Range Queries
Segment Trees
(Lazy Propagation)

Point Updates + Range Queries ~ O(log n)

O(n) Build, Update + Query ~ O(log n)

Update value at index "pos" = x ①

Range sum → (l, r) $\sum_{i=l}^{r} a_i$ → ②

$O(\log n)$ time

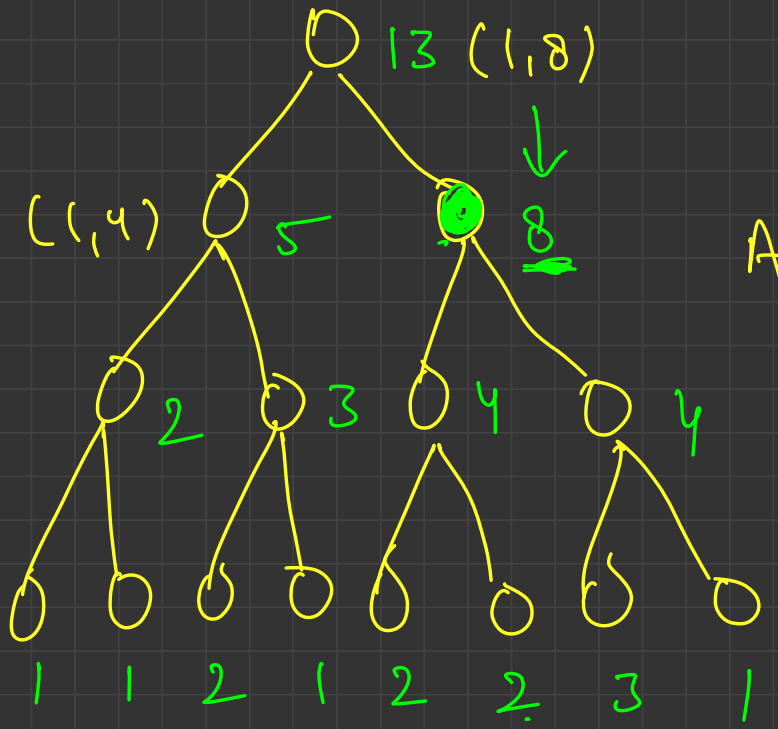
① ② ② ① ① ——— —

Update all the values from l to r

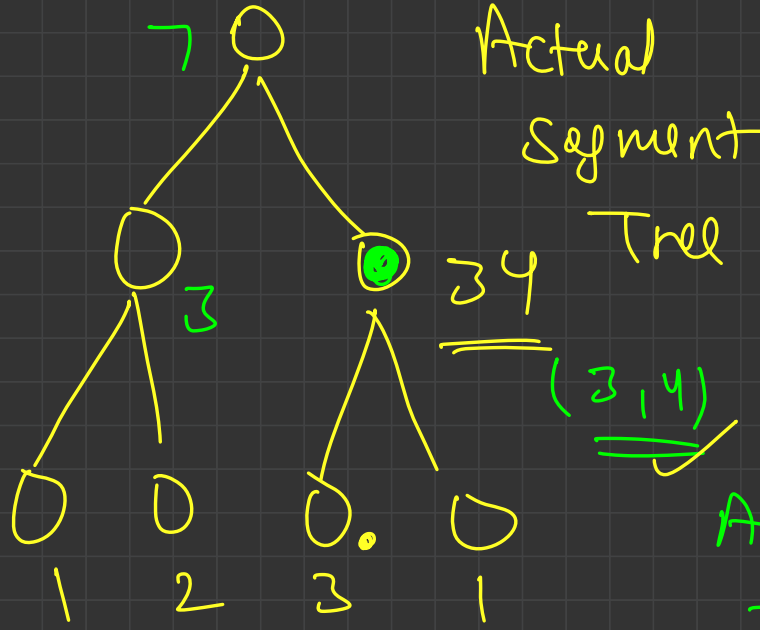
by adding x to ①
all of them

Range sum → ②

} O(log n)



Add 5 to all indices in range (5,8)



Actual Segment Tree

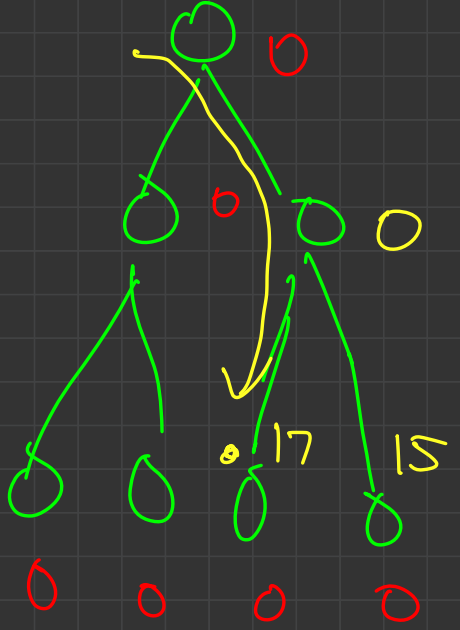
3 4

(3, 4)

Add +5

add 2 to (3,3)

Copy / Lazy Segment Tree



Add +10 to (3, 4)

Range Queries

Query

① Complete overlap [[ ]]

Node

O(log n)

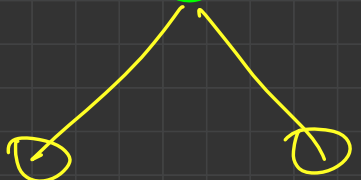
Return value

② Disjoint [ ] [ ]

Return Default Value

③ Partial overlap [[ ]]

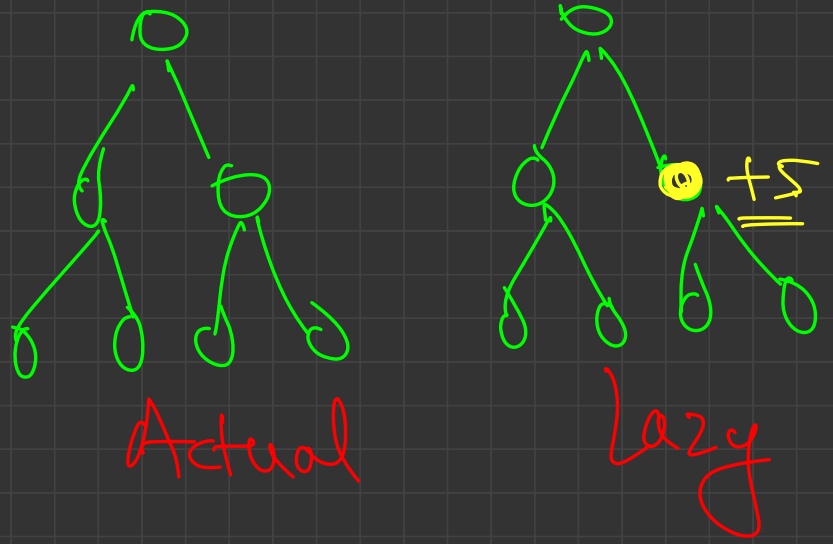
[[ ]]



Range Update

① Complete overlap [[ ]]

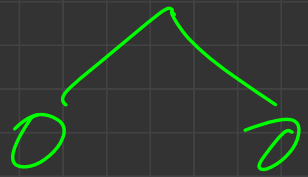
Add 5 to all nodes in given range

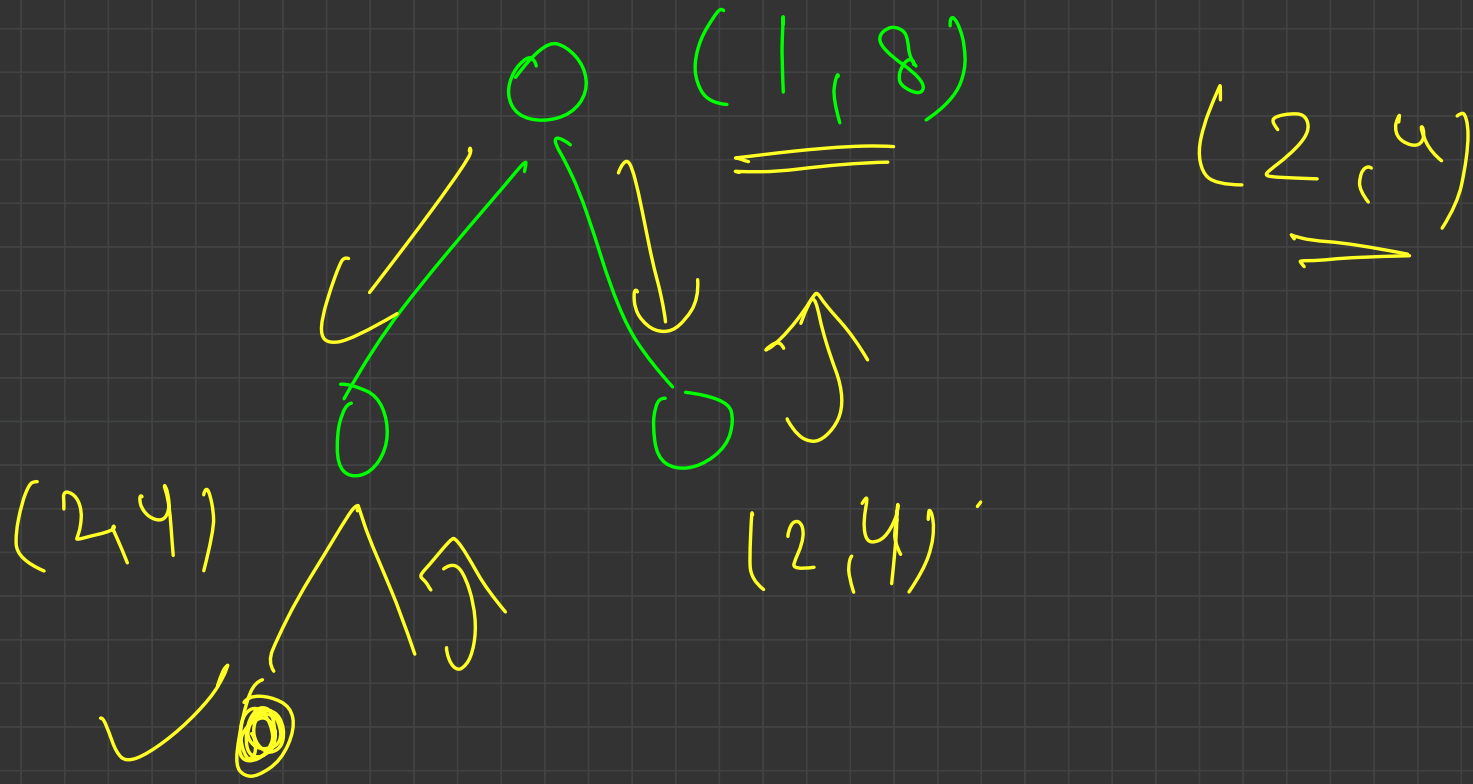


② Disjoint [ ] [ ]

just return

③ Partial overlap [[ ]]




(1,8)
(2,4)
(2,4)
(2,4)

Range update using lazy segment Tree

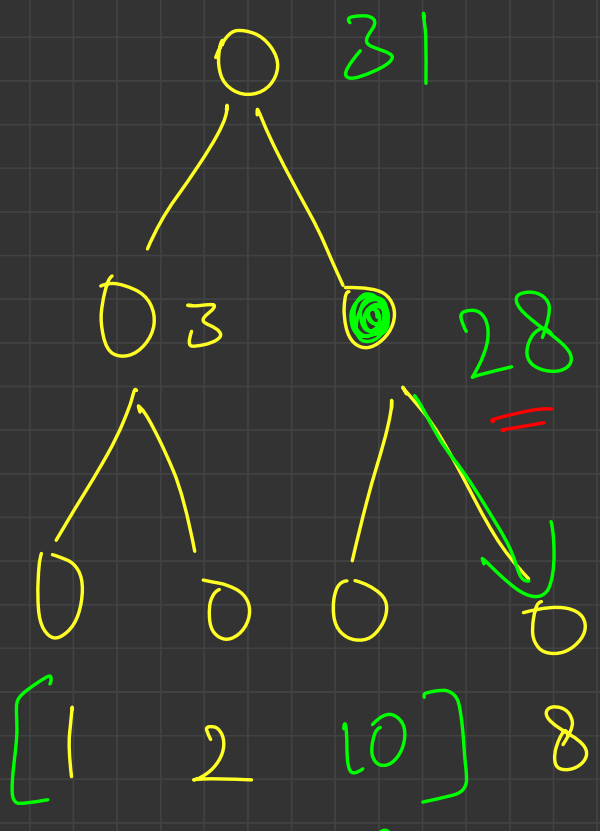
$O(\log n)$

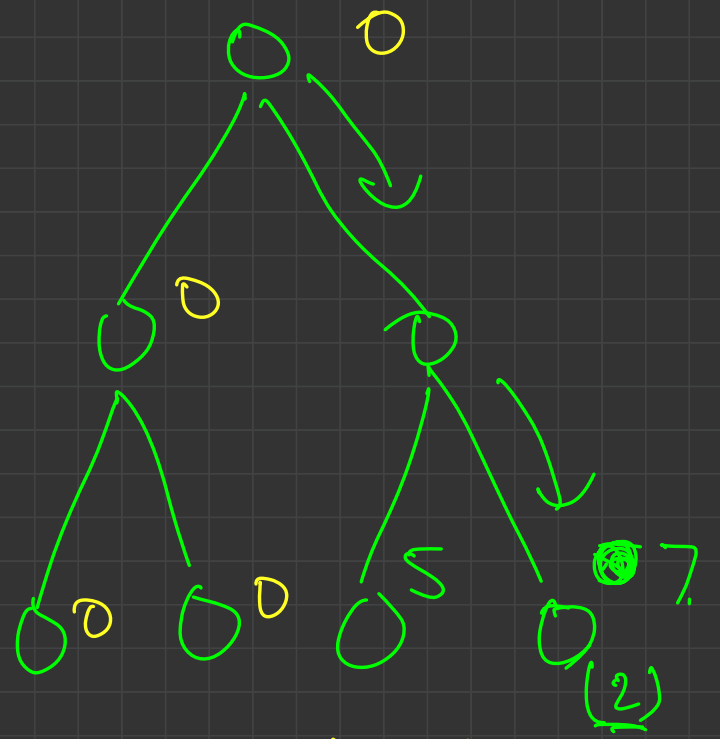
Range query $\rightarrow$ $O(\log n)$

build &

update ( )

query — check if there are any pending changes in lazy segment tree

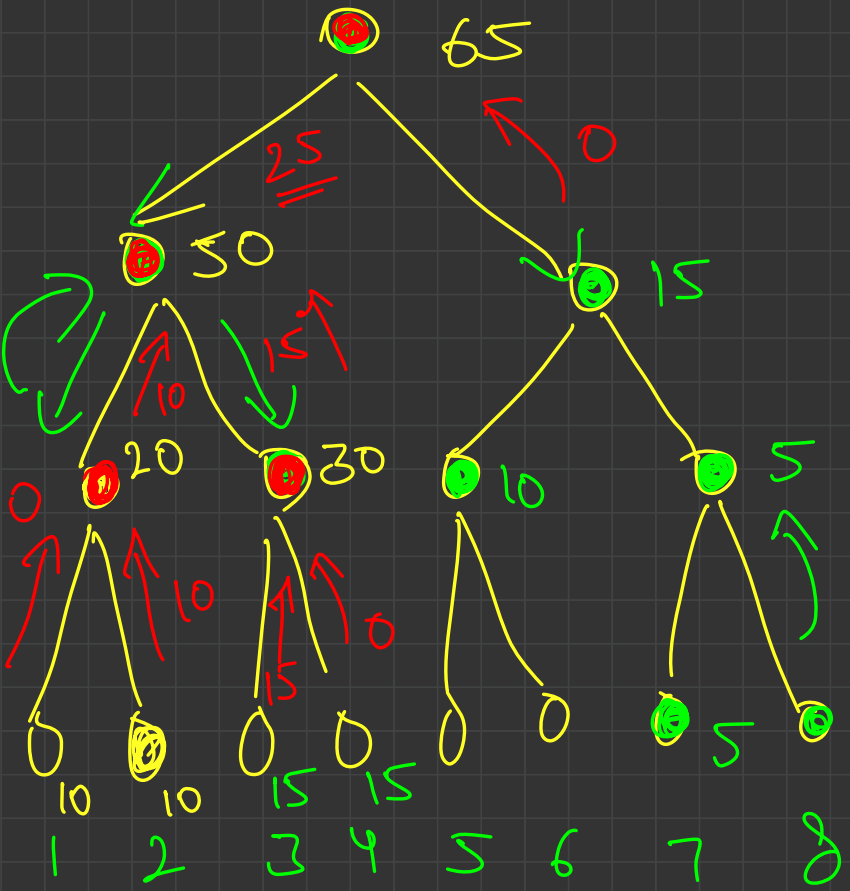




Range update (3,3), 3

Range update (3,4), 5

Range query (4,4)



Range update

(3,7) → 5

Range update

(1,4) → 10

Range query

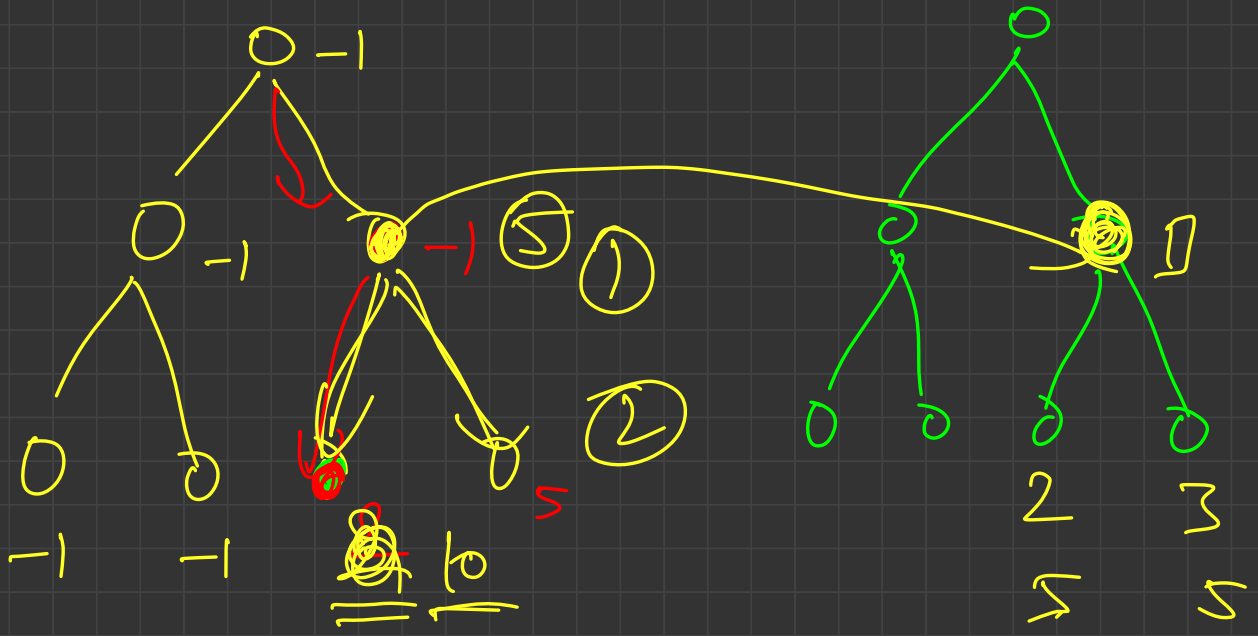
(2,3) 10

Use Case

- Perform the following 2 queries on an array:
  - 1 L R X -> Add X to all elements from L to R
  - 2 L R -> Return the sum of all elements from L to R

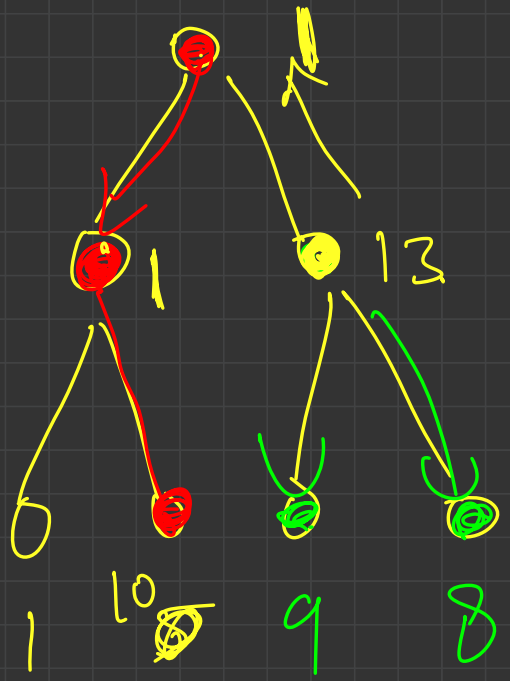
- Perform the following 2 queries on an array:
  - 1 L R X -> Add X to all elements from L to R
  - 2 L R -> Return the min of all elements from L to R

- Perform the following 2 queries on an array:
  - 1 L R X -> Make all elements from L to R = X
  - 2 L R -> Return the xor of all elements from L to R

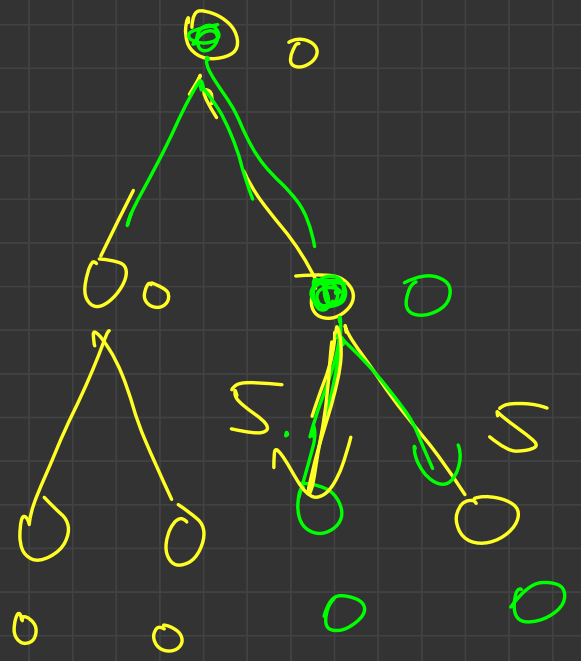


lazy[2 x index] += lazy[index]

lazy[2 x index] = lazy[index]



(5,6)



1,2 2,4

Add (3,6) → 5

Lazy Segment Tree

- **Range Updates are similar to Range Queries**
- **Same number of recursive calls for query as that for update**
- **Time Complexity for each update: O(logN)**
- **Time Complexity for each query: O(logN)**
- **Extra O(N) space for storing an additional tree**
- **Slower than normal segment tree**
- **Harder to code (Use Generic Segment Tree Code)**
- **Nothing fancy if you understand it properly - Super intuitive**