* Range Sum Query - Mutable.

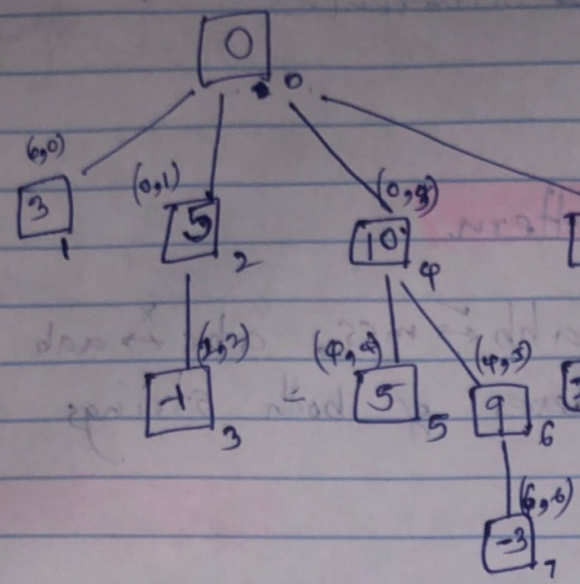
- Fenwick or Binary Indexed Tree:

In an array finding sum for a range is O(1) if we have taken cumulative sum.
But if the array is changing, then O(nm)
Cause have to take cumulative sum again & again with updatings.

Store fenwick tree → O(n)
~~O(n)~~
time to search → O(log n)
update value → O(log n)
create for first time → O(n log n)

3	2	-1	6	5	9	-3	3	7	2	3
0	1	2	3	4	5	6	7	8	9	10

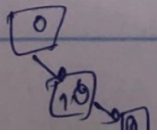
have (n+1) nodes in tree. to take parent flip right most bit



0001 → 0000
0010 → 0000
0100 → 0000
0011 → 0010
0101 → 0100
0110 → 0100
0111 → 0110
1001 → 1000
1010 → 1000
1011 → 1010

$1 \rightarrow 0+2^0$
$2 \rightarrow 0+2^1$; starting from 0, next 2 elements
$3 \rightarrow 2^1+2^0$; " " 2, " 1 "
$4 \rightarrow 0+2^2$
$5 \rightarrow 2^2+2^0$
$6 \rightarrow 2^2+2^1$
$7 \rightarrow \underbrace{2^2+2^1}_{6}+\underbrace{2^0}_{1}$; " " 6, next 1 element
$8 \rightarrow 2^3$; starting from 0, next 8 elements
$9 \rightarrow 2^3+2^0$
$10 \rightarrow 2^3+2^1$
$11 \rightarrow \underbrace{2^3+2^1}_{10}+2^0$

to take sum of (0,5) go to 6 & add to end parent.



⇒ 9+10 = 19

No: _____ Date: __/__/____

get parent

(1) take 2's complement

10 → 01010

inverse → 10101

add 1 → 10110

(2) AND it with original

01010 & 10110

= 00010

(3) Subtract from original number

01010

− 00010

01000 → 8

get Next

(1) 2's complement

10 → 01010

inverse → 10101

add 1 → 10110

(2) AND with original

01010

& 10110

00010

(3) Add to original number

01010

+ 00010

01100 → 12

4 → 25

0100

1011

1100

and 4 & 12

0100

add 4 + 4

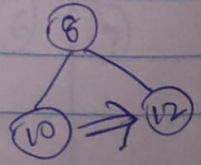
00100

00100

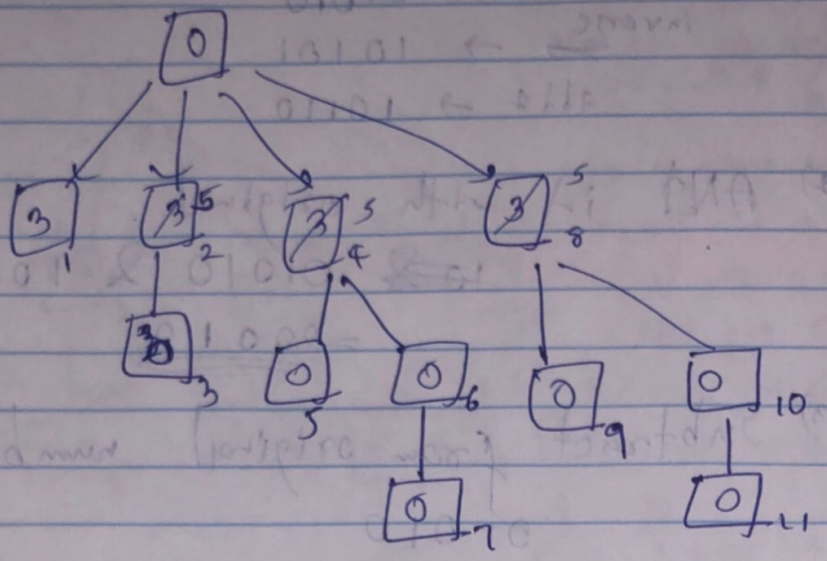
01000

10 next → 12

4 next → 8



3	2	-1	6	5	4	-3	3	7	2	3
0	1	2	3	4	5	6	7	8	9	10



~~when~~ lets take 0 → 3 as and add

(1) next → 001 → 111 → 001 → 010 = 2

(2) next → 010 → 0110 → 010 → 100 = 4

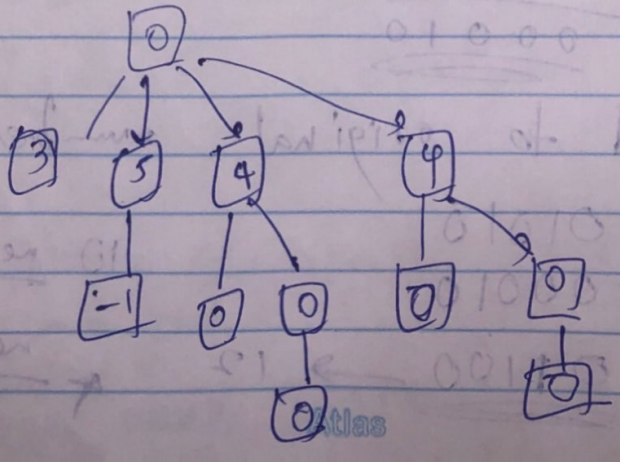
(4) next → 0100 → 1100 → 0100 → 1000 = 8

~~lets take 1 → 2~~

lets take 2 → -1

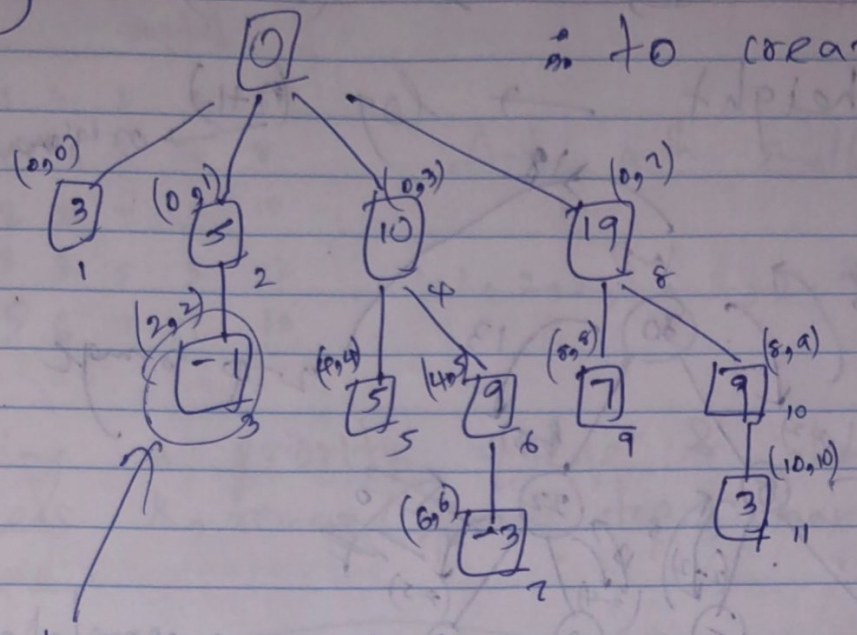
(3) next → 0011 → ~~1101~~ → 0001 → ~~0100~~ = 4

(5) next → 8



No: _____ Date: __/__/____

finally →



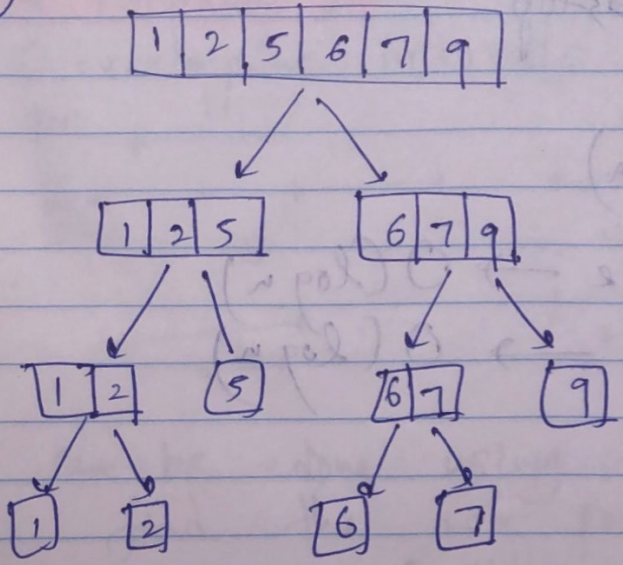
∴ to create O(nlogn)

to take sum for a range (l-r) - O(r-l)

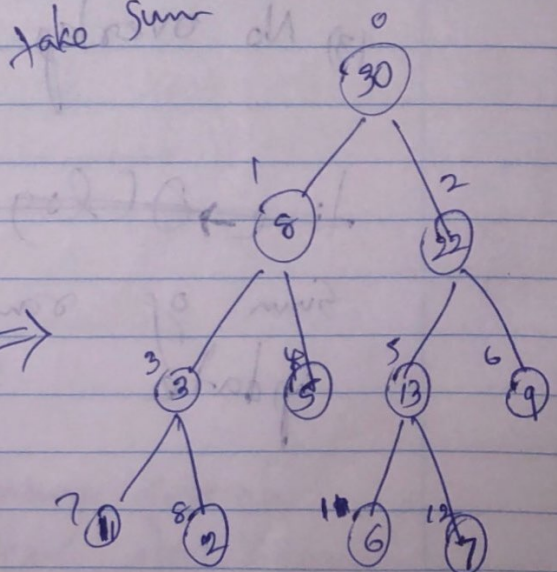
when we update this node 4, 8, ... should be updated.

node 4 → taken (0-3) so node 3, 2nd position value is also included.

- Segment Tree



take sum



parent $(n-1)/2$

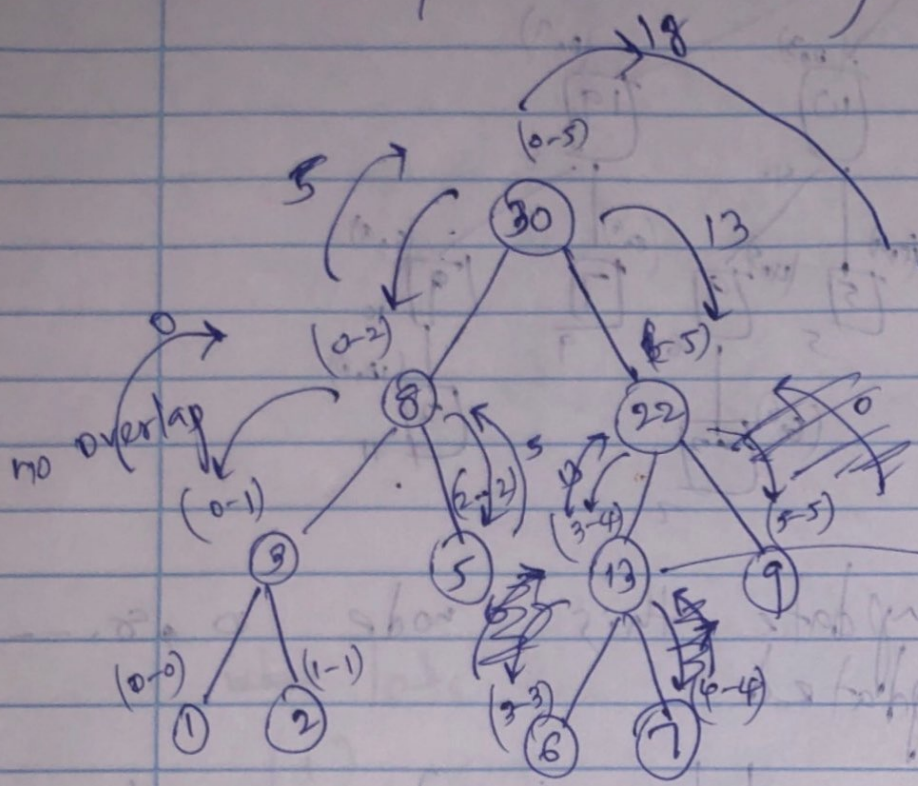
left → $2i+1$

right → $2i+2$

No: _____ Date: __/__/____

total nodes → ~~2n+1~~ (2n-1)

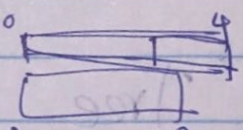
height → log (n+1) → original array



(2-4) on (3-4)
no need to go down

(1) partial overlap → look on both sides.

(2) total overlap → stop



(3) No overlap → stop

~~time → O(log n)~~

Sum of range → O(log n)

update → O(log n)