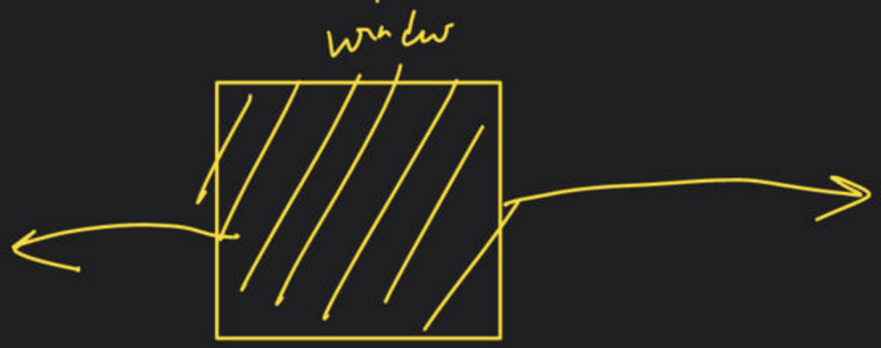
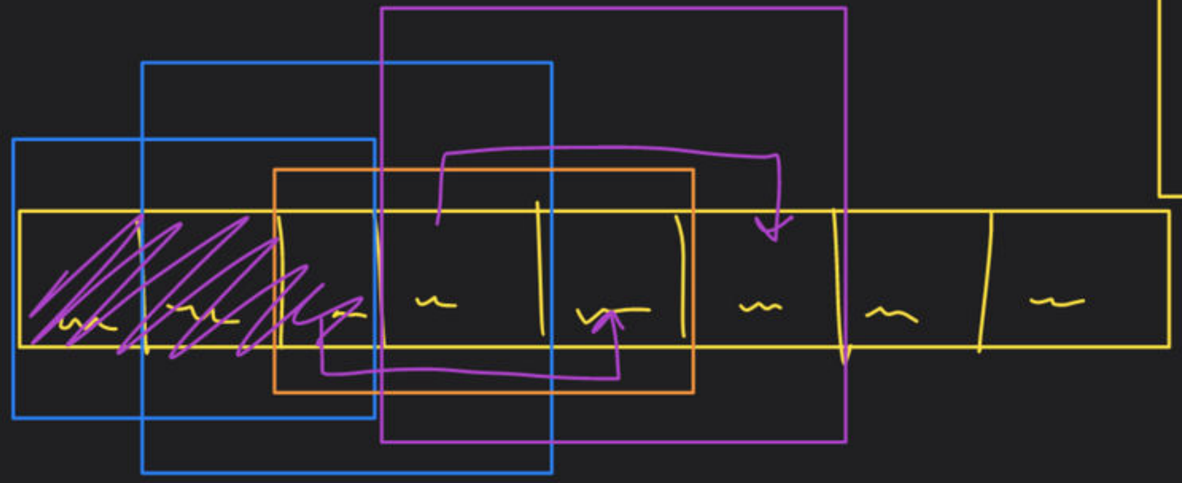
Sliding Window Technique

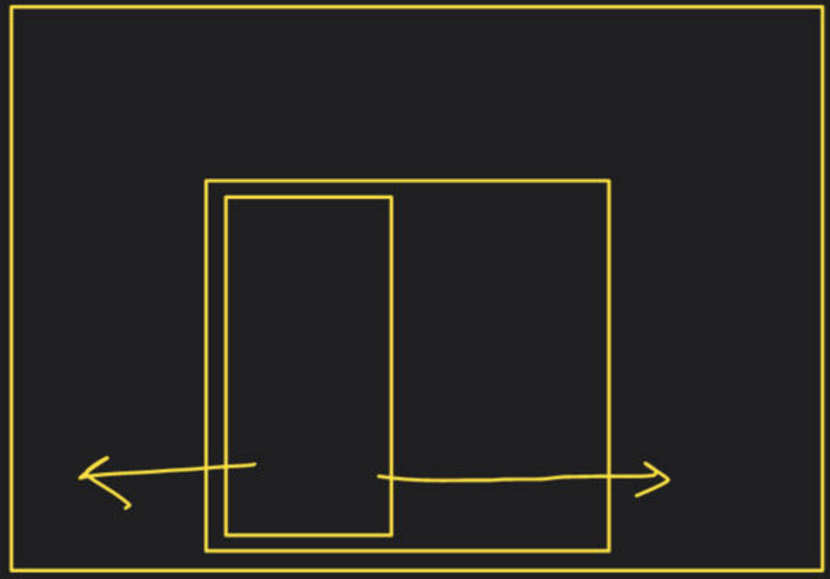
Special class

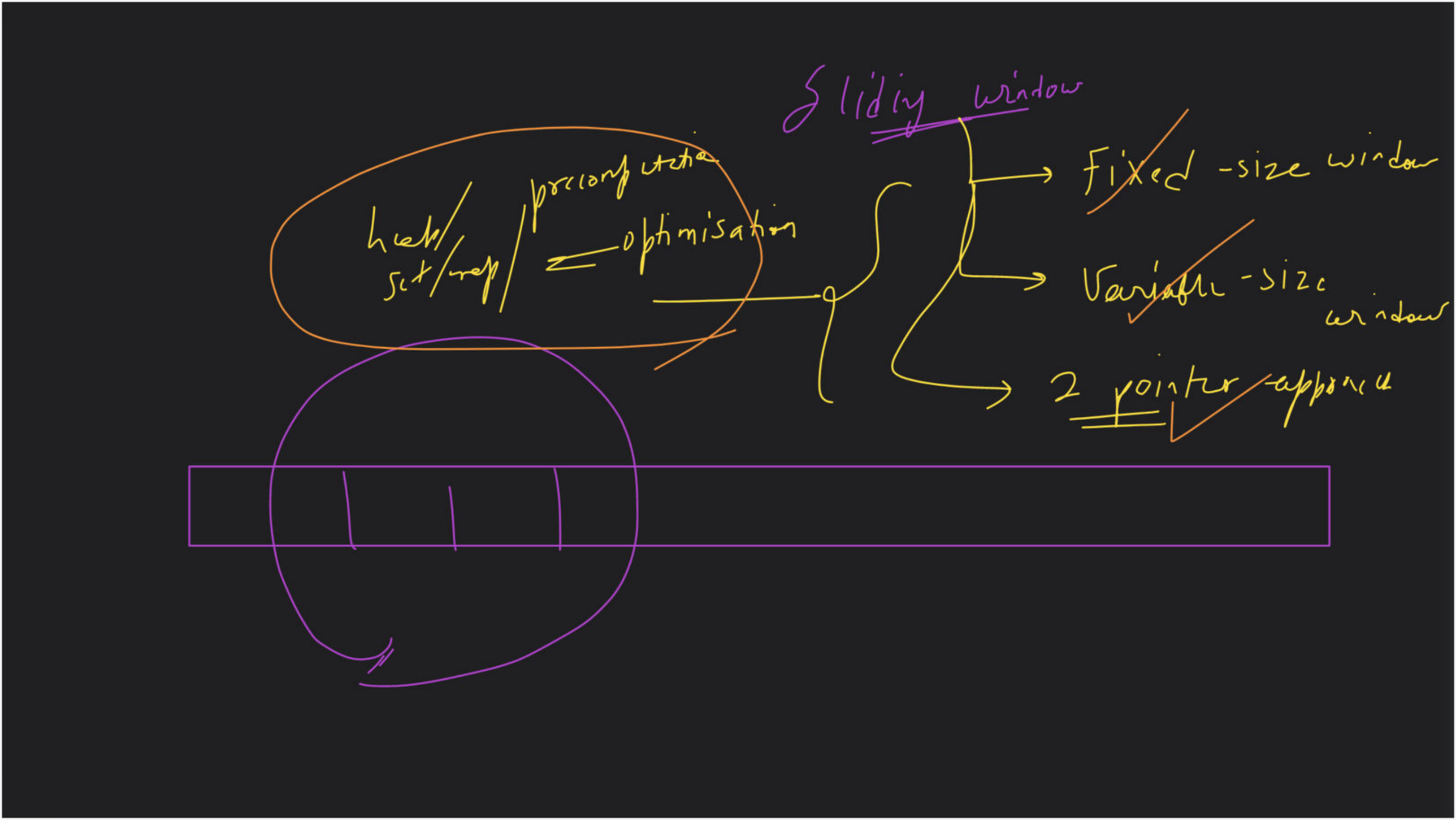
Love Babbar • Mar 14, 2024

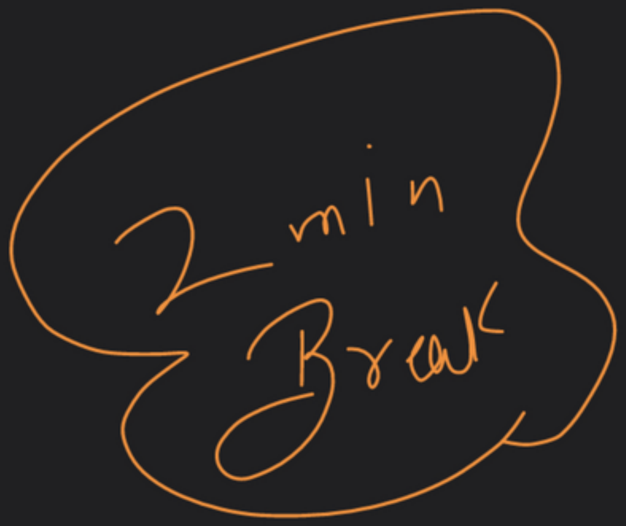
Sliding Window







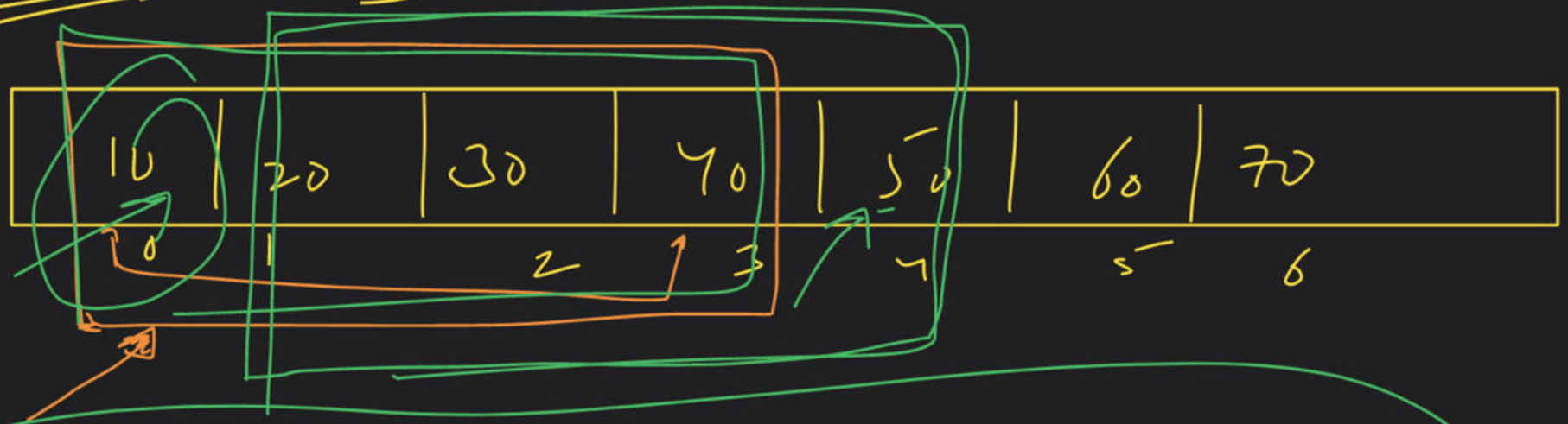

Sliding window
hash/set/map
precomputation
= optimisation
Fixed-size window
Variable-size window
2 pointer approach


2 min
Break

① Fixed - size window:

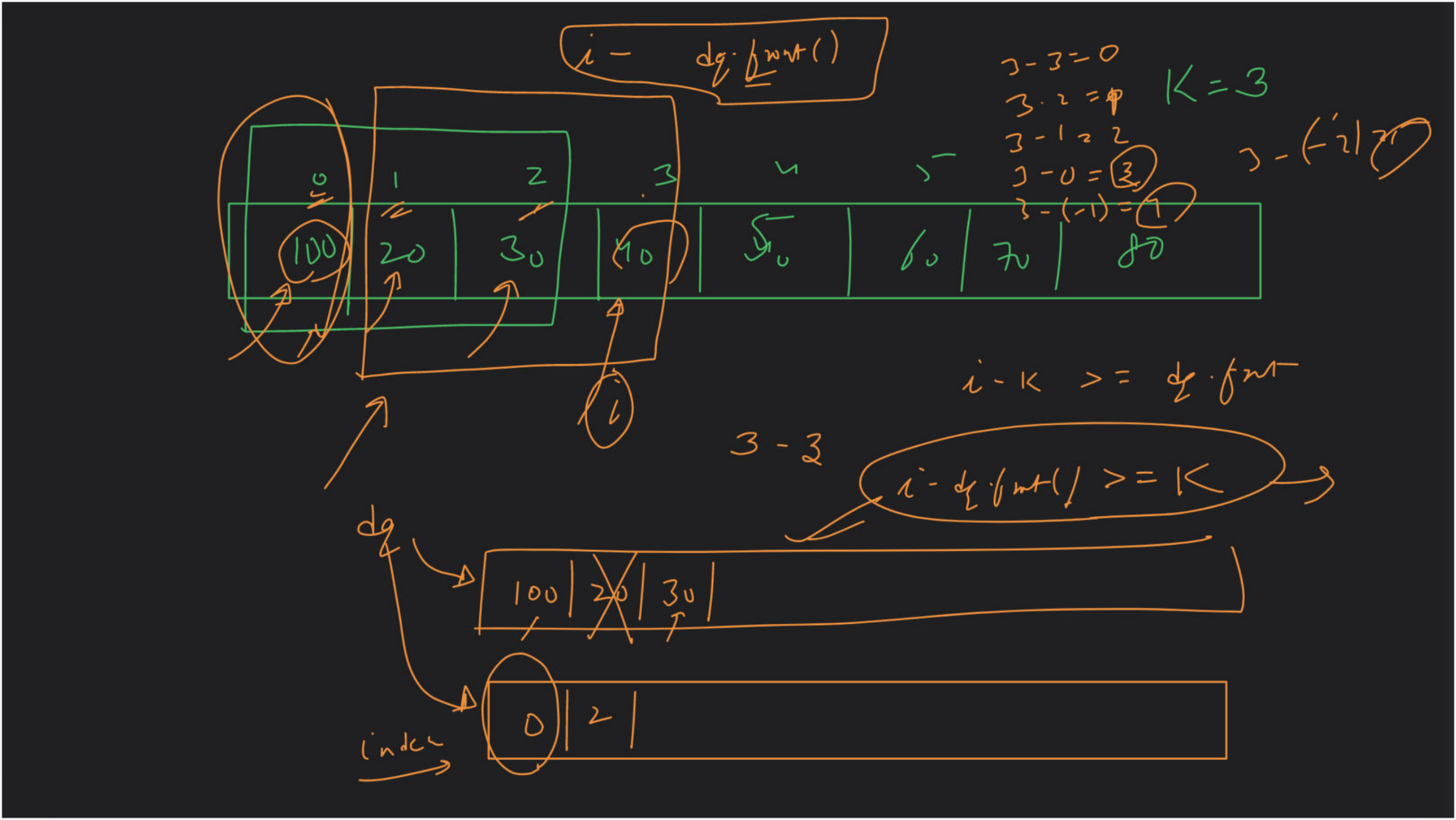
size = 4

K size

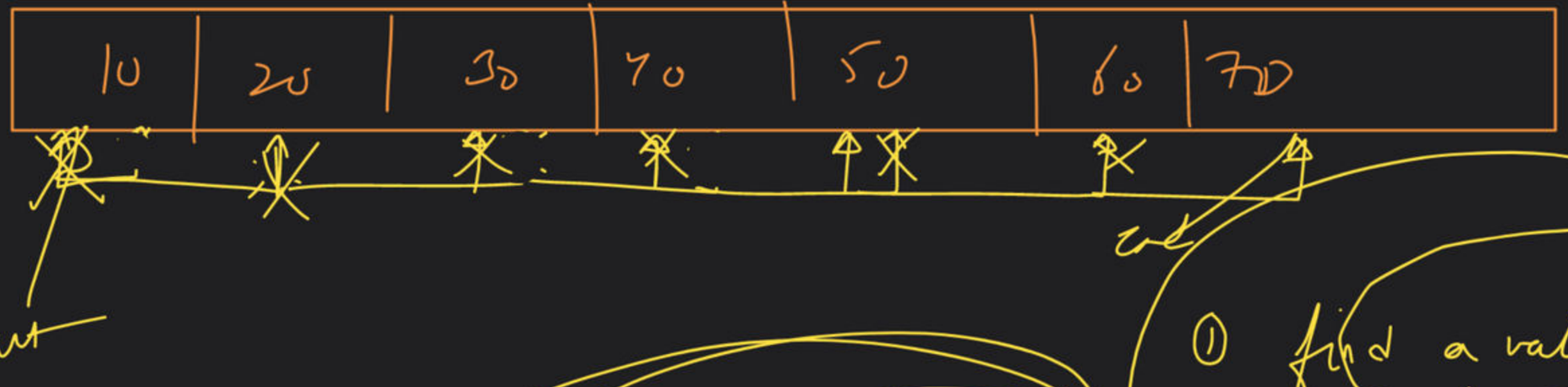


① process first "K" elements → initial state

② process remaining windows → removal
addition
an state


i - dq.front()
K=3
3-3=0
3-2=1
3-1=2
3-0=3
3-(-1)=4
0
1
2
3
100
20
30
40
50
60
70
80
i
i - K >= dq.front
3-3
i - dq.front() >= K
dq
100
20
30
0
2
index

Variable size :-



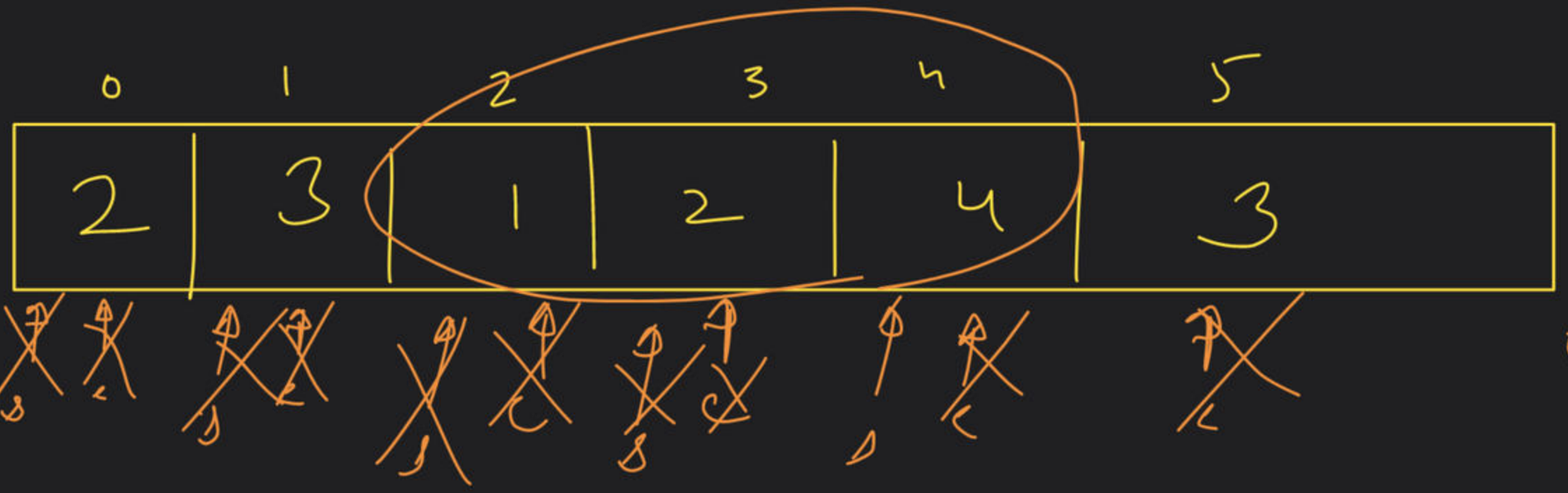
Valid ans → minimise
↳ start++

Invalid ans → explore → end++

While → no valid ans
explore
end++

① find a valid ans
true
minimise
start++

T = 7



2 < 7

5 < 7

6 < 7

8 > 7

6 < 7

10 > 7

7 = = 7

6 < 7

9 > 7

7 = = 7

3

2

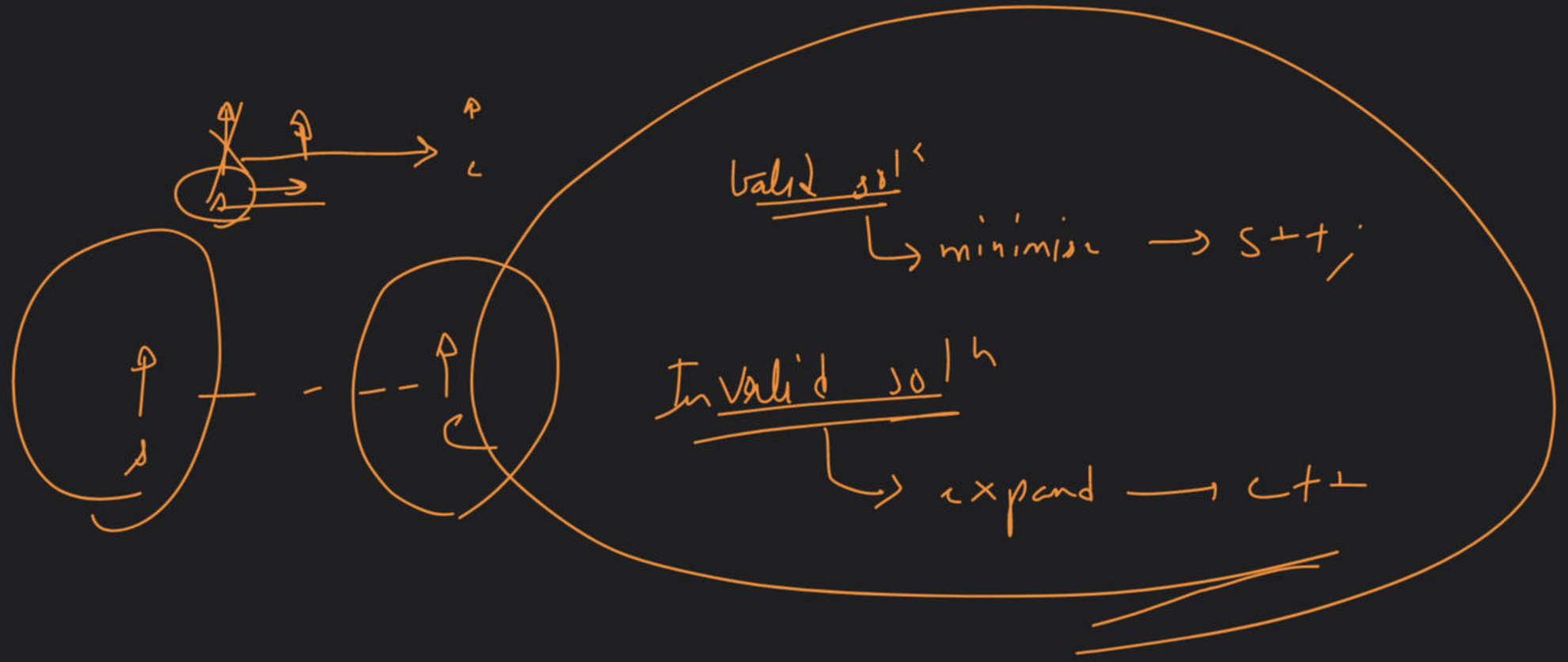
1, 2, 4

4, 3

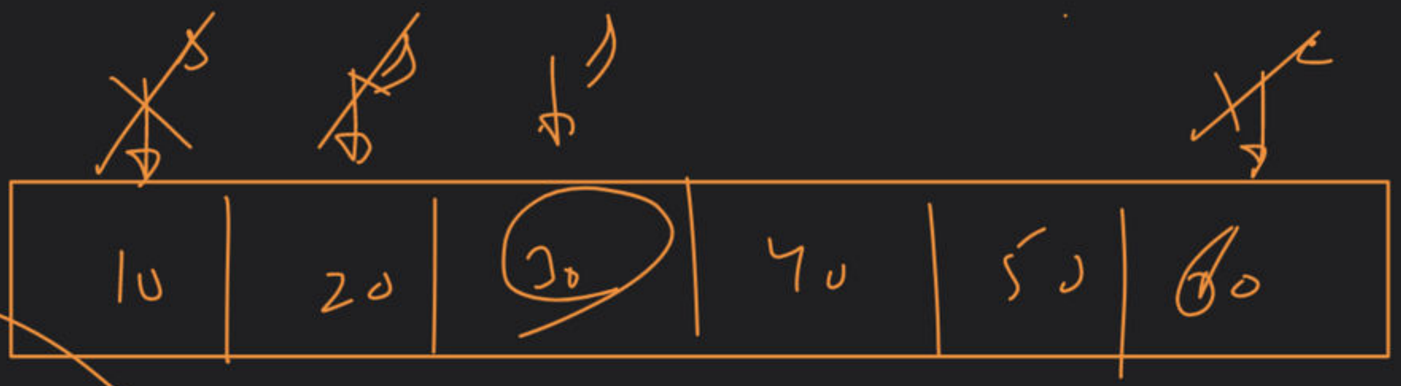
ans

1

Variable size :-



→ 2 ptr



sum < Target → s++

sum > Target → e--

s >= e
↳ Break

n > 90 → e--

Sort

10 + 60 → 70 → != 90

70 < 90 → s++

20 + 60 → != 90

do < 90 → s++

2 ptr

cond1 → nums < num2

cond2 → s !

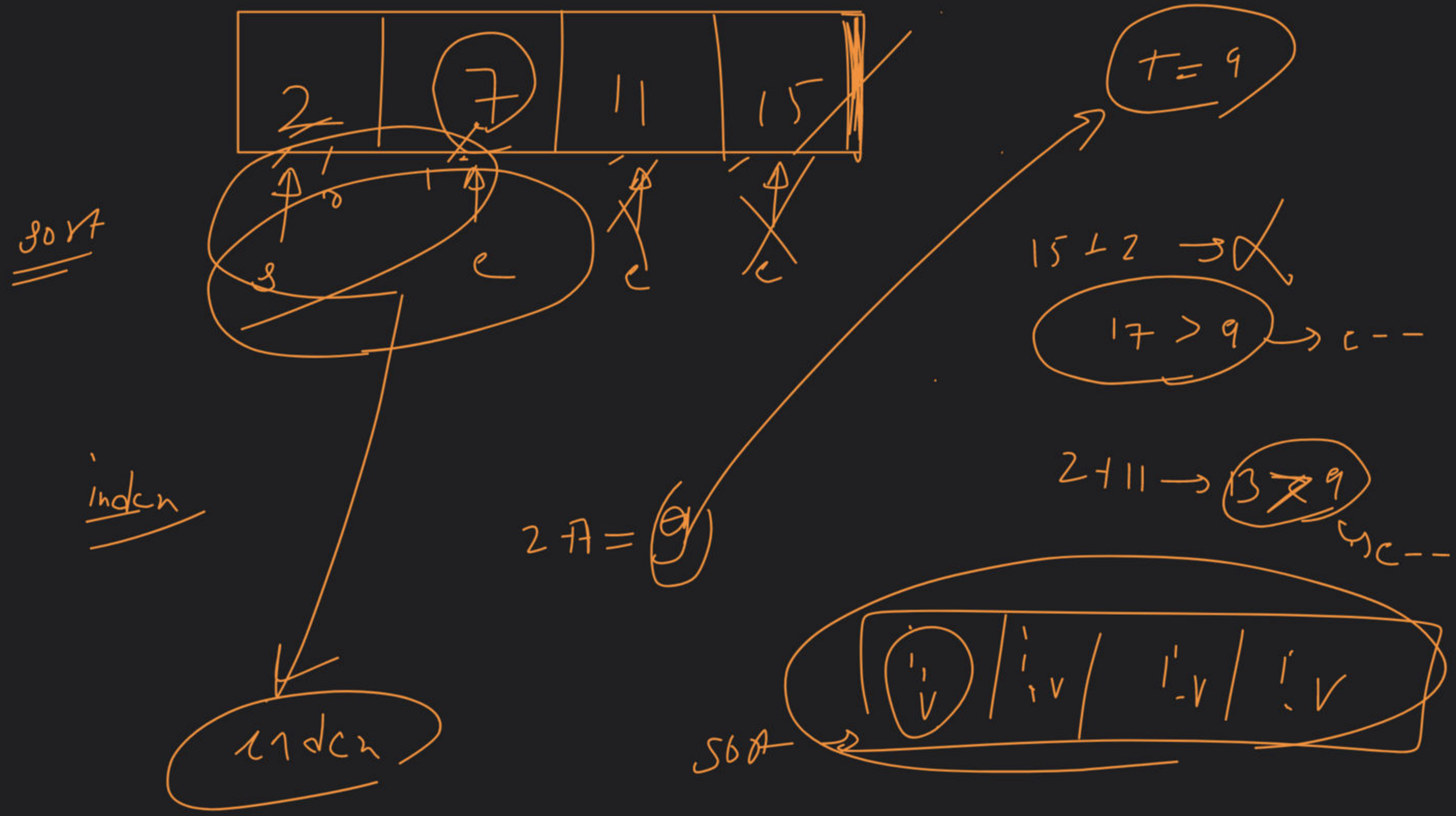
cond3 → e !

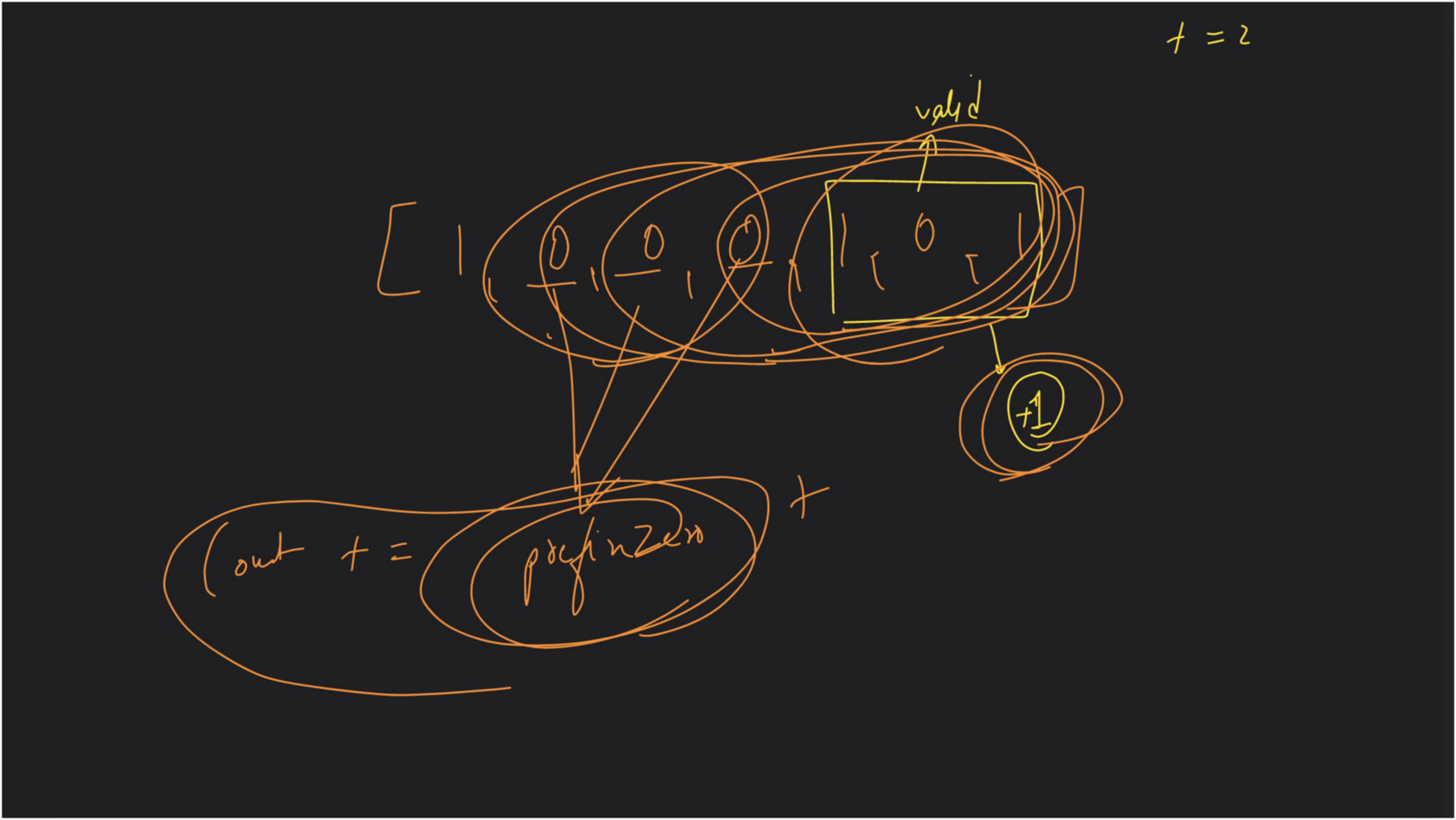
30 + 60 → 90

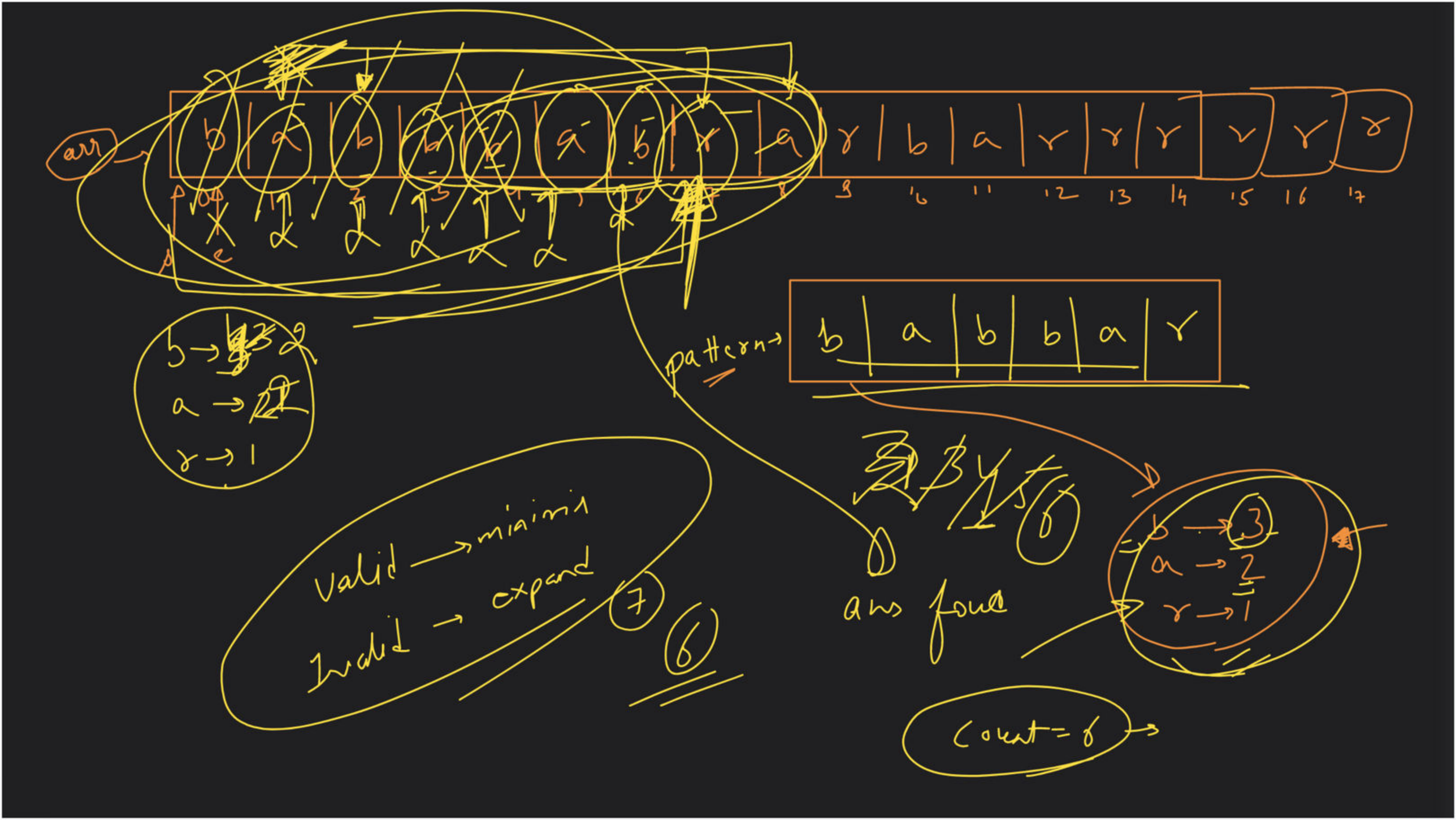
2 ans

sum

90


2
7
11
15
T = 9
sort
17 > 9
index
2+7 = 9
2+11 → 13
sol


t = 2
valid
[1 0 0 0 1 0 1
+1
count += popcnt
+


arr
b a b b b a b r a r b a r r r r r
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
pattern
b a b b a r
b → 2
a → 2
r → 1
Valid → minimise
Invalid → expand
7
6
ans found
b → 3
a → 2
r → 1
count = 6