Segment Trees -2

Question: Bob and Queries

→ Given array of Length N with all 0's

3 Queries

1) Update1 (i) → A[i] → 2*A[i]+1 ✓
2) Update2 (i) → A[i] → A[i]/2 ✓
3) Range (l, r) → Concatenate Binary Rep of all no-s in the range and return the no. of 1's

1 → 2*1+1

[0, n-1]

N = 5

A = 1 3 1 0 0 (indices 0 1 2 3 4)

01 01 01

011101 00 ⇒ 010101

U1 (0)
U1 (1)
U1 (2) → A[2] → 2*A[2]+1 → A[2] =
A[2] → 1*0+1 = 1
U1 (2)

Range [0,2] → 3
U1 (1) ⇒
Range [1,3] → 3

Brute Force:

U1 → A[i] = 2*A[i]+1; } O(1)
U2 → A[i] = A[i]/2;

Range[l, r] → O(n * 32) → O(n)

O(n * log(max(A)))

n.

logn ?

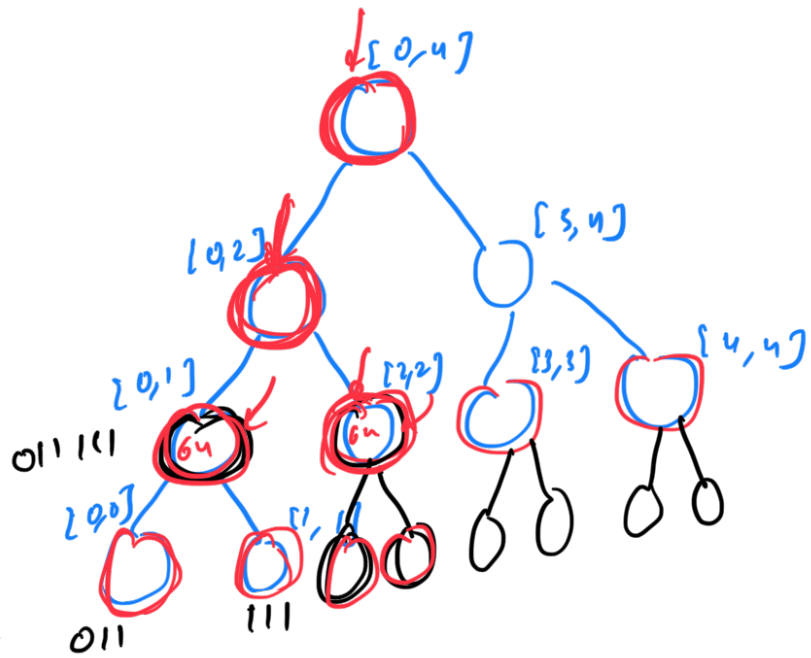
worst case T.C

Approach 2:

What should I store in any node of seg tree?

A =	3	7	6	5	9
	0	1	2	3	4

Integer → 32 bits



O(H)

We 2 strings S_1, S_2 $\quad O(l_1 + l_2)$

l_1 $\quad l_2$

operations = $64 + 128 + 256 + 512$

$64(1 + 2 + 4 + \dots)$ logn.

GP

$\cancel{2}^{\log_2 n} = n$

operations = $\dfrac{a(r^t - 1)}{r - 1} = \dfrac{1(2^{\log_2 n} - 1)}{2 - 1}$

operations = $n - 1 = O(n)$

operations

$\frac{}{2-1}$

Range Query : $O(n)$

Approaches:

We only want the no of set bits.

Lets store no. of sets bits of array in Segment Tree.

Observations:

U1(i) → A[i] → 2*A[i]+1

0 → 1 → 3 → 7 → 15 → 31 → 63

0 1 11 111 1111 11111 111111

the no. of set bits is increasing by 1

U2(i) → A[i] → A[i]/2

15 → 7 → 3 → 1 → 0

1111 111 11 1 0

No. of set bits is decreasing by 1

1) What should I store at any node of Segment Tree?

[l,r]

No. of set bits of array elements in [l,r]

Children

2) How to compute parent using children

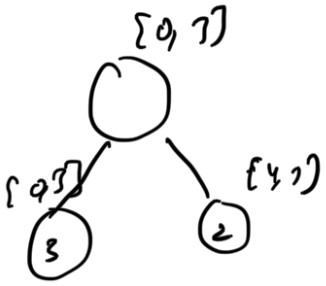
Parent → Sum of children

A = [0 0 0 0 0 0]

B = [0 0 0 0 0 0] N

No. of set bits

Segment tree on the array

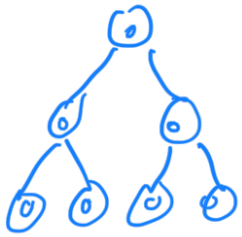


1) update1 (i)

Increase B[i] by 1

2) update2 (i)

decrease B[i] by 1



3) Range [l, r]

Range sum Query in the [l, r]

A = [3 7 15 31 9]

Set_bits = [0 0 0 1 0 0 0]

0 1 2 3 4 5 6

Construct Segment Tree. → O(n)

U1 (5) :

U2 (4) :

T.C: U1 : O(logn)

U2 : O(logn)

Range : O(logn)

O(log(max))

Question: Binary Updates

Given an array of size N with all 0

Queries

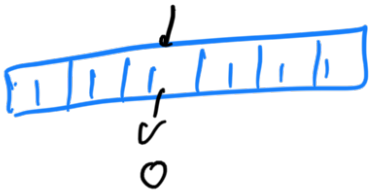
1) update(i) → makes A[i] = 0

2) k^{th} One (k) → Index of kth One in the array

N = 5

A = 1 1 0 0 1

0 1 2 3 4



k^{th} One (3) → 2

update(2) → ✓

k^{th} One (3) → 3

update(3)

k^{th} One (3) → 4

Brute Force:

update : A[i] = 0 → O(1)

k^{th} One (k) → Iterate and get the kth one O(n)

Approach2:

[0 . . 3] N = 8

A = 1 0 0 1 0 0 1 1

0 1 2 3 4 5 6 7

k = 2 ⇒ 3

→ Find index 'i' such that Ones(0 . . i) == k

Answer?

i = 4 Ones(0 . . 4)? = 2

i = 5 Ones(0 . . . 5)? = 2

i = 3, 4, 5

the min 'i'

choose the

	0	1	2	3	4	5	6	7
A =	1	0	0	1	0	0	1	1

K = 2 ⇒ Ans = 3

$\boxed{i = 6}$ ⇒ $\boxed{\text{Ones}(0 \ldots 6) = 3}$ > K ✗

i = 7, 8, 9 . . . .

i = 2 ⇒ Ones(0 . . . 2) = 1 < K

i = 2 ✗

i = 1, 0

K

BINARY SEARCH

i

Search space [0 . . . . . n-1]

```
low = 0
high = n-1;
while( low ≤ high){
    mid = (high + low)/2;
    if( Ones(0 . . . mid) > K)
        high = mid-1;
    if( Ones(0 . . . mid) < K)
        low = mid+1;
    else {
        ans = mid;
        high = mid-1;
    }
}
```

O(logn)

logn BS iteration

Ones(0 . . . . . i)

prefix Sum Array?

So need the No of 1's in the range (0 . . . i)

1-A operations

→ ? have some operation

Segment Tree →

sum(0 ... l) } O(log n)

update : O(log n)

	0	1	2	3	4
A =	1	1	0	1	1

N = 5

$\boxed{k = 3}$

→ update(2)

→ k^{th} One(3) →

low	high	mid	Ackers
0	4	2	Ones(0...2) = 2 (l, r)
3	4	3	Ones(0...3) = 3 ans = 3
3	2		

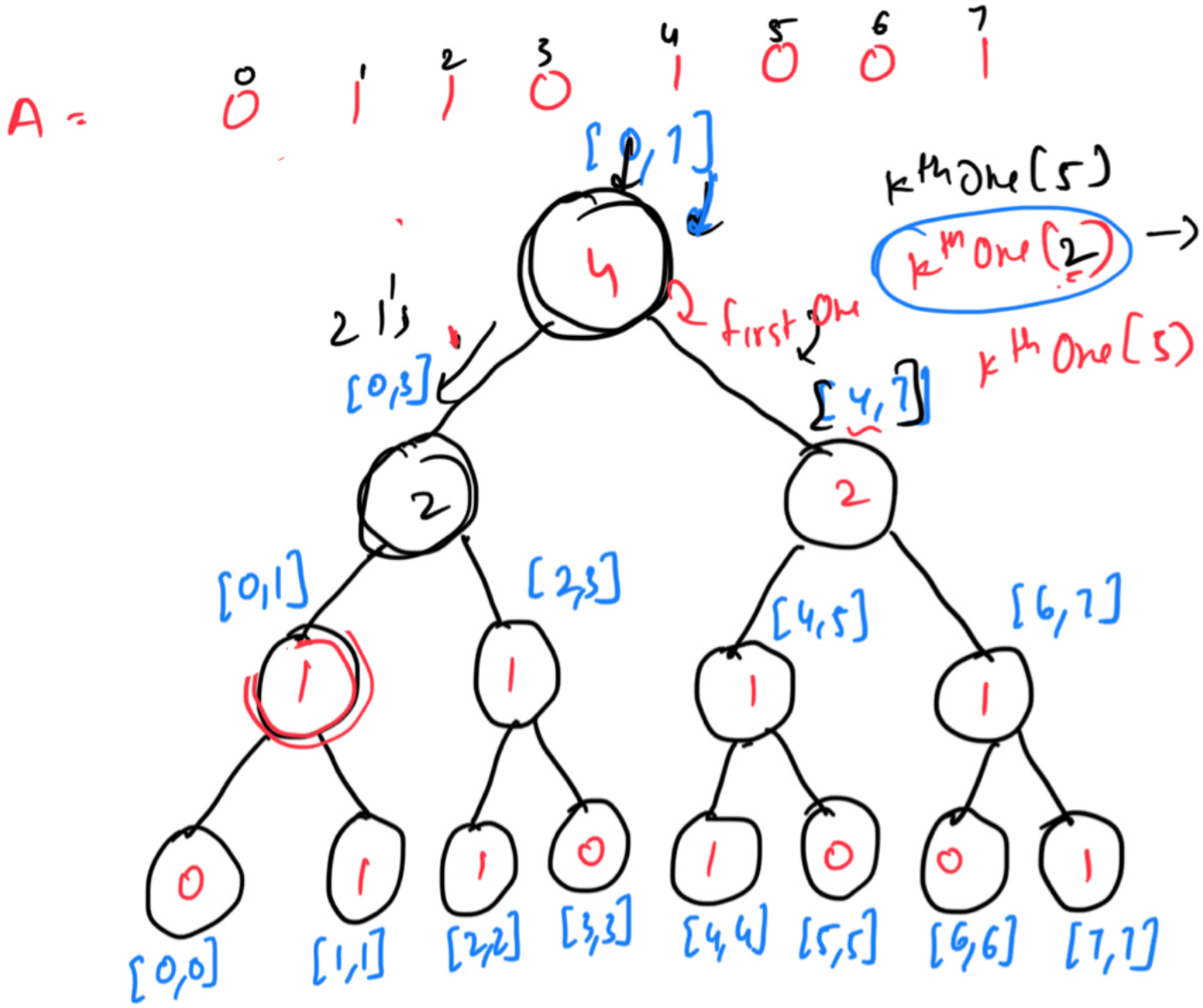
T.C : O(log n × log n) ← k^{th} one.

O(log n) ← update

Approachs:

Maintain a sum segment tree.

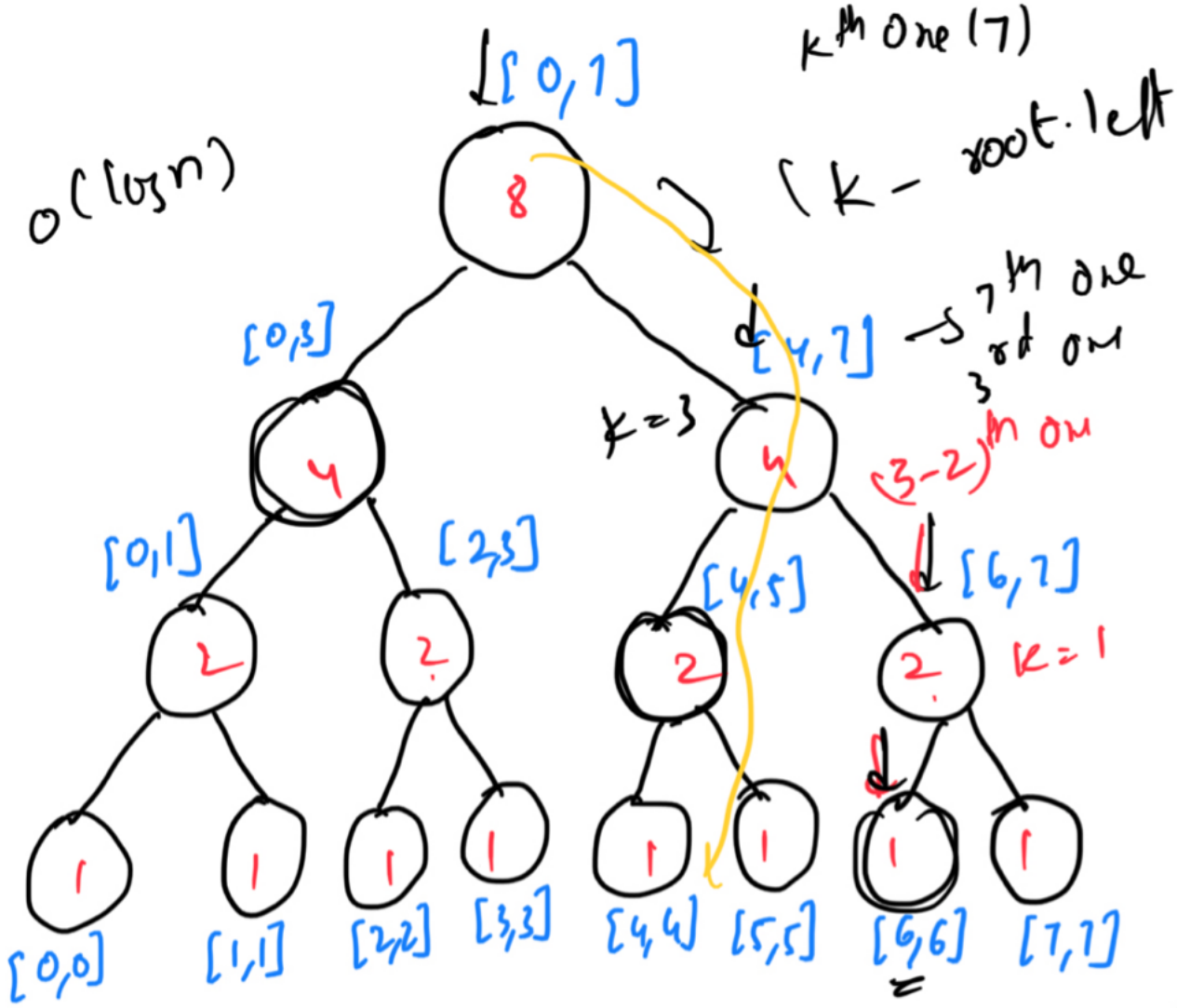
	0	1	2	3	4	5	6
A =	1	1	0	1	1	0	0



SEGMENT - TREE

A = 1 1 1 1 1 1 1 1

T.C :- O(log n)



SEGMENT - TREE

Question: Special Sum

Given an array of size N

Queries

1) update(ind, val)

→ A[ind] = val

2) Range [L, R]

$1 \cdot a[L] + 2 \cdot a[L+1] + 3 \cdot [L+2] + \dots (R-L+1) A[R]$

Range [3, 6]

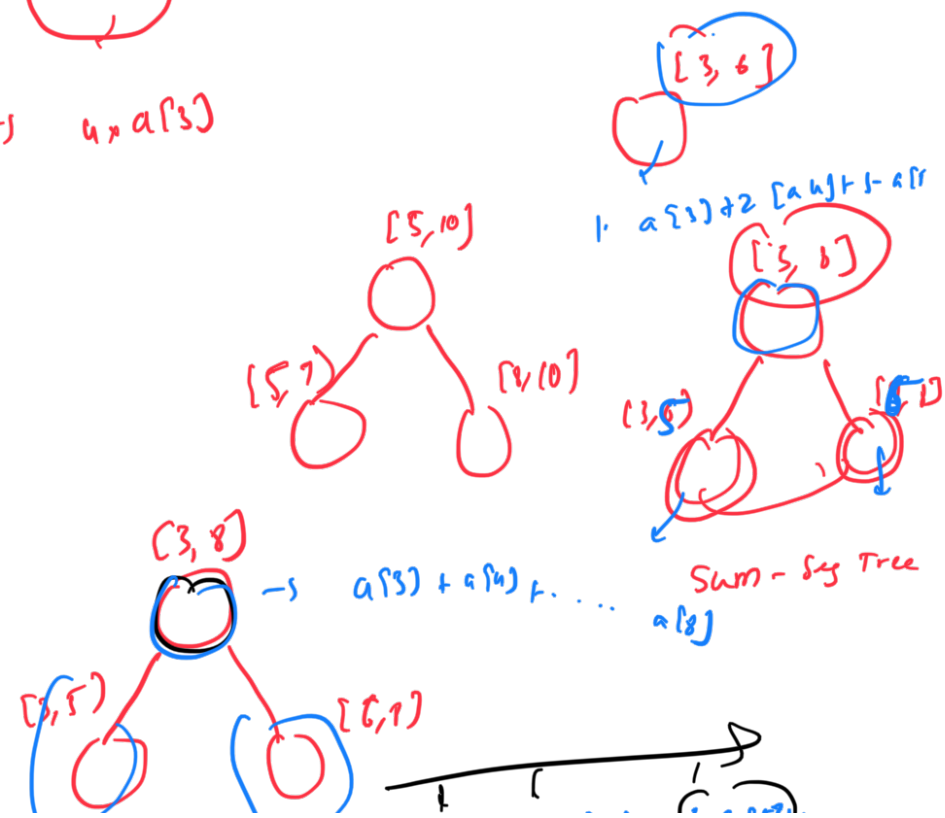
$1 \cdot a[3] + 2 \cdot a[4] + 3 \cdot a[5] + 4 \cdot a[6]$

[2, 5]

→ $1 \cdot a[2] + 2 \cdot a[3] + 3 \cdot a[4] + 4 \cdot a[5]$

[3, 6] → $1 \cdot a[3] + 2a[4] + 3 \cdot a[5] + 4 \cdot a[6]$

[0, 4] → $4 \times a[3]$



$a[3] + a[4] + \dots a[8]$

$a[4] + a[5] + \dots a[8]$

$a[5] + \dots a[8]$

$a[6] \dots a[8]$

$a[5] + 2 \cdot a[4] + 3 \cdot a[5] + \dots$

$\rightarrow a[4] + a[5] + a[6] + \dots$

$a[5] + a[6] + \dots$

$[0, 3]$ $[0, 8]$

2

$[L, R] \rightarrow 1 \cdot A[L] + 2 \cdot A[L+1] + 3 \cdot A[L+2] + \dots (R-L+1) A[R]$

$$\sum_{i=L}^{R} (i - L + 1) \cdot A[i]$$

$\Rightarrow 1 \cdot A[L] + 2 \cdot A[L+1] + 3 \cdot A[L+2] + \dots (R-L+1) A[R]$

$$\Rightarrow \sum_{i=L}^{R} i \cdot A[i] - (L-1) \sum_{i=L}^{R} A[i]$$

$i - L + 1$

$L+1 - L + 1$

$A[L] + A[L+1] + A[L+2] + \dots A[R]$

Range Sum Query

$2A[L] + (L+1) A[L+1] + (L+2) A[L+2] \dots$

(L, R)

$$\sum_{i=L}^{R} i \cdot A[i]$$

$O(\log n)$

$B[i] = i \cdot A[i]$

$L = 2$ $R = 5$

$B = \{0 \cdot A[0], 1 \cdot A[1], 2 \cdot A[2], 3 \cdot A[3], 4 \cdot A[4] \dots\}$

$\rightarrow$ Create Segment Tree on Array A $\rightarrow$ seg B

$\rightarrow$ Create new array B where $B[i] = i \cdot A[i]$

$\rightarrow$ Build one more Seg Tree on array B

$O(\log n)$ $\quad$ SegB $\quad$ $O(\log n)$

$$\text{Range}(L, R) = \underline{\text{Range}(\text{segB}, L, R)} - (L-1)\,\text{Range}(\text{segA}, 1, R)$$

T.C of Range Query : $O(\log n)$

Update (ind, val)

update (segA, ind, val) $\log N$

update (segB, ind, ~~val~~) $\log N$

ind val

$\boxed{O(\log N)}$

Question: Segment Tree ?

1) Update (ind, val)
   A[ind] = val

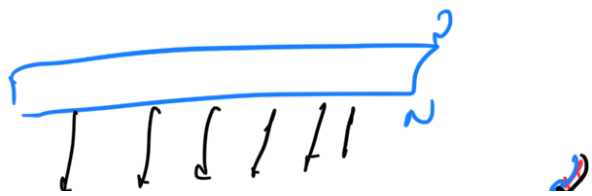
2) RangeSum (L, R)
   sum of A[L....R]

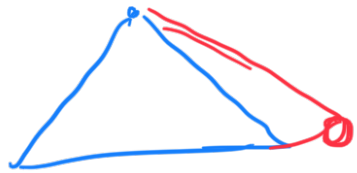
3) Append (val)
   Append this element at end of the array

4) Delete (ind)
   Delete the element at index 'ind'

Appends

A :





A :

→ O(N)

Approach 1 :

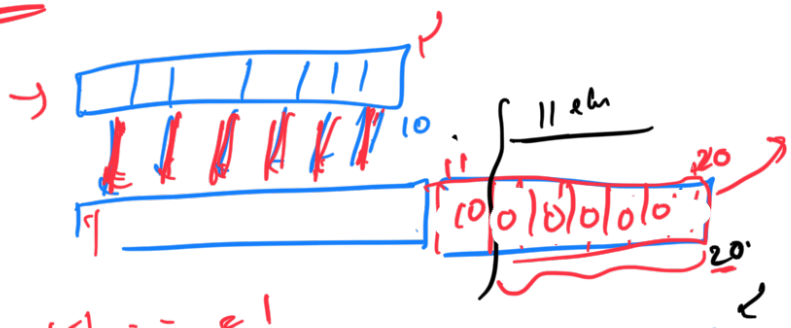
→ Create a new array of size N+1 → O(N)

→ Rebuild a segment tree → O(N)

T.C : O(N) + O(N)

Append = $\boxed{O(N)}$

Approach 2 :



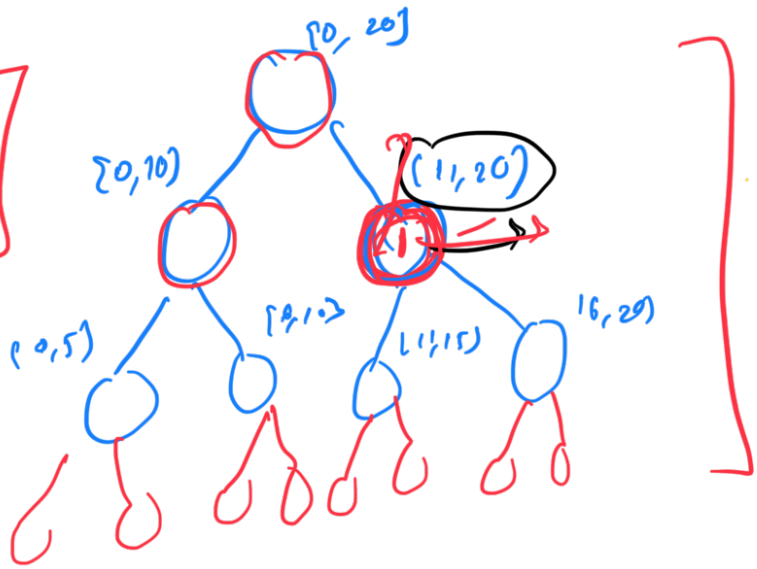
Create a segTree of this New array

Sum(11 ... 20)

10 - 1 - 1 - 1 - 1 - - - 1

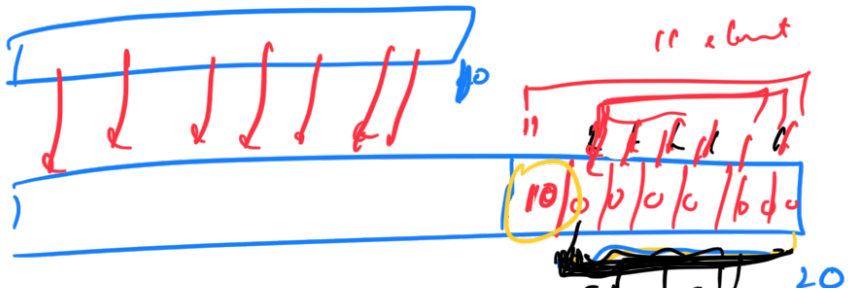
9 times

Sum(11, 12, 13 ... 20)



A =

A' =



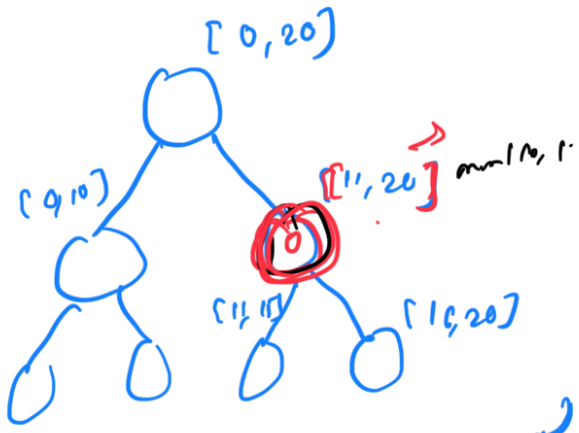
Append(-10)

Minimum

Sum → 0

Min → INT_MAX

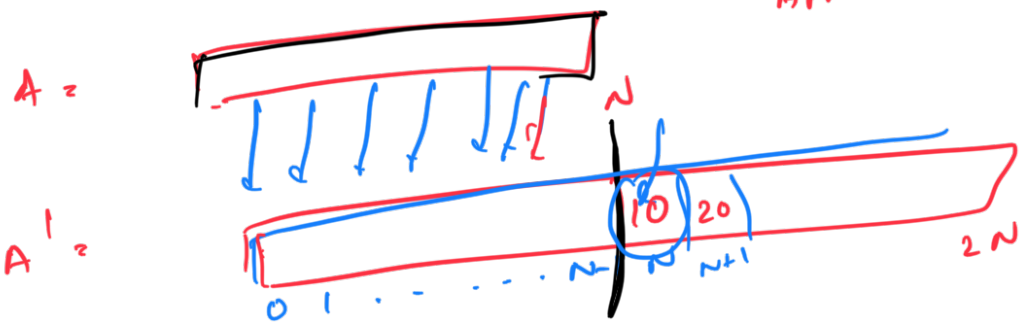
Max → INT_MIN



Append(100)

$\boxed{O(N)}$

A' [N+1] = 20



Append(X)

A' [N] = X

update (N, X)

Append(10)
Append(20)
update (N+1, 20)

Amortized O(logN)

Very Few operations → O(n) + O(logn)

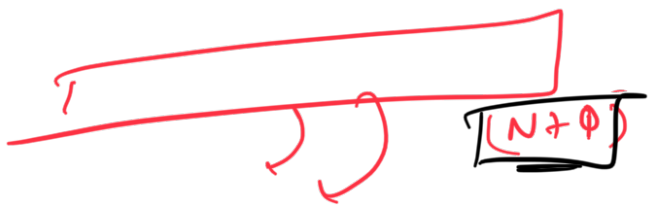
Most operations → O(logn)

Amortized O(logn)

Approaches:

1. N, Q ≤ 10^5

A = 2 × 10^5



Delete (ind)

Delete Operation :

	0	1	2	3	4	5
A =	5	3	6	1	3	2

sum (0, 4) = 5+3+6+1+3 = 18

Delete (2)

Approach1 :

→ Delete element from Array → $\boxed{O(n)}$

→ Rebuild the Sq fn → O(n)

	0	1	2	3	4
A =	5	3	1	3	2

$\boxed{T.C: O(N)}$

Approach2 :

	0	1	2	3	4	5
A =	5	3	0	1	3	2

sum(0, 4) = 18

Del (2)

	0	1	2	3	4
A =	5	3	1	3	2

sum (0, 4) = 5+3+1+3+2 = 14

[0, 4] → [0, 5]

Task : Given index 'i' of the new array after some deletions, I have to find index 'i' in the original array which is corresponding to this

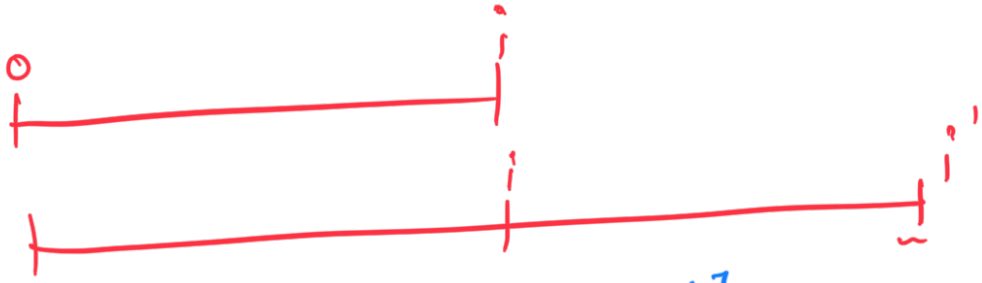
4 deletions

[0, 10] → [0, 10+4]

[0, 14] ?

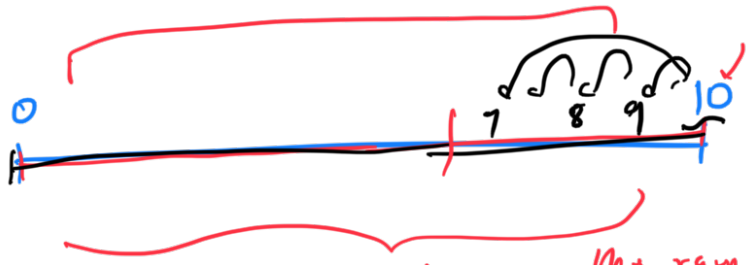
[0, 10+num_d

sum [0 ..... 10)



sum [0 ... i] → sum [0 ..... i']

$i' \geq i$



10 − 3

$i' \Leftrightarrow 10$ 3 deletions in this range ⇒ 10 − num_del.

→ Let i' be the index in the original array

$$i = \boxed{i' - \text{num_del}(0 \ldots i')}$$

i → Index in the new array

i' → Index in the original array

$$\boxed{i' - \text{num_del}(0 \ldots i') = i}$$

Requirement:

→ No. of deletes in a range (0, L) ✓

→ No. of deletes in a range can change [L, R]

Segment Tree

	0	1	2	3	4	5	6	7
del =	0	0	0	1	0	0	0	0

construct segment tree from this arrays

del[i] = 1 ⇒ if A[i] is deleted

del[i] = 0 ⇒ if A[i] is not deleted

del(3)

Range Sum (L, R) → O(log n)

update → O(log n)

$i' - num_deleted(0, i') == i$

$i' \geq i$

Range of $i' = [i, n-1]$

N = 8

	0	1	2	3	4	5	6	7
A =	5	3	0	1	0	2	4	9

Del(2), Del(5)

	0	1	2	3	4	5
A' =	5	3	1	2	4	9

sum(0 ... 3) = 11

[0, 3] → [0, 5]

[i, N-1]

$i = 3$ Range of $i' = [3, 7]$

$i' - \text{num_del}(0 \ldots i') == i$ (3)

$i' = 7, 8, 9$ ✗

Case 1: $i' = 7$

$7 - 2 = 5 > k$

Decrement

$i' = 3, 2, 1$ ✗

Case 2: $i' = 3$

$3 - \text{num}(0 \ldots 3)$

$3 - 1 = 2$ (< 3)

if($i' - \text{num_del}(0 \ldots i') > k$)

Decrement i'

else if($i' - \text{num_del}(0 \ldots i') < k$)

Increment i'

else($i' - \text{num_del}(0 \ldots i') == k$)

We Found the Mapping?

	0	1	2	3	4	5	6	7
A =	5	3	6	1	0	0	4	9

Del(4), Del(4)

	0	1	2	3	4	5
A' =	5	3	6	1	4	9

sum(0..3) = 15

[0, 3] → [0, 3]

$i = 3$ $\qquad i' = 3$

$i' = 4$

$= 4 - num_del(0 \ldots 4)$

$= 4 - 1 = \boxed{3}$

$i' = 5 = 5 - num_del(0 \ldots 5)$

$5 - 2 = 3$

```
if( i' - num_del(0....i') == k){
    ans = i';
    // Go left.
}
```

```
getMapping(i) {
    low = i;   high = n-1;
    while (low <= high) {
        mid = (high + low)/2;
        if mid - num_del(0...mid) == k {
            ans = mid;
            high = mid-1;
        }
        else if mid - num_del(0...mid) > k
            high = mid-1;
        else low = mid+1;
    }
    return ans;
}
```

$[i, N-1] \Rightarrow N$

B.S Iterations = $\log N$

T.C per iteration = $\log N$

T.C: $O((\log N)^2)$

Append:

$n = n+1;$ → $O(\log n)$

$A[n-1] = val;$

Update$(n-1, val)$ // SegTree of A

Update (ind, val)

new_ind = find Index (ind);

update (new_ind, val);

$O((\log n)^2)$

Delete (ind)

new_ind = find Index (ind); → $O(\log n^2)$

update (new_ind, 0) // update SegTree of A

del [new_ind] = 1

// update the SegTree of del

Range (L, R) → $O((\log n)^2)$

$L' =$ find Index (L);

$R' =$ find Index (R);

Range $[L', R')$ on SegTree of A