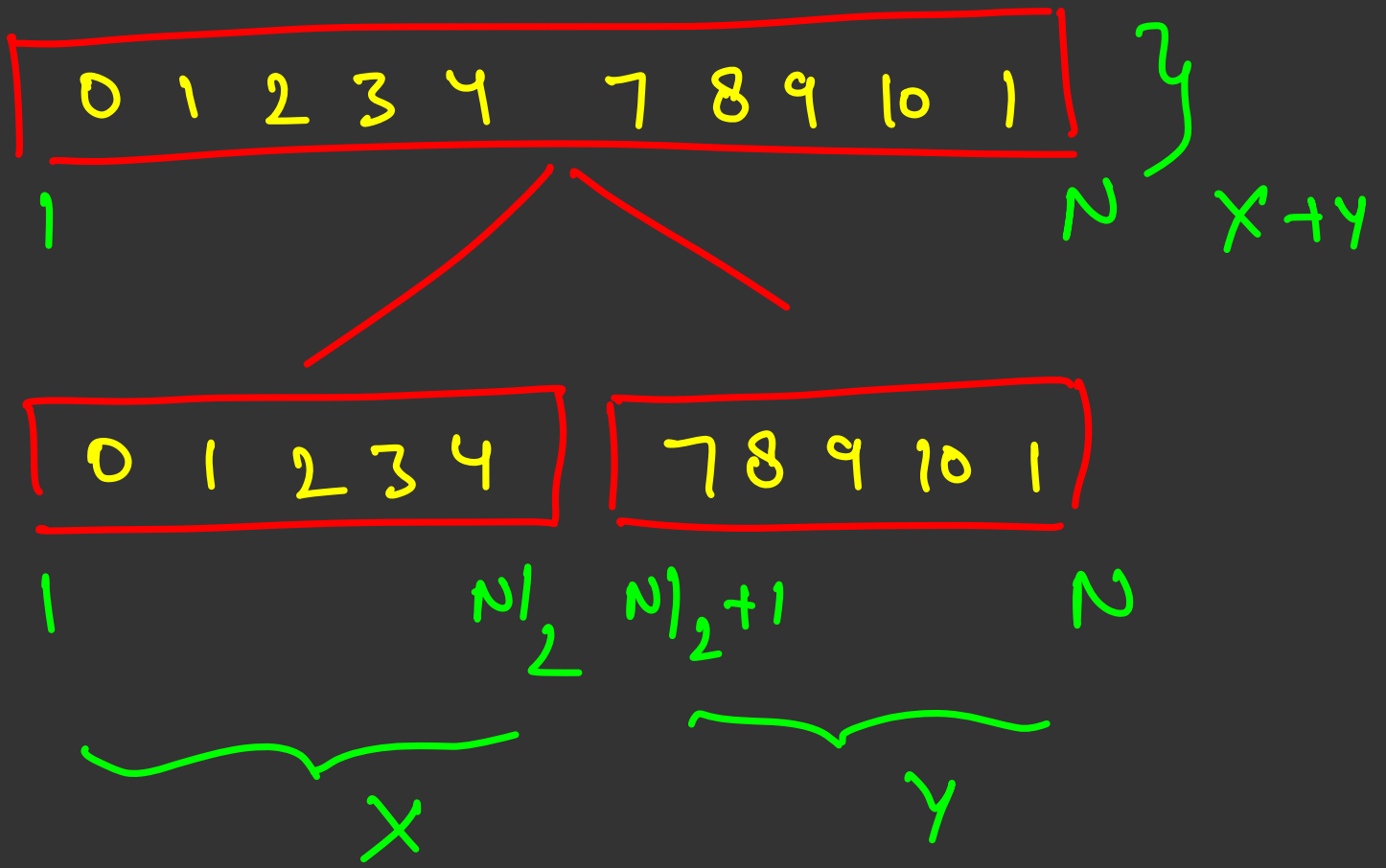
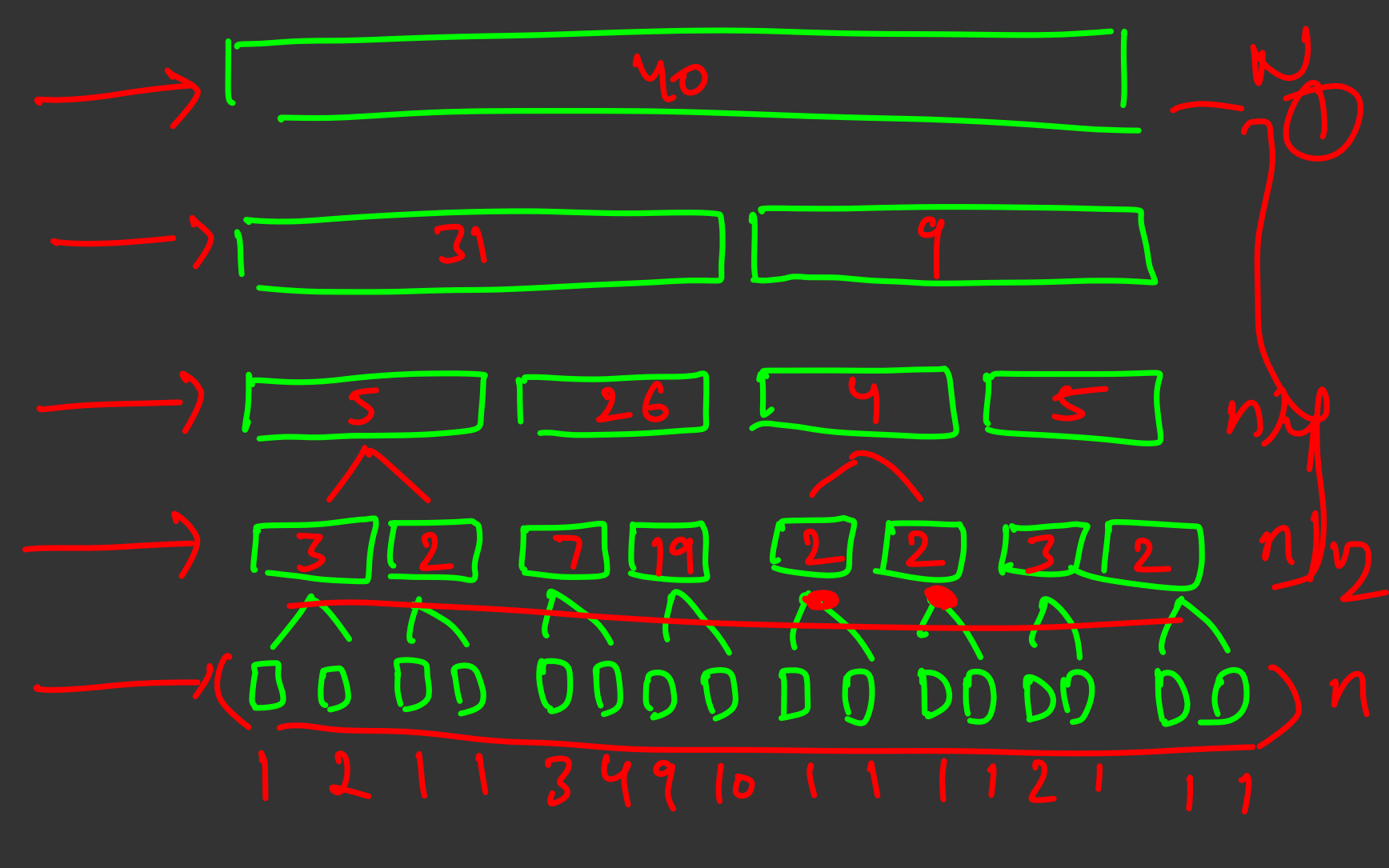
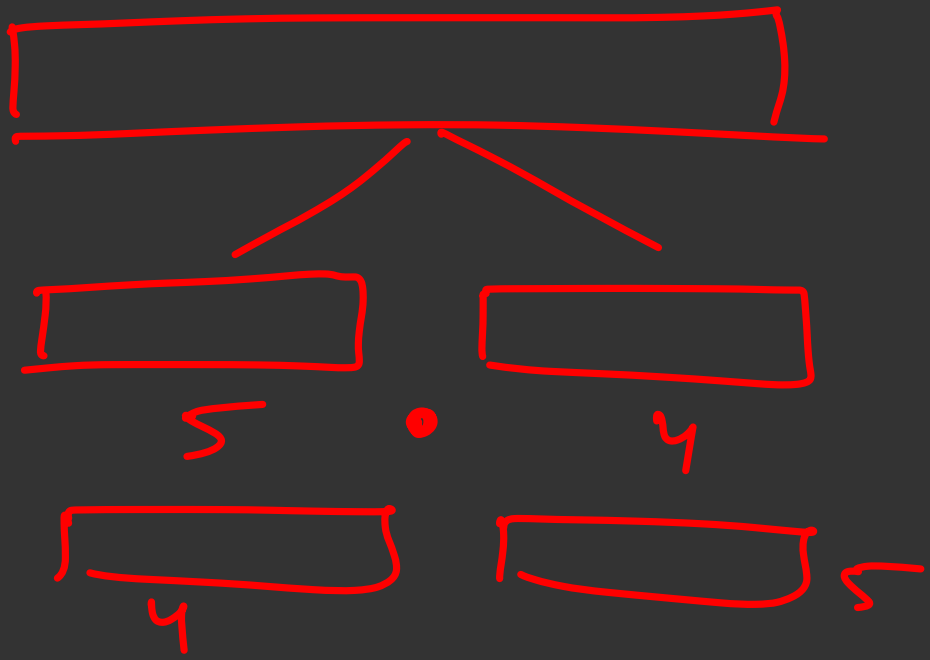

0 1 2 3 4 7 8 9 10 1
1
N
X+Y
0 1 2 3 4
7 8 9 10 1
1
N/2
N/2+1
N
X
Y


40
31
9
5
26
4
5
3
2
7
19
2
2
3
2
1 2 1 1 3 4 9 10 1 1 1 12 1 1 1
n
n
n
n
1
2

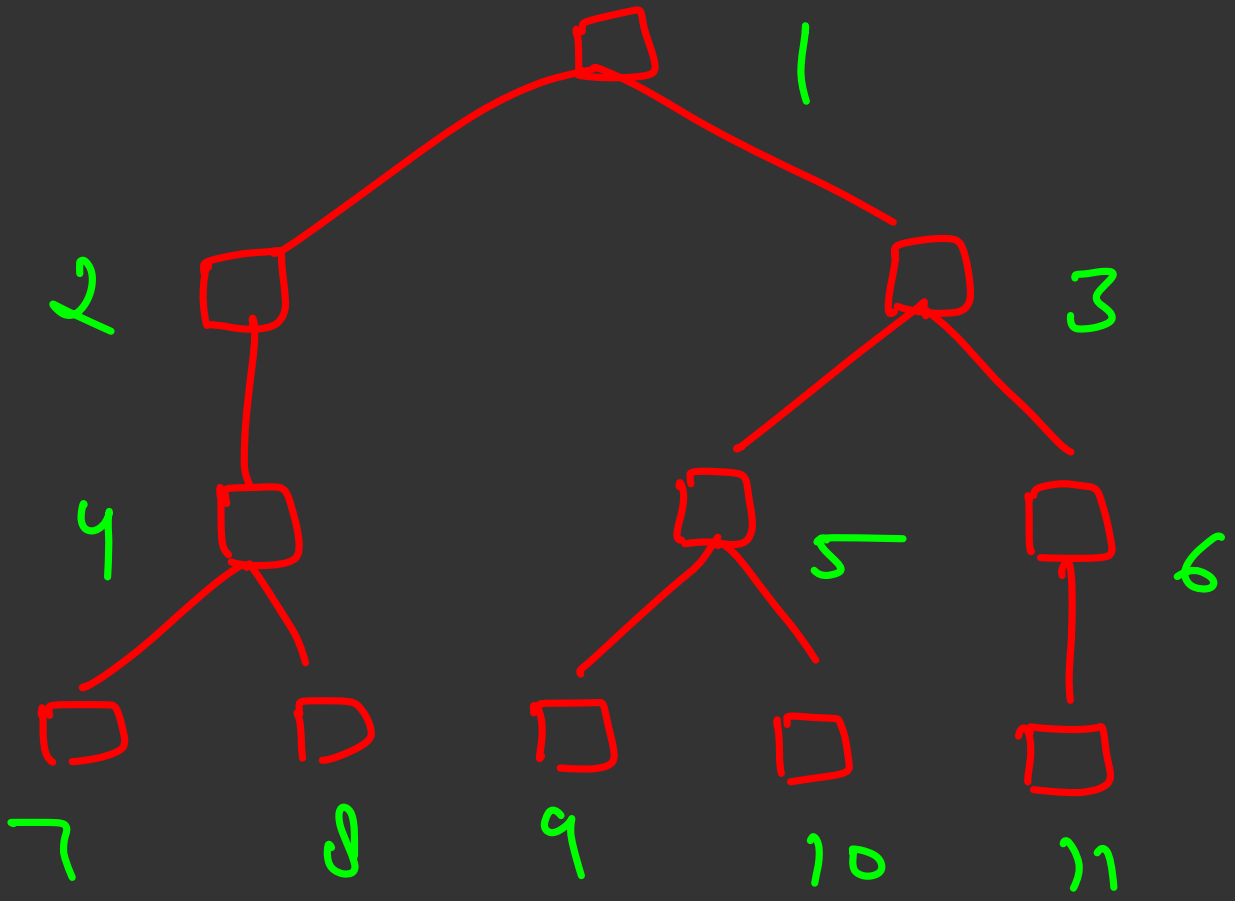


N = 9

$$n + n/2 + n/4 - \cdots - (1)$$

$$\leq \underline{2n}$$

(n) $\longrightarrow$ array elements $\longrightarrow$ $\leq \underline{2n}$



$11 \leq 2 \cdot 5 = 10$

if you have n elements in array

— total segment tree nodes $\leq 2n$

if n is a power of 2

→ 5 → 8

n → $\leq 2n$ (nearest power of 2 $\geq n$)

if n is not a power of

$$2^k < n < 2^{k+1}$$

$2^k < n$

$2^{k+1} \rightarrow 2 \cdot 2^k$

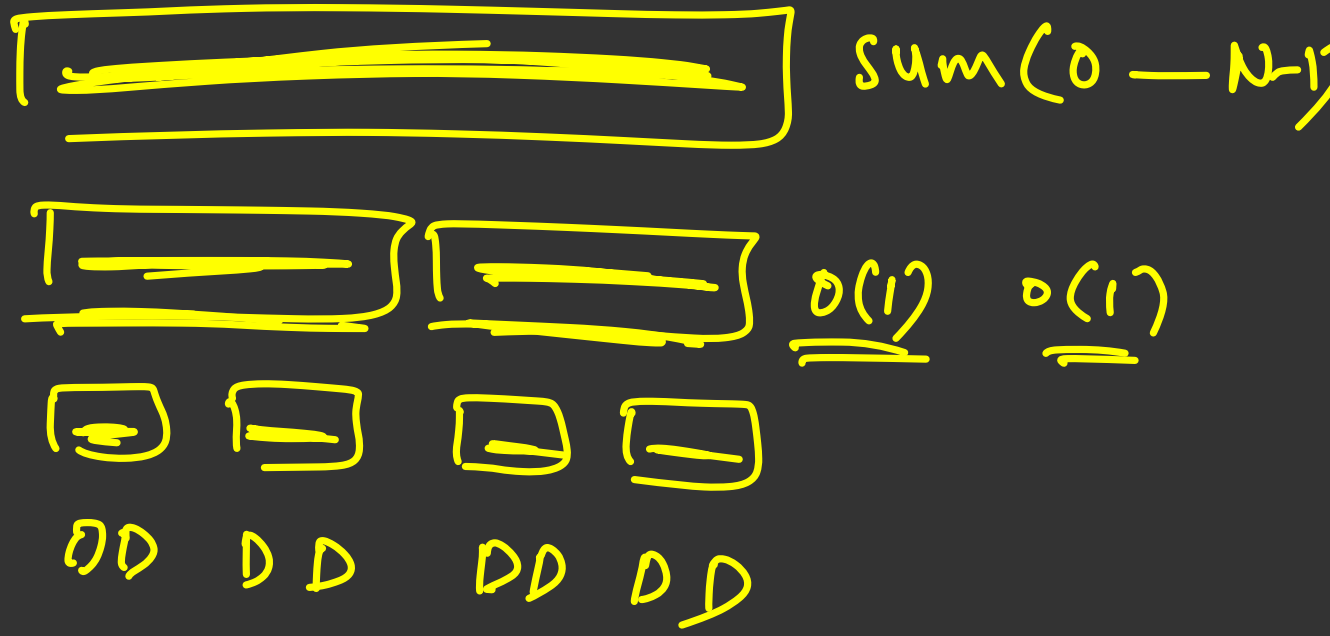
$\leq 2n$

$n \rightarrow$ # nodes $\leq 2n$

$\leq 2n \rightarrow$ # nodes $\leq 2 \cdot (2n)$

for any general $n \rightarrow$ # $\leq 4n$

$\leq 4n$


sum(0 — N-1)
O(1)
O(1)

$4n \rightarrow O(4n) \rightarrow \boxed{O(n)}$

$n, 2n \rightarrow 4n$

$n, 4n$ S.C $\rightarrow O(n)$

$\leq 4n$

ST

B.B

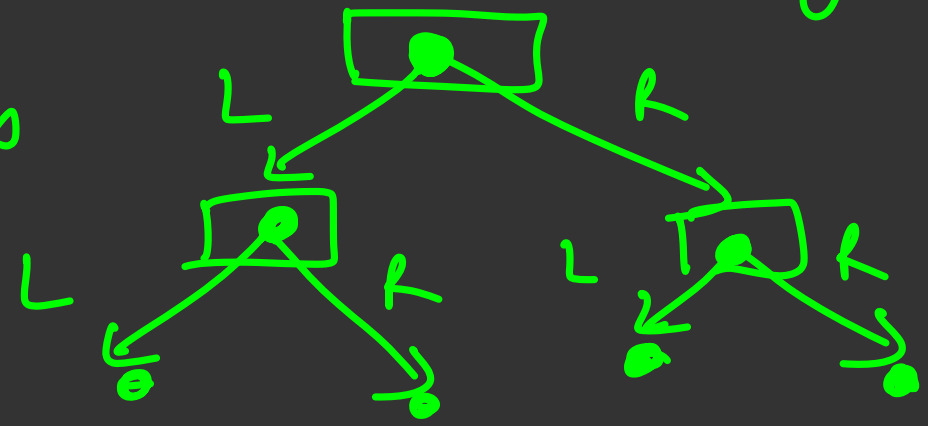
FT

99.99%

B.B

Represent a Segment Tree in Memory

— Pointers

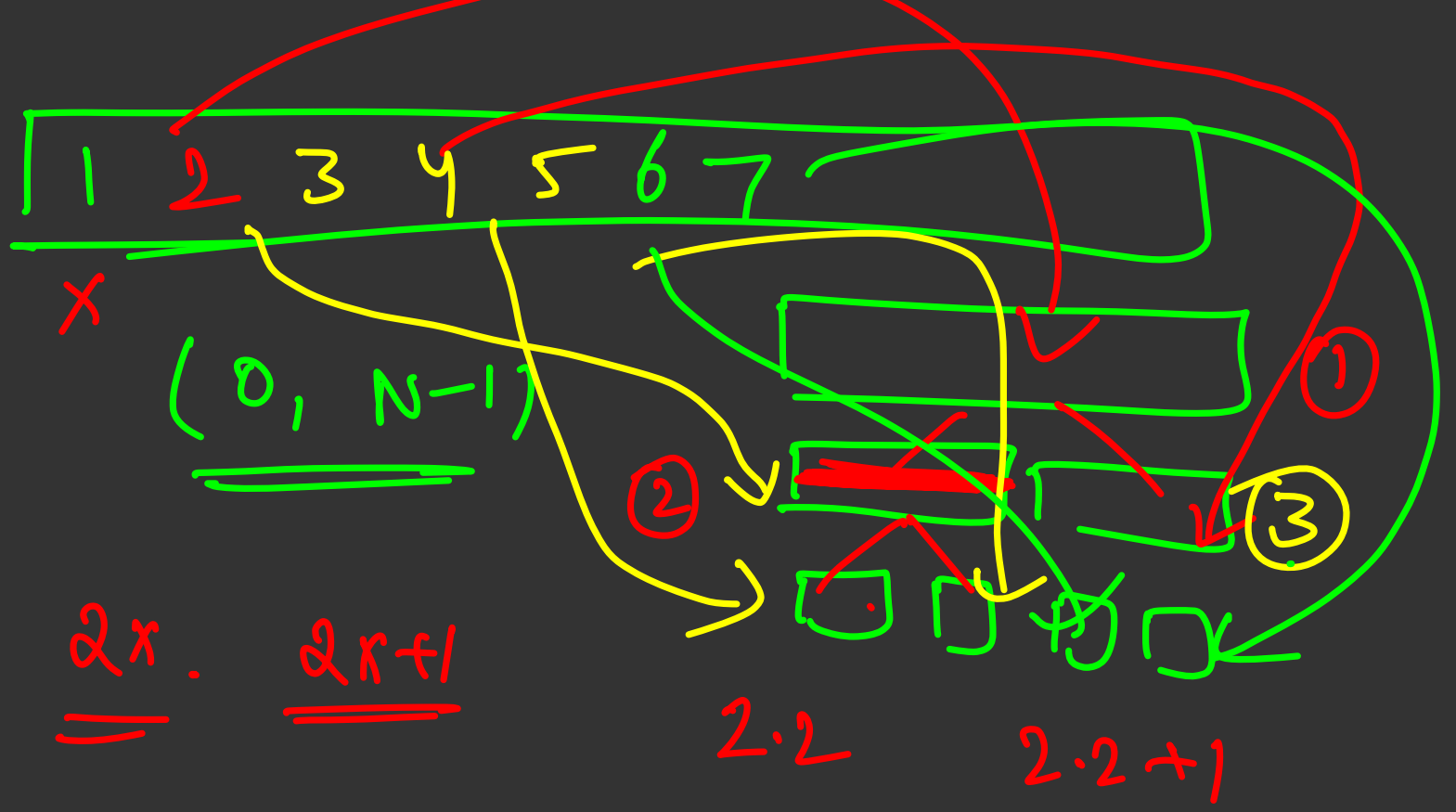


Full Binary Tree

0

(Every node has either 2 children or none) —> Full Binary Tree

— Array Approach


1 2 3 4 5 6 7
x
(0, N-1)
2x
2x+1
2·2
2·2+1
1
2
3

Pointer ✓

Space ✓

$n \to 2n$

$2n$ $\times 2$

int $4n$

Array $O(1)$ ✓

$4n$

$4n$

$\leq 4n$

$2n \leq x \leq 4n$

$n \rightarrow 2n$

$2n$ nodes

$2n$ pointers

$4n$ nodes

$n-1 \quad 2n+5$

$2n+10$, $2n+10$

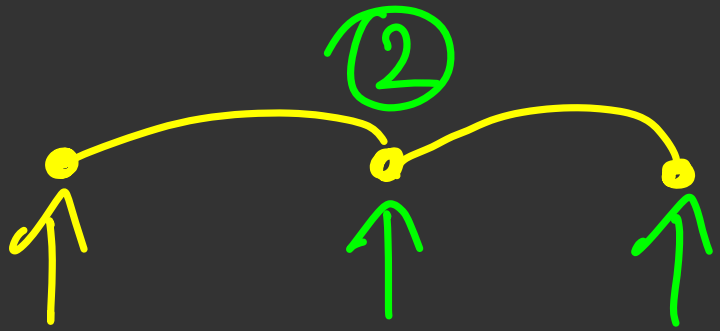
$4n$

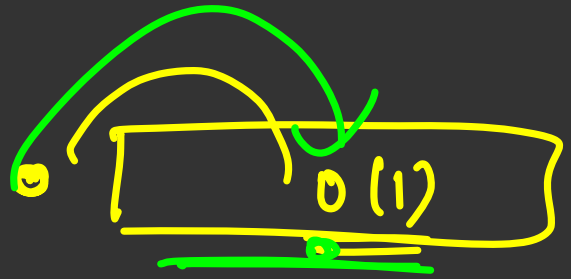
Pointer

Array

Space ✗

Time ✓





Root → (1)

Left child of node x → 2x

Right child of node x → 2x+1


1
1
2
3
5
11
9
8
3
6
5
3
5
1 2 1 5 3 2 1 2

node → val = arr(start)

when node → (start, end) → arr(start+1) — — arr(end)

```
void update (int node, int start, int end
             , int ind, int val)
{
    if (start == end) {
        arr(ind) = val
        seg(node) = val }
    int mid = (start + end)/2
    [start, mid]  , [mid+1, end]
    if ( ind > mid)
        update( 2*node+1, mid+1, end, ind, val)
```

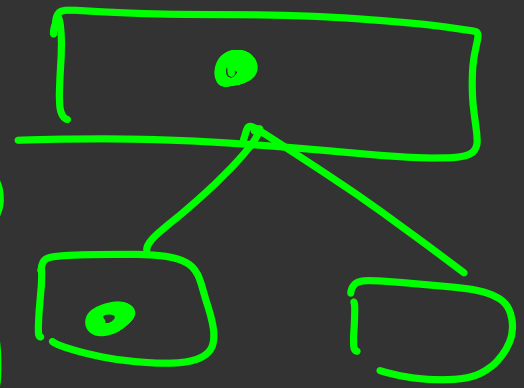
else

update(2·node, start, mid, ind, val)

① choice

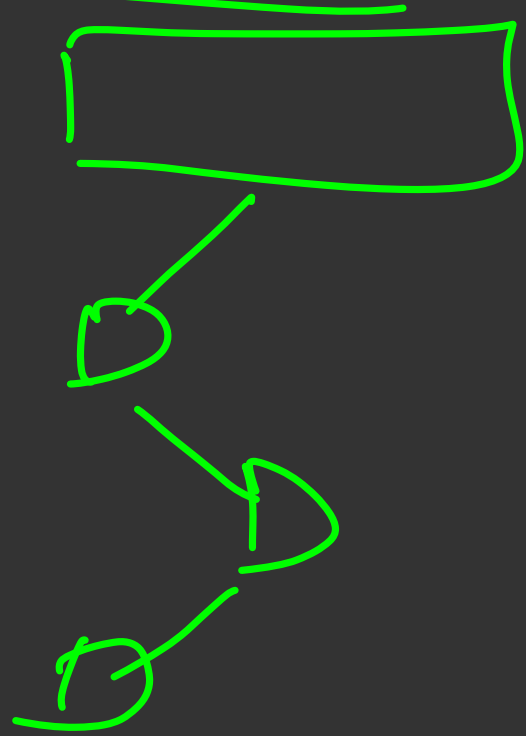
tree(node) = tree(2·node)

+ tree(2·nodet 1)



}

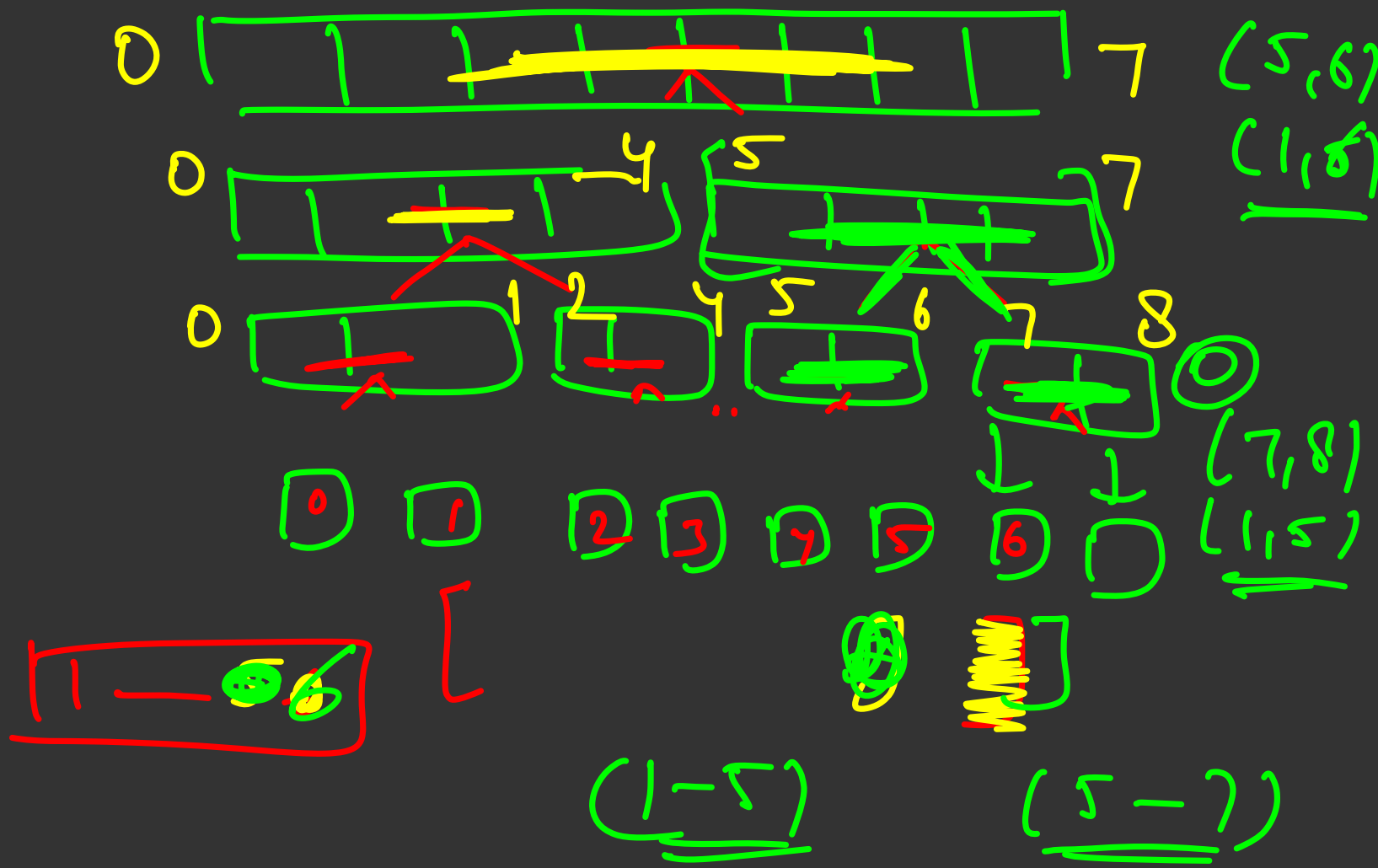
O(height of tree)

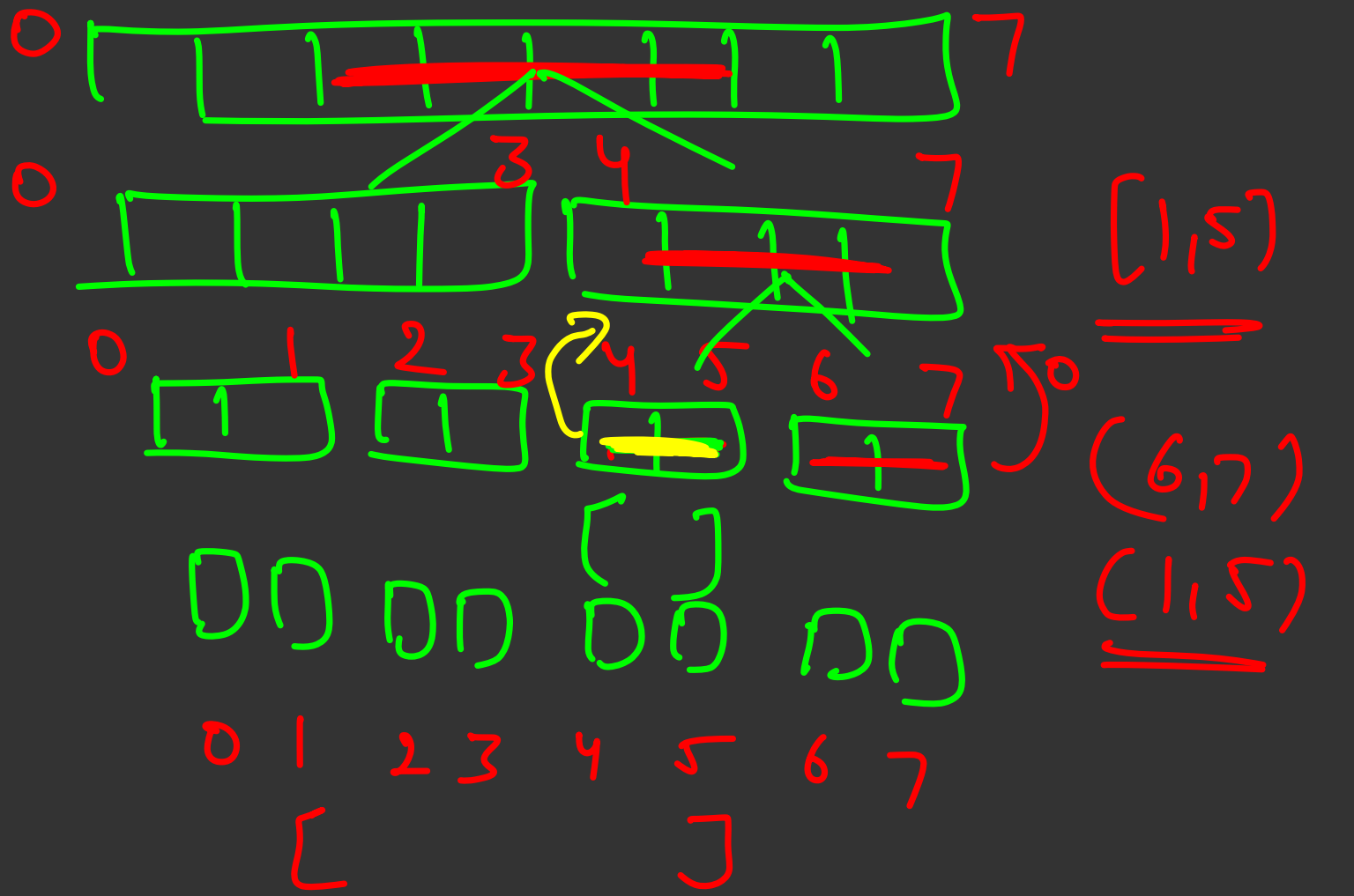


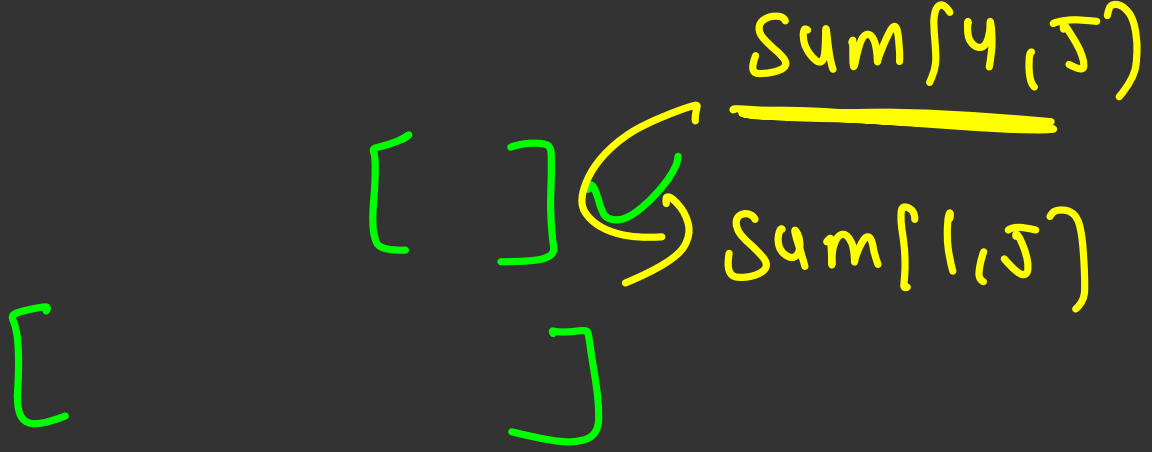
n , $n/2$ $n/4$ - - - - - - 1

$\log(n)$ $\longrightarrow$ $O(\log n)$

$O(\log n)$


0
7
(5,6)
0
4
5
7
(1,8)
0
1
2
4
5
6
7
8
(7,8)
0
1
2
3
4
5
6
(1,5)
1 — 6
(1-5)
(5-7)


0
7
1 1 1 1 1
0
3
4
7
1 1 1 1
1 1 1
0 1 2 3 4 5 6 7
1 1 1 1
[1,5]
(6,7)
(1,5)
0 1 2 3 4 5 6 7
[ ]


sum[4,5)
[ ]
sum[1,5]
[ ]

3 conditions

(l, r) → query

(s, e) → segment tree node

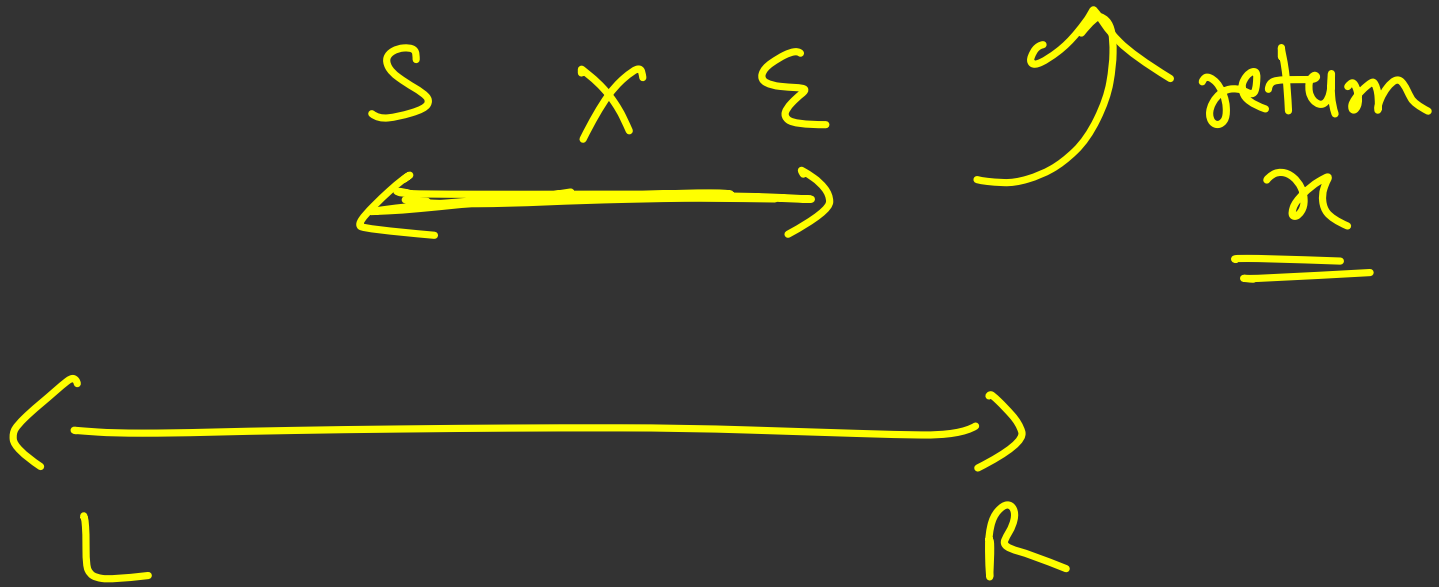
if (s, e) and (l, r) are completely disjoint

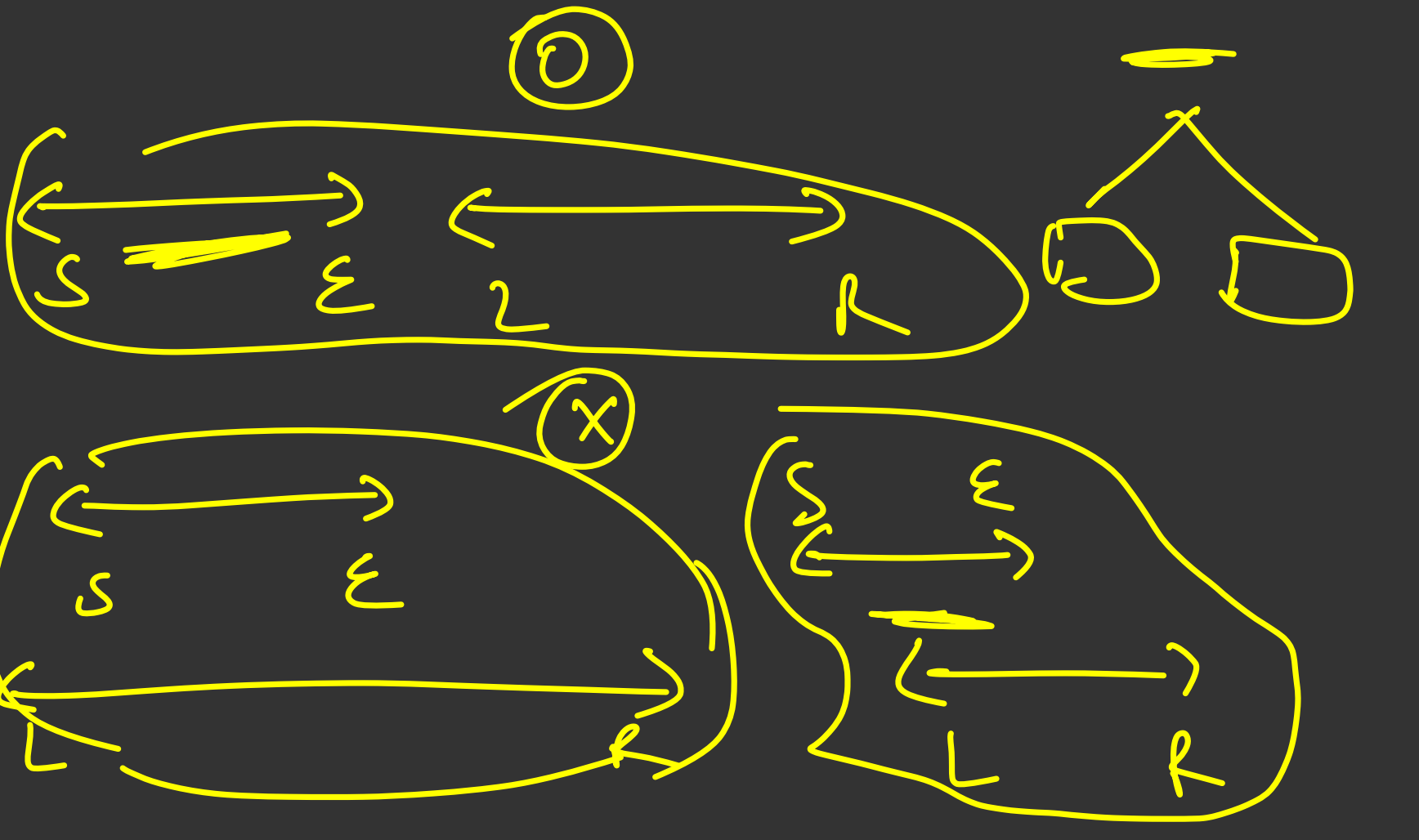
↳ 0

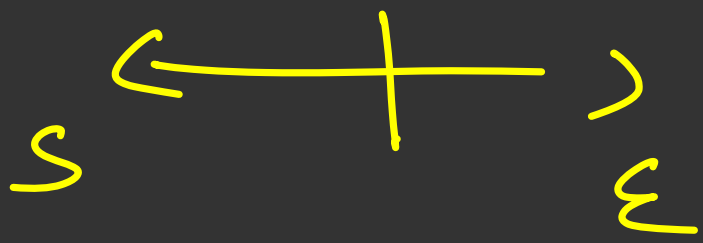
else if (s, e) is completely enclosed in (l, r)

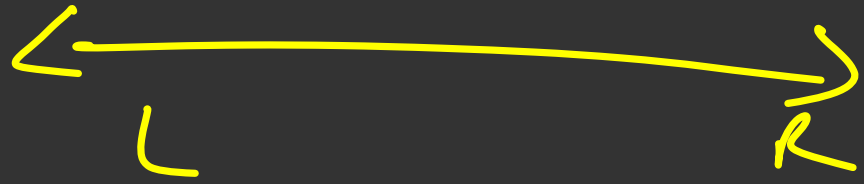
return seg.sum (s, e)

else

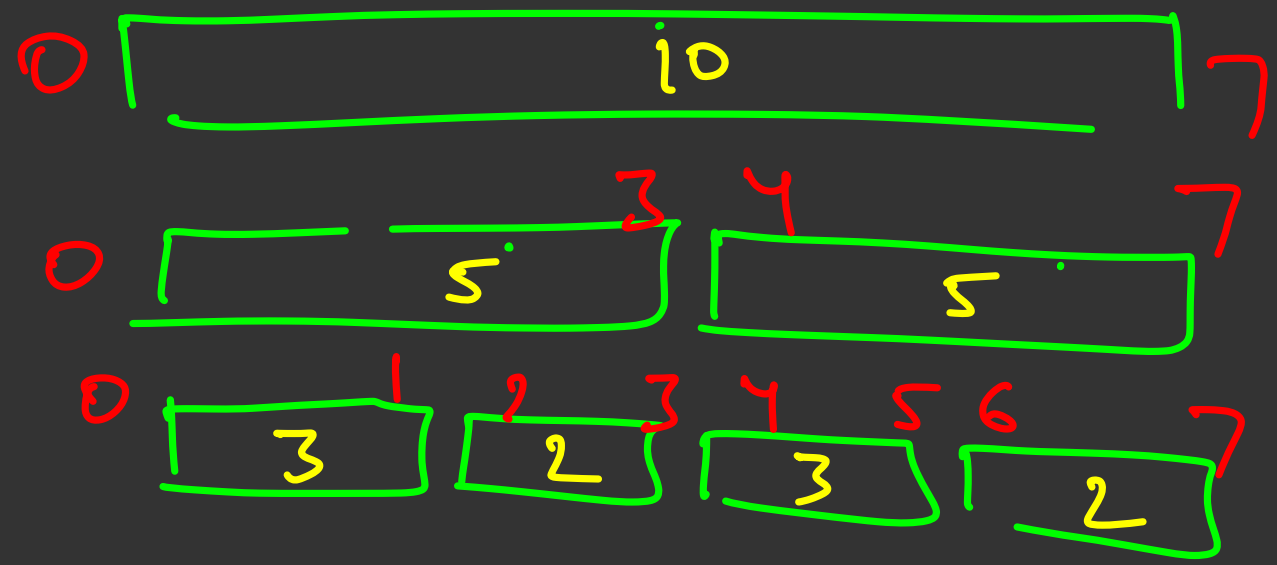



O
S
ε
L
R
X
S
ε
L
R
ε
S
L
R





$$ans = [(S, M) \Delta (L, R)]$$
$$[\pm (M \pm 1, \epsilon) \Delta (L, R)]$$

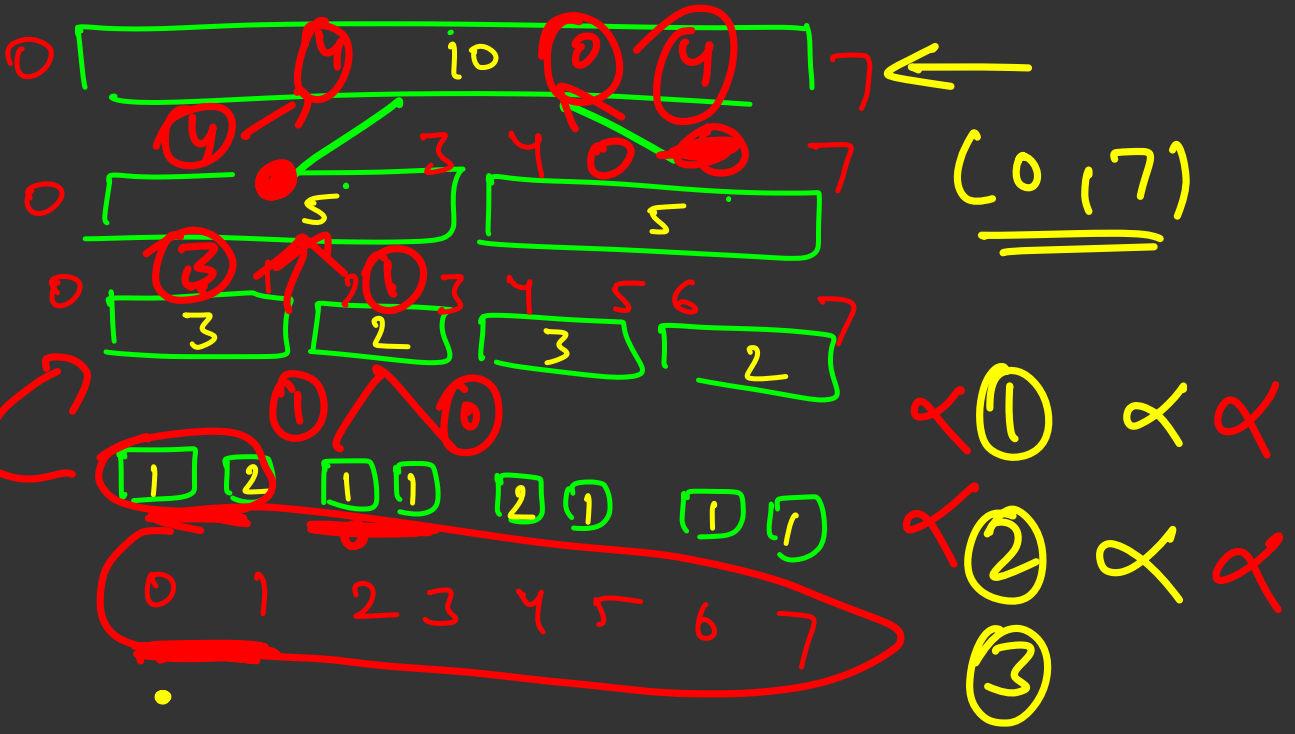


4

(0-2)

1 2 1 1 2 1 1 1

0 1 2 3 4 5 6 7



(0,7)

(0,3)

(4,7)

(0,2)

(0,1)

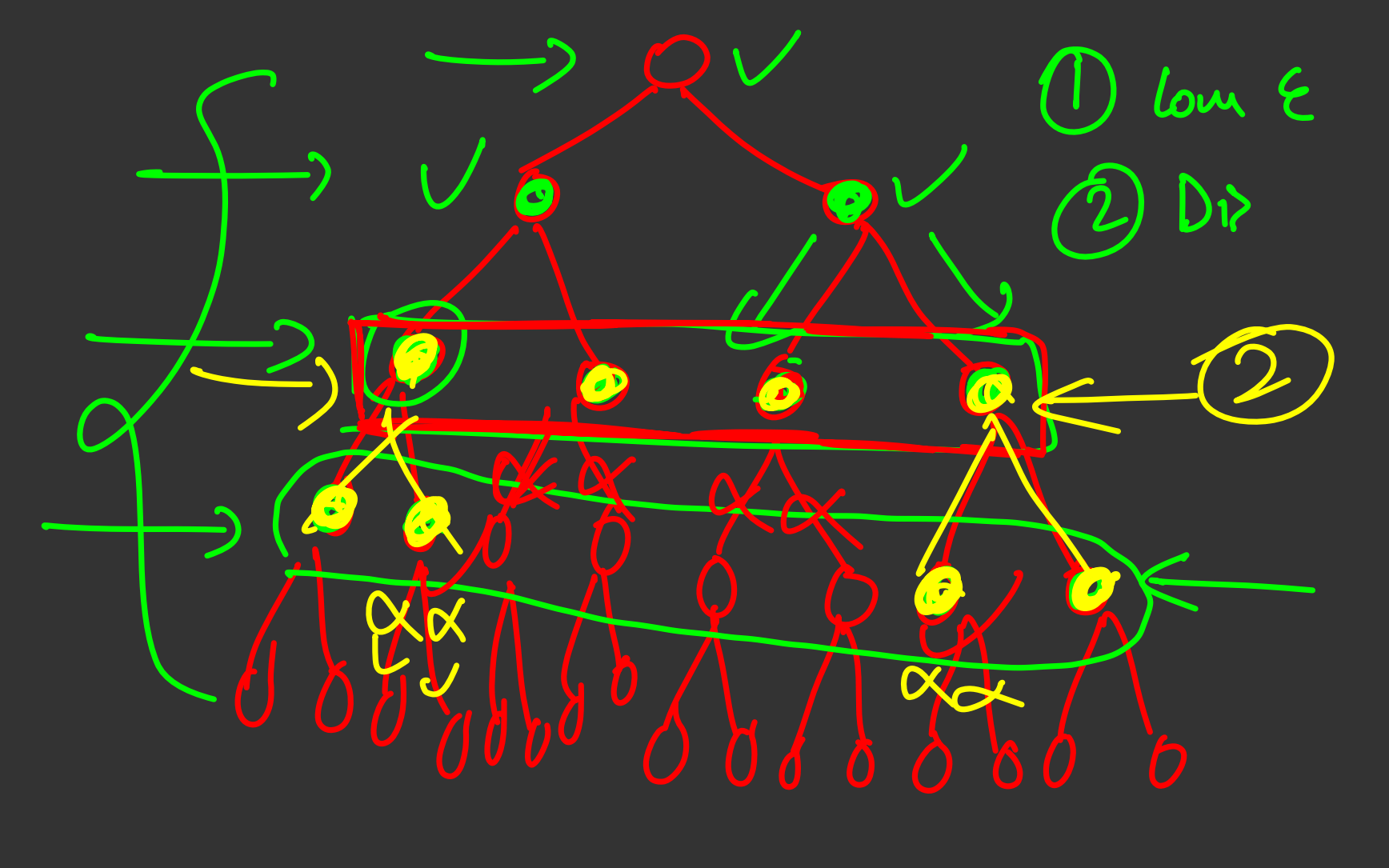
(2,3)

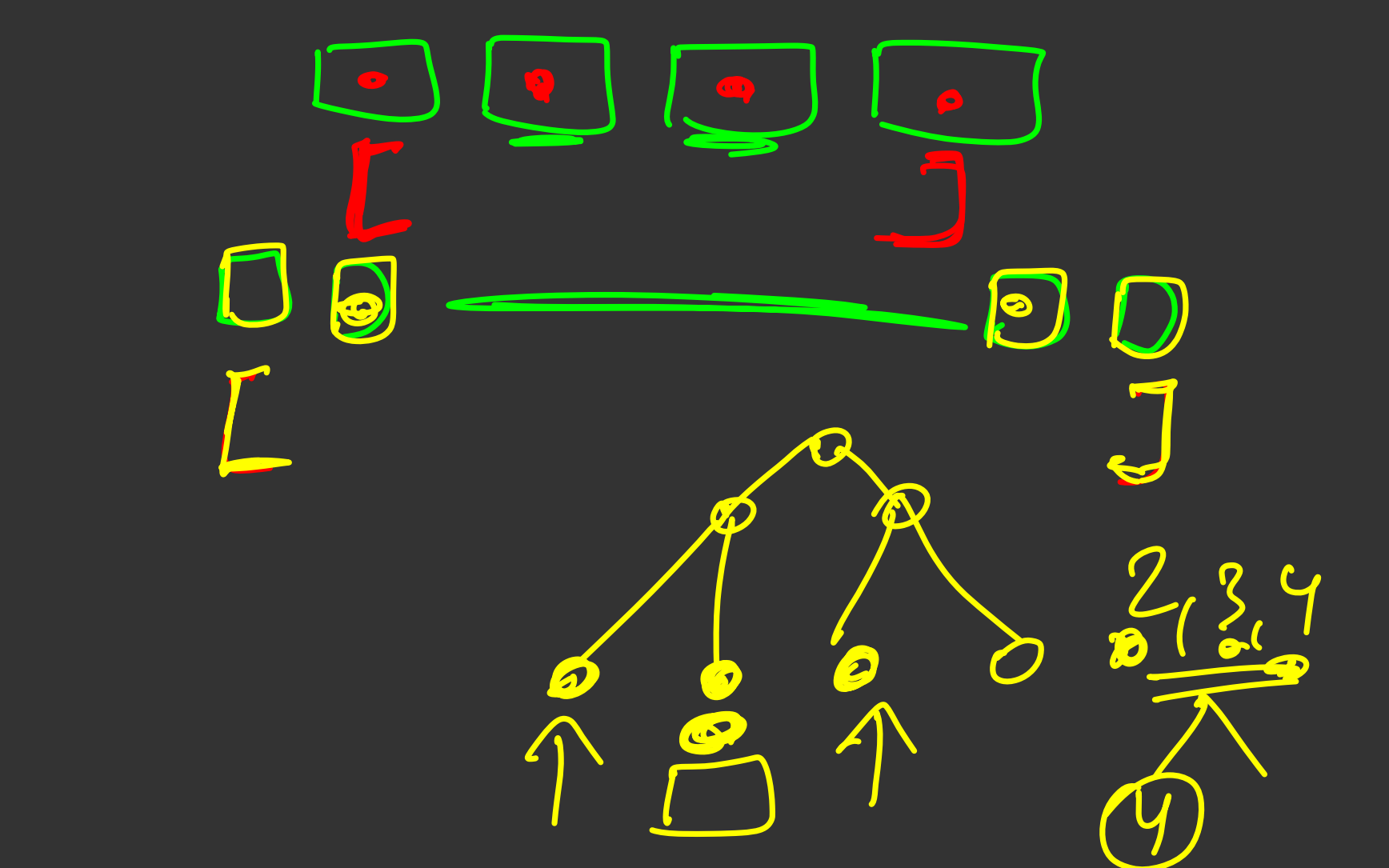
Time Complexity → Query

L, R

O(log n)

{Claim ①, ② → for any level in the tree you will never consider more than 4 nodes}


① lom ε
② Dᵢ
2


2 3 4
4

Binary search of a [redacted] Queries

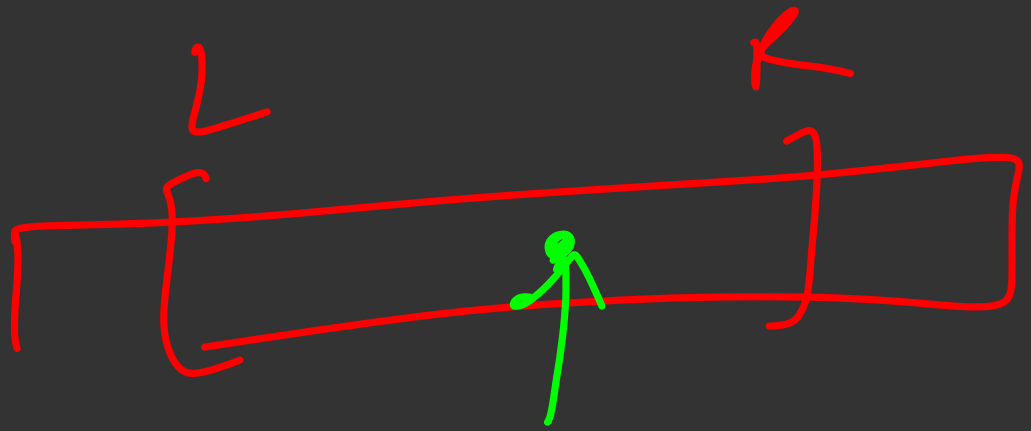
$O(\log^2 n)$ }

Binary search of the segment tree

$O((\log n)$

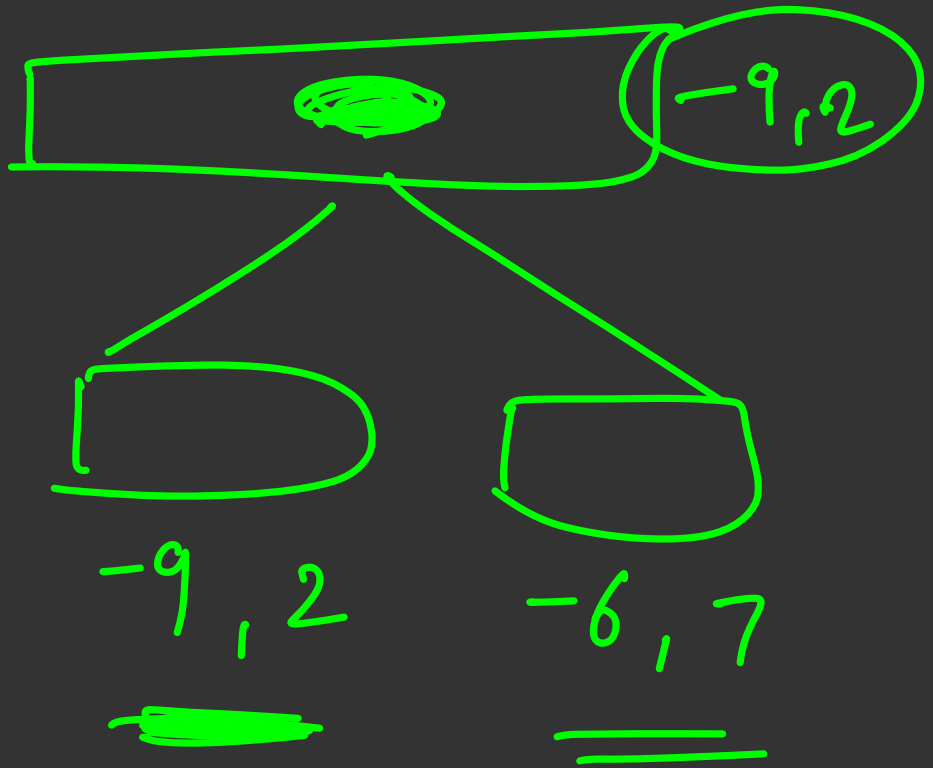
Problem C → $O(n^2 \log n)$

$O(n \log^2 n)$



—> min element from (L to R)

give the index


-9,2
-9,2
-6,7

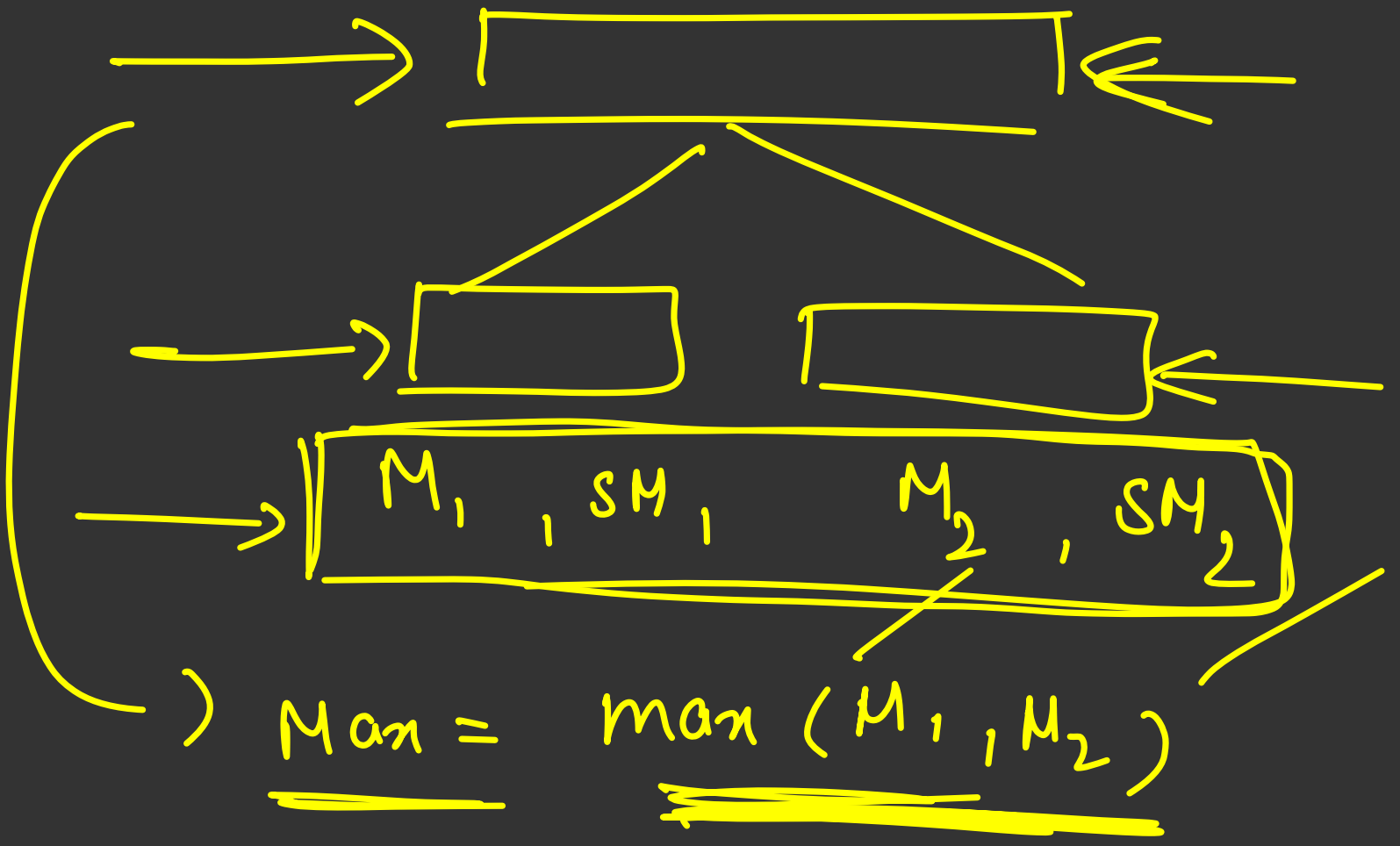
N → 1e5 → array elements

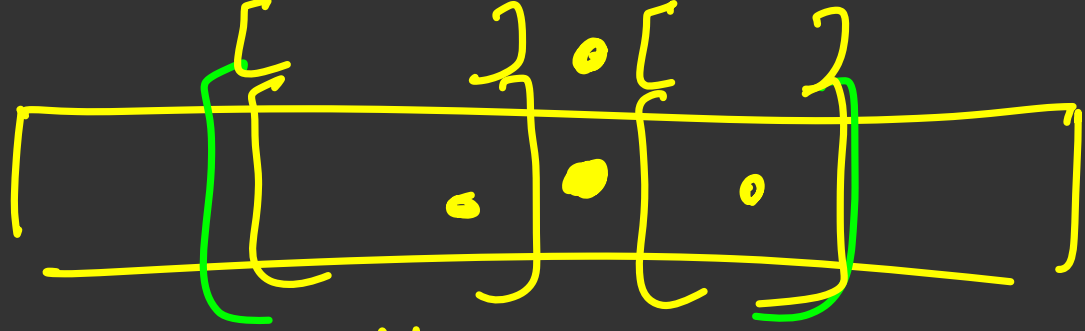
Q → 1e5

all value in array are distinct

Q 1 → (update value at ind → val)

Q 2 → { find the second maximum in a range from L to R


M1 , SM1
M2 , SM2
Max = max (M1 , M2)



L M_1 M_2 R

Second max

① → ②

max (M_1 , M_2)

③

3rd maxim

kth max

$k \cdot \log n$

$k \cdot \log k \log n$