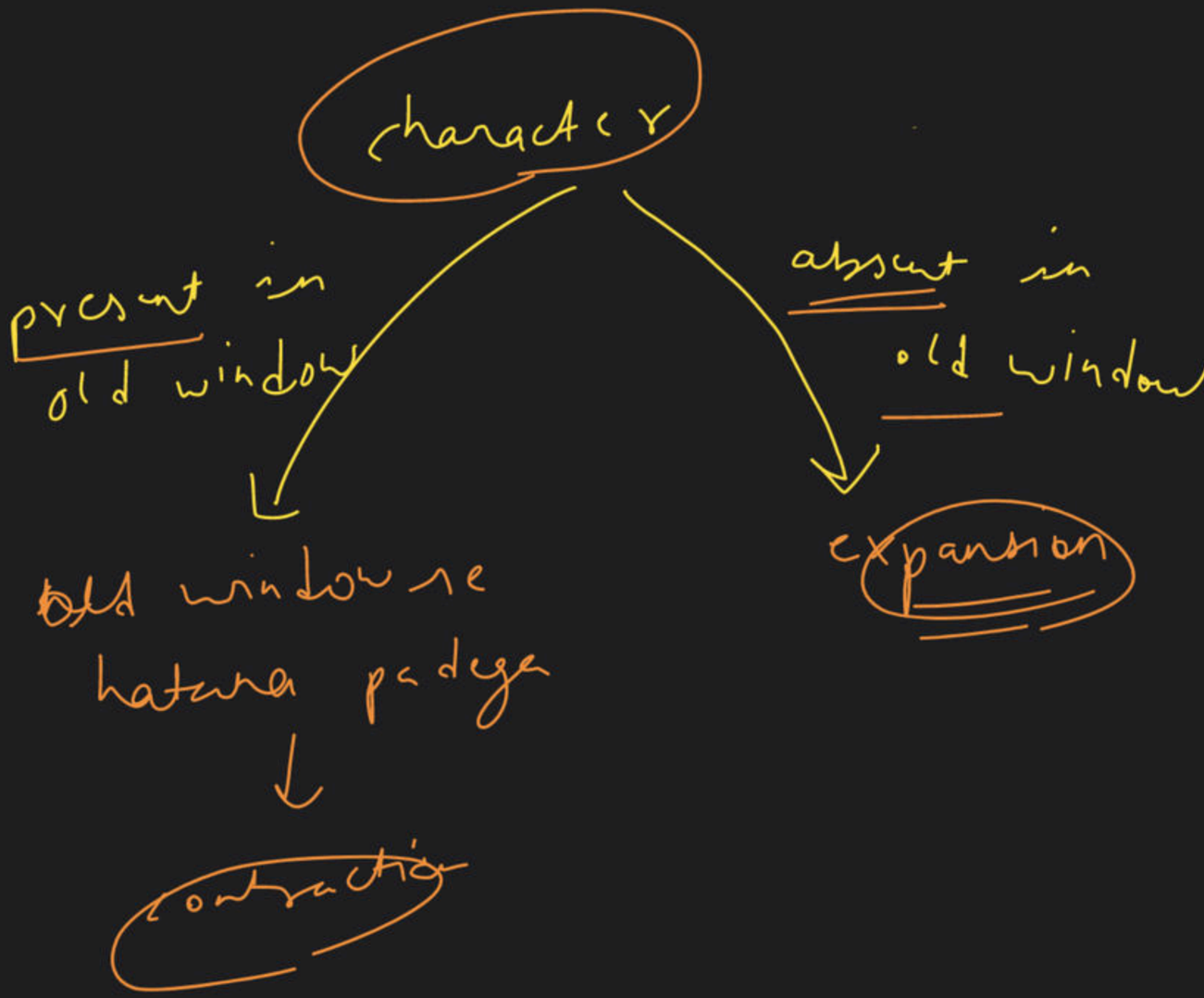
check → duplicate ∆ track layout

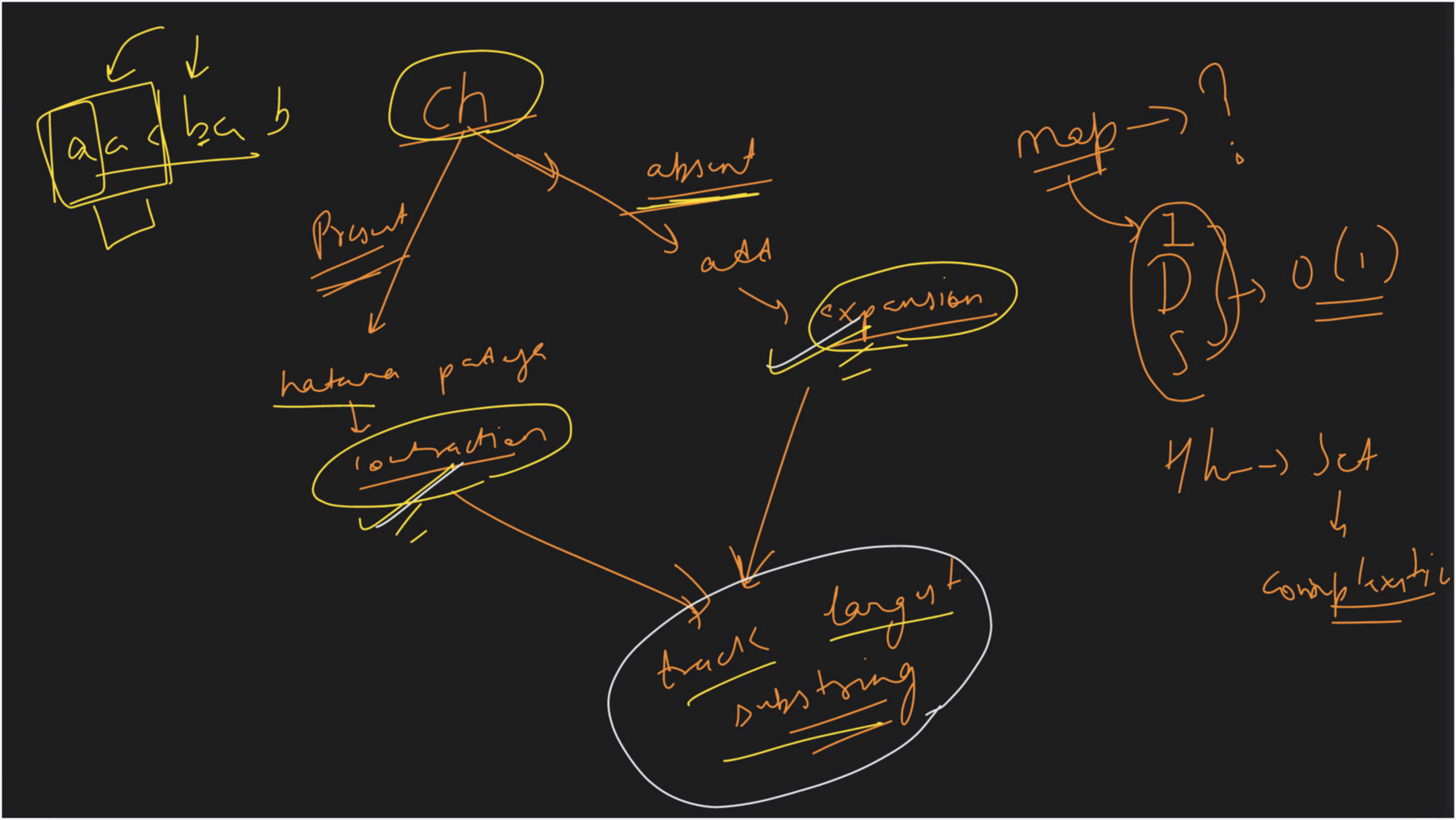
$O(n)$

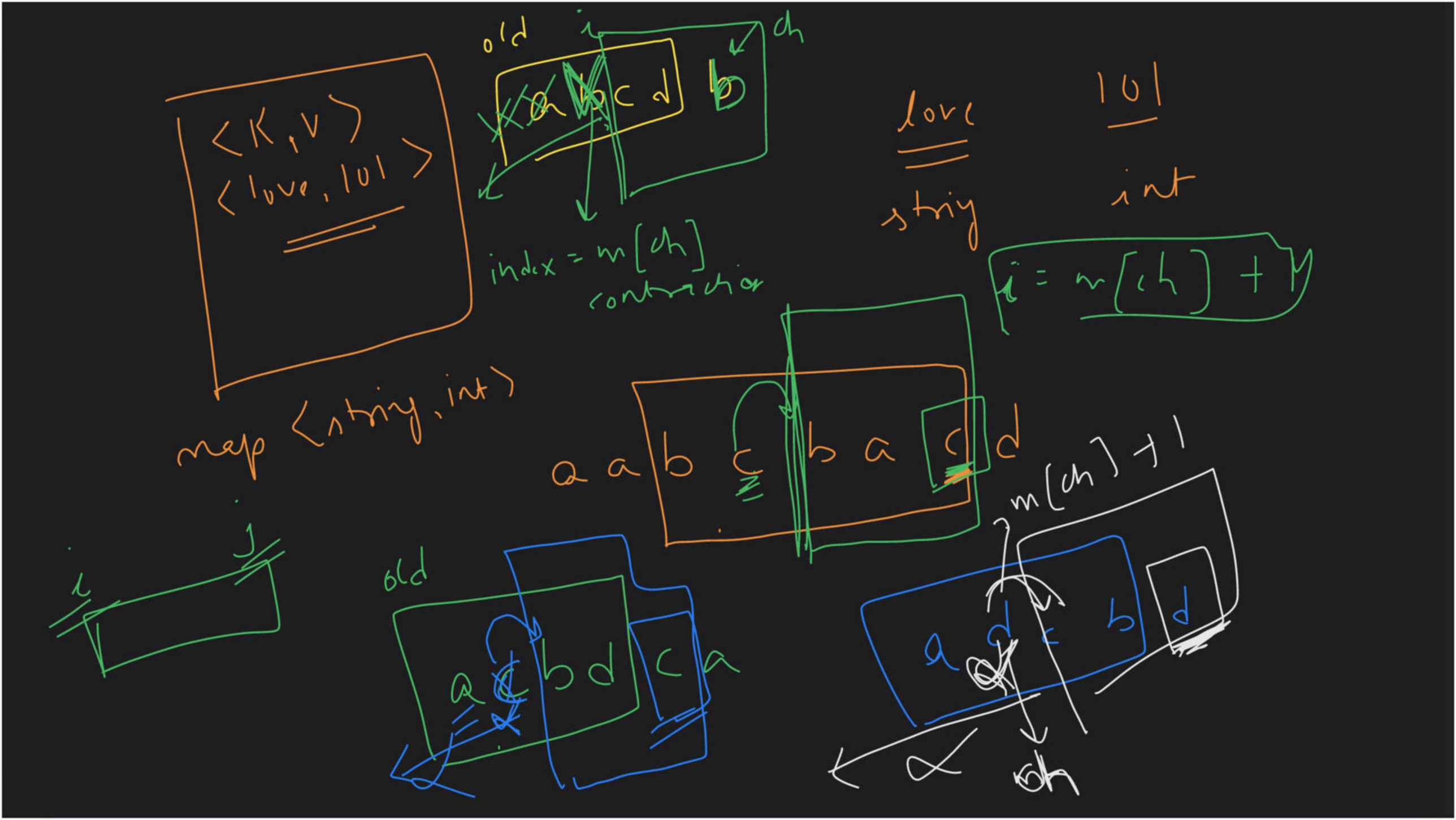
for ( )
{ for ( ) → $O(n^3)$
}
}

#12 Sliding window






ch
present
absent
add
expansion
feature package
contraction
track
longest
substring
map
O(1)
JCA
complexity

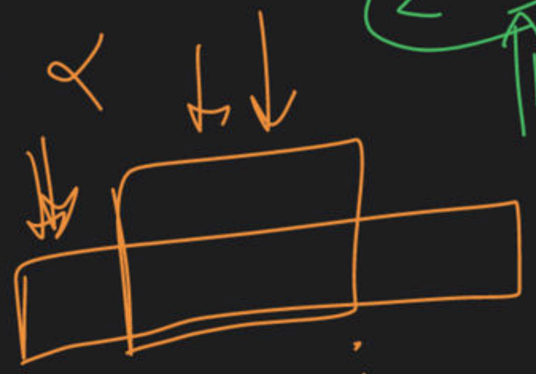

<K, V>
<love, 101>
map <string, int>
old
ch
index = m[ch]
contraction
love
string
101
int
i = m[ch] + 1
m[ch] + 1
old
ch

Code :-

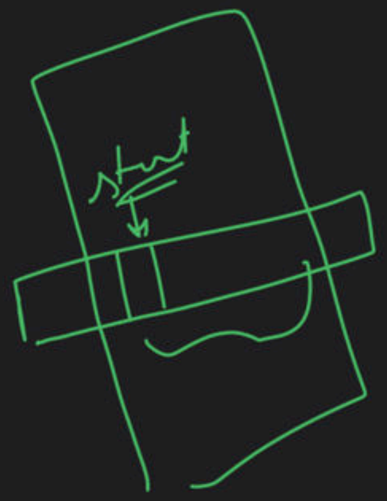
```
int i = 0 , j = 0 ;
unordered_map < char , int > m ;
while ( j < n )
{
    char ch = str[j]
    // present in old window
    while ( m.count(ch) && m[ch] >= 2 )
    {
        i = m[ch] + 1 ;
    }
    currLen = j - i + 1 ;
    maxLen = max( maxLen , currLen ) ;
    m[ch]++ ;
    j++ ;
}
```

int start = -1 ;
int currLen = 0 ;
int maxLen = INT_MIN ;

a b c d c



a -> 2
b -> 1
c -> 2
d -> 1



// include → expansion

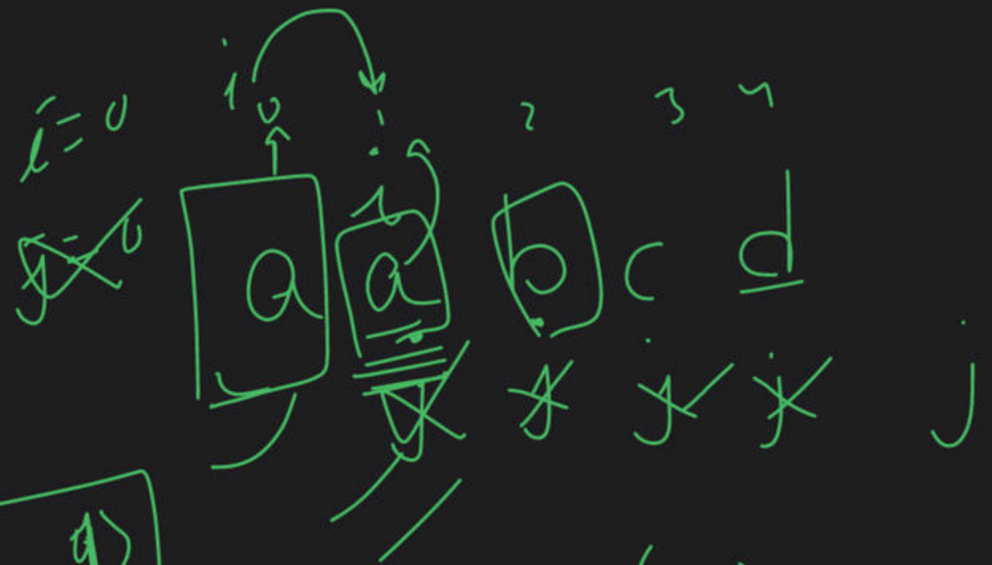
```
m[ch] = j;
currLen++;
j++;
```

// track

```
if (currLen > maxLen)
{
    maxLen = currLen;
    start = i;
}
```

layout substring

```
return str.substring(start, maxLen);
```



O(n)

f·(-)?

Hard → Ehr

i, j → windowLen

maxLen

→ Easy → ?A → f·w

n³ → n² → n log n → n

→ Med → n³ → n