Segment Trees - 2

Question: Bob and Queries

Given an array of length N with all 0's

N = 5

3 Types of Queries

1) Update 1 (i) → A[i] → 2 * A[i] + 1
2) Update 2 (i) → A[i] → A[i] / 2
3) Range [L, R] → concatenate binary strings of all numbers and return no. of 1s

N = 5

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

A[2] = 0

A[2] → 2 * A[2] + 1
= 2 * 0 + 1 = 1

Queries

1) U 1 0

U 1 1

U 1 2

Range [0, 2] → 3

0 → 0 * 2 + 1 = 1

A[0] → 2 * A[0] + 1 → 2 * 0 + 1 = 1

A[1] → 2 * A[1] + 1
→ 2 * 1 + 1 = 3

01 01 01 → 010101 → 3

U 1 1

Range [1, 3] 3

A[1] → A[1] / 2 = 3/2 = 1

U 2 1 1

Range [0, 4] 3

A[i] = 2*A[i]+1

Brute Force

U1 : O(1)

U2 : O(1)

Range : O(n * log(maxCA))

T.C : O(n * 32)

[0, n-1]

↓

n *

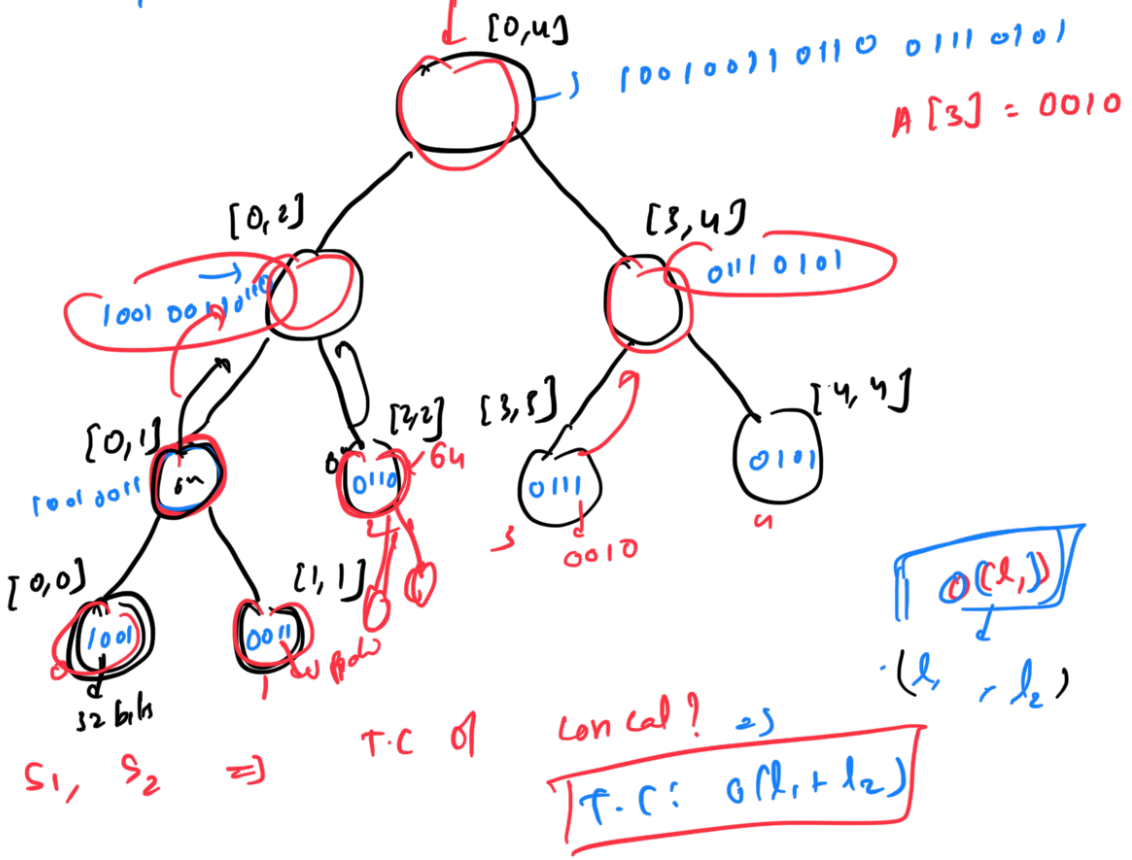
Approach2:

Segment Trees

A = [0 1 2 3 4]

1 1 1 0 0

9 3 6 7 5



1001 0011 0110 0111 0101

A[3] = 0010

S1, S2 ⇒ T.C of concat ⇒ T.C: O(l1 + l2)

O(l1)

(l1, l2)

T.C Update Query:

64 + 128 + 256 + ... ↓ logn

64 (1 + 2 + 4 + ... logn)

$\frac{a(r^n-1)}{r-1}$ ⟵ $a=1$, $r=2$, $n=\log n$

$$\frac{1\left(2^{\log_2 n}-1\right)}{2-1} = \frac{n-1}{2-1} \approx O(n)$$

Update1, Update2, Range ⇒ $O(n)$

Approaches:

Only Interested in the No. of 1's any number has!

$1 \rightarrow 10_1$ (1)

Update1 : Left Shift operator + 1

0	1	3	7	15	31	63
0	1	11	111	1111	11111	111111

$(n << 1) + 1$

No. of 1's in the binary Rep of n by 1

Update2 : Right shift $n >> 1$

31	15	7	3	1	0
11111	1111	111	11	1	0

No. of 1's in the binary Rep of n

1) Node(l, r) stores the No. of 1's in that range

2) Sum of Left and Right

Sum Segment Tree

O(log n) { U1 → Update B[i] = B[i] + 1
U2 → Update B[i] = B[i] - 1 } Segment Tree

Range : Range (1 ... N) ⇒ O(log n)

Question : Binary Update

Given N

→ We have an array of size N with all 0

N = 5

A :

0	1	2	3	4
1	1	0	0	1

1) Update (i) → A[i] = 0

2) Kth One (k) → Returns the Index of Kth One

Query

Kth One (3) → 2

Update (2) →

Kth One (3) → 3

Update (3)

Kth One (1) → 4

Kth One (5) → -1

Brute Force :

Update : O(1)

Upd...

k^{th} one : $O(n)$

1 Approach: (0's, 1's)

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

K = 2 => ind = 3

→ Find [min 'i'] such that Ones([0....i]) == K

i = 4 [0....4] = 2

i = 5 [0....5] = 2

i = 7, 8, 9.... ✗

i = 6 => One(0...6) = 3

i = 2 => Ones(0...2) = 1 i = 1, 0 ✗

Binary Search on Answer Space

Search space : [0, n-1]

→ We need Ones(0....i) & Sum(0....i) → O(n)

[l, r]

l = 0

| 1 | 0 | 0 | 1 | 0 | 0 |

1) Update (i) => A[i] = 0

2) Sum(0...i) => Sum of the range (0...i)

Segment Tree

4

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

→ Update (2)

→ K^{th} One (3)

low	high	mid
0	4	2
3	4	3
3	2	2

Actions

One (0...2) = 2

One (0...3) = 3

ans = 3

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

K^{th} One (2)

low	high	mid
0	4	2
0	1	0
1	1	1

Range Sum query on segment Tree

Actions

One (0...2) = 2

ans = 2

One (0...0) = 1

Go Right

One (0...1) = 2

ans = 1

0 1

Sum Segment Tree

Update : O(log n)

K^{th} One (K) : [0 ..... n-1] → [log n]

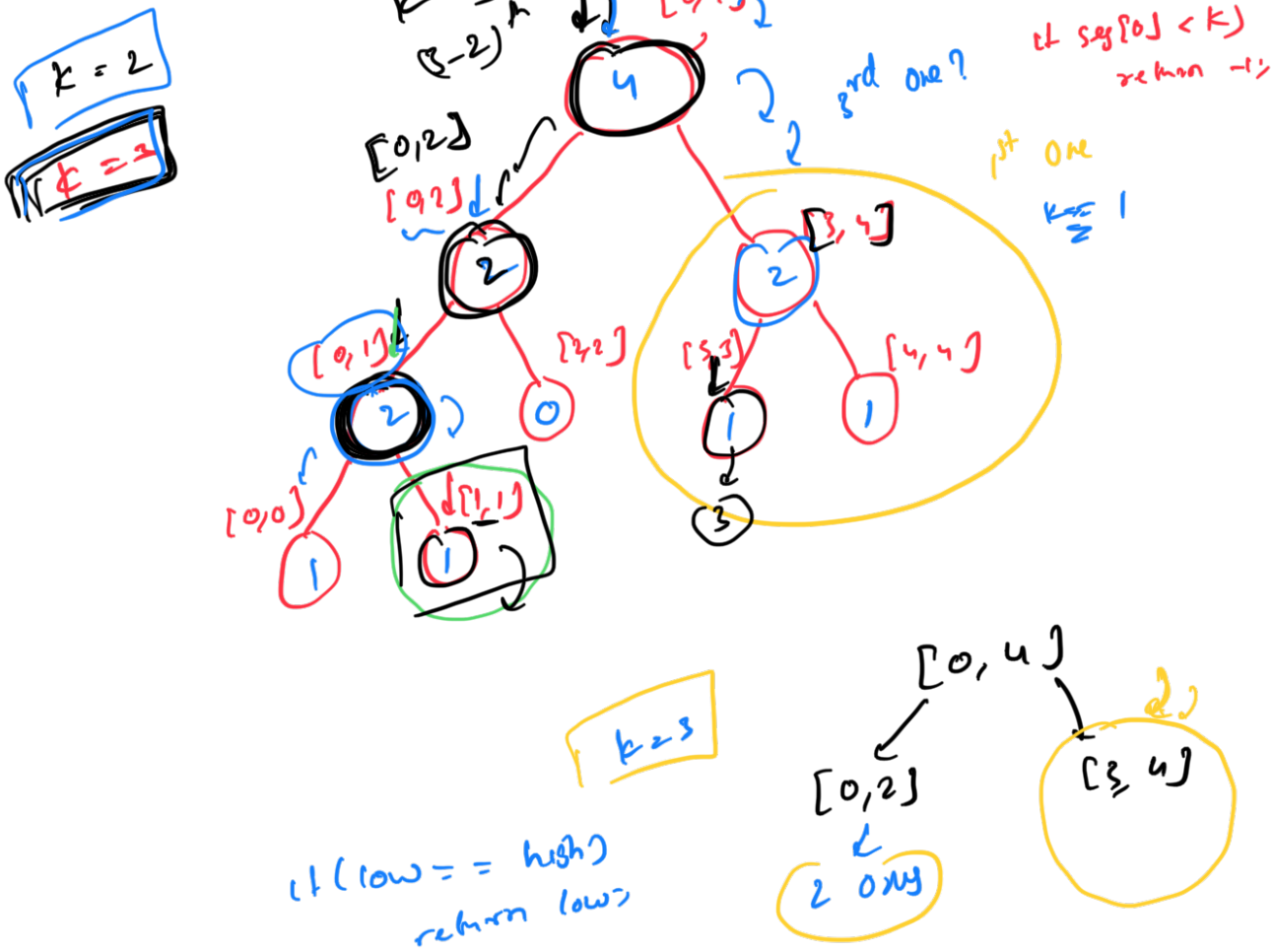
iterations = log n

T.C per iteration = log n

Approach 3 :

A =

1	1	0	1	1
0	1	2	3	4



}

T.C: O(log n)

Question: Count the Primes

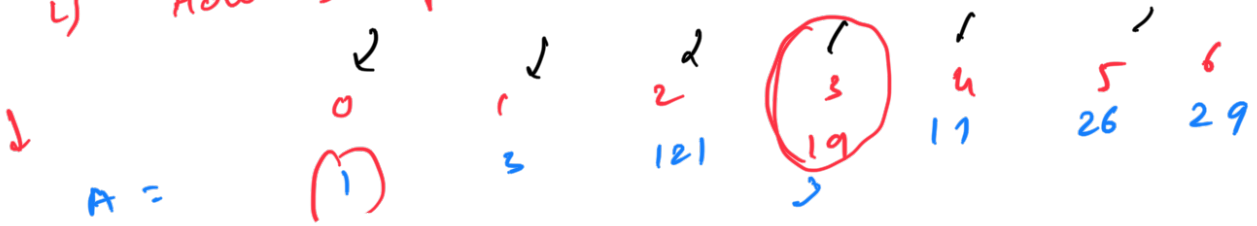
-> Given an array of integers

1) Update 2) A[i] = Val

2) Range Query: [l, r] -> No. of primes in this range.

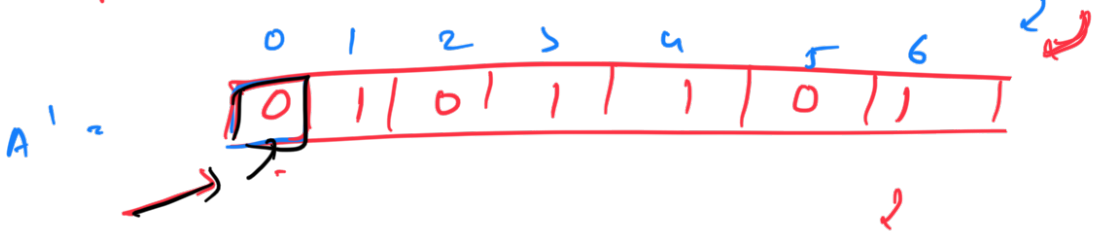
1) What should be stored at a node? -> No. of Primes in that range

4) How is parent computed? sum



Range [0, 6] = {3, 17, 293}

Update [3, 19) = Range [0, 6] => 4



T.C to check Prime: O(Ek)

T.C : $O(n \times \sqrt{max(A)} + n)$

T.C : $O(n \times \sqrt{max(A)})$

$1 \le N \le 4 \times 10^4 \times 10^5$ → $4 \times 10^4 \times \sqrt{10^6}$

$1 \le A[i] \le 10^6$ $= 4 \times 10^4 \times 10^3 = 4 \times 10^7$

Sieve $= 10^6$ ⇒ P ⇒ $O(P \log\log P)$

$O(10^6 \log\log 10^6) \approx 4 \times 10^6$

$\log_2 10^6 \approx 6 \times 3 = 18$

$\log_2^{18} \approx 4$

1) Sieve : $max(A) \log\log(max(A))$
2) Buildng : $O(n)$
3) Update : $O(\log n)$
4) Range : $O(\log n)$

(i, val)

A[i] = 1
A[i].

Question: Power of 3

Given a binary string (0's and 1's)

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

(001) % 3

Queries:

1) update (i) ⇒ A[i] = 1
2) Range (l, r) ⇒ (Decimal value of S[l...r]) % 3

0 1 2 3 4

S = 1 1 0 1 0

Queries

1) Rank (2, 4) → 010 = (2)%3 = 2

2) Rank [2, 3) = 1%3 = 1

3) Update (1)

4) Rank(0, 2) = 110 ⇒ 6%3 = 0

5) Rank(0, 3) = 1101 ⇒ 13%3 = 1

Brute Force

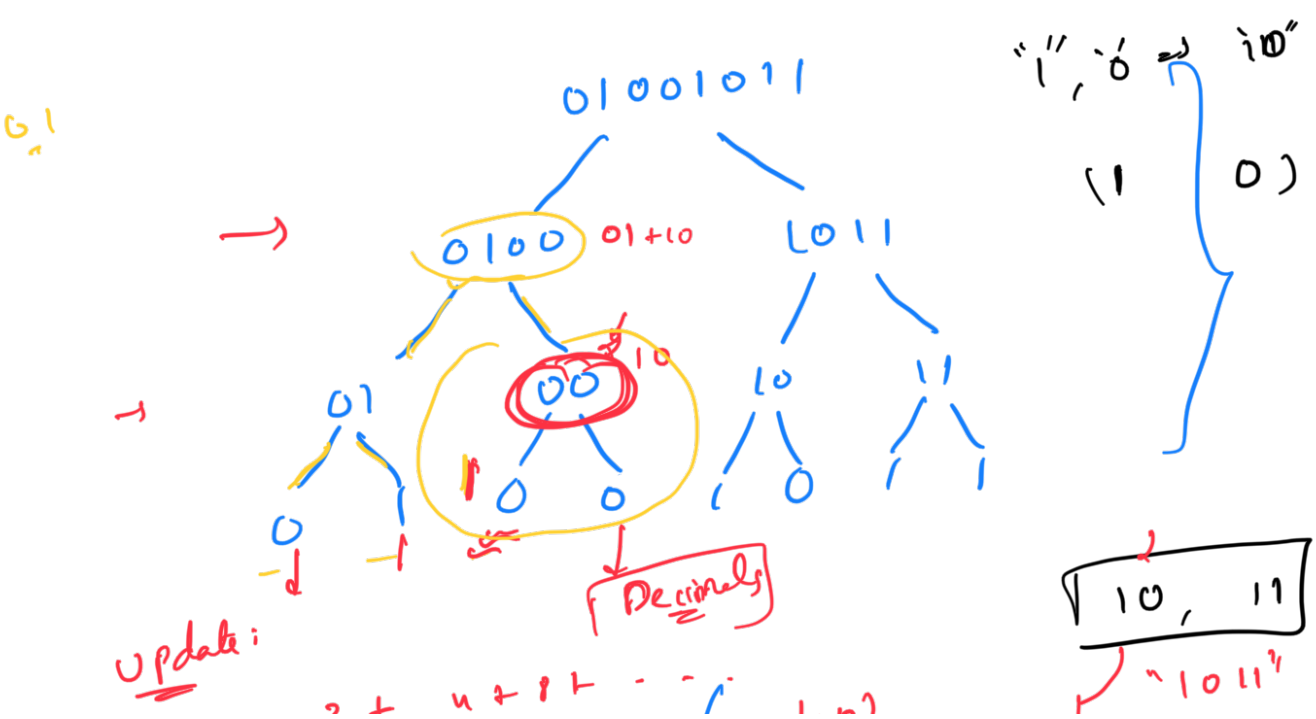
update : O(1)

Rank[l...r] : O(n)

Approach 2 :

Store the Binary Rep

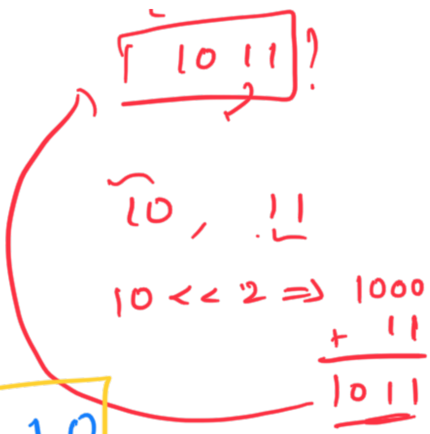
0	1	2	3	4	5	6	7
0	1	0	0	1	0	1	1



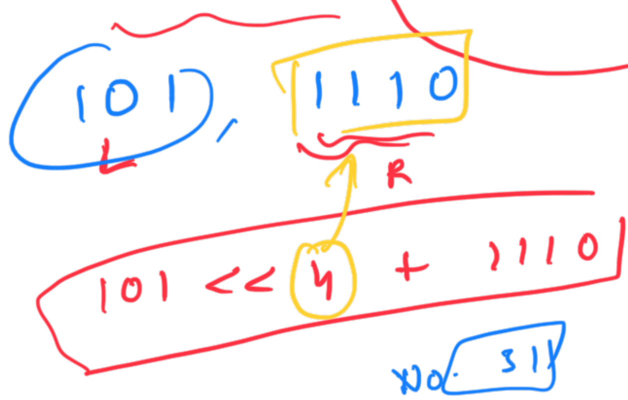
2(1+2+4 ...

-> O(n)

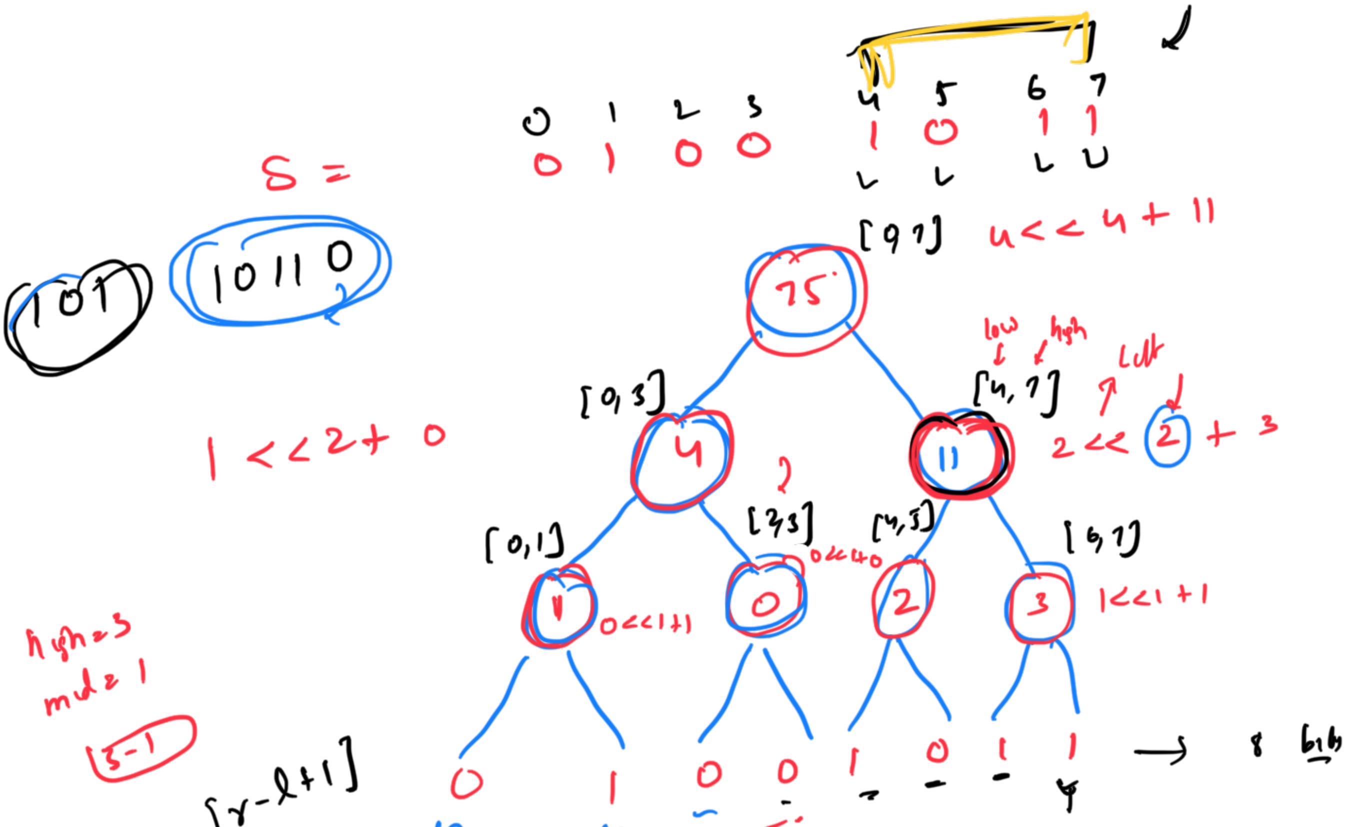
Range Query -> O(n)



Approach:



Store Decimal Value?



[l, r] => [r-l+1]

r = high

l = mid+1

high-(mid+1)+1 = high-mid-1+1 = high-mid

1) storing at Nodes?

storing decimal value of (A[l....r]) %3

2) Parent val ?

seg[i] = ((seg[2*i+1] << (high-mid)%3) %3 + (seg[2*i+2]%3)%3

seg[i] = (seg[2i+1] << (high-mid)%3 + seg[2i+2]%3)%3

seg[i] = ((seg[2i+1] . 2^(high-mid))%3 + seg[2i+2]%3)%3

(a.b)%3 = (a%3 . b%3)%3

seg[i] = (((seg[2i+1]%3) . (2^(high-mid)%3))%3 + seg[2i+2]%3)%3

(3-1) (3-1)

Powers of 2

Powers (2)%3

%3

Observations

$2^1 \% 3 = 2$	$2^5 \% 3 = 2$
$2^2 \% 3 = 1$	$2^6 \% 3 = 1$
$2^3 \% 3 = 2$	
$2^4 \% 3 = 1$	

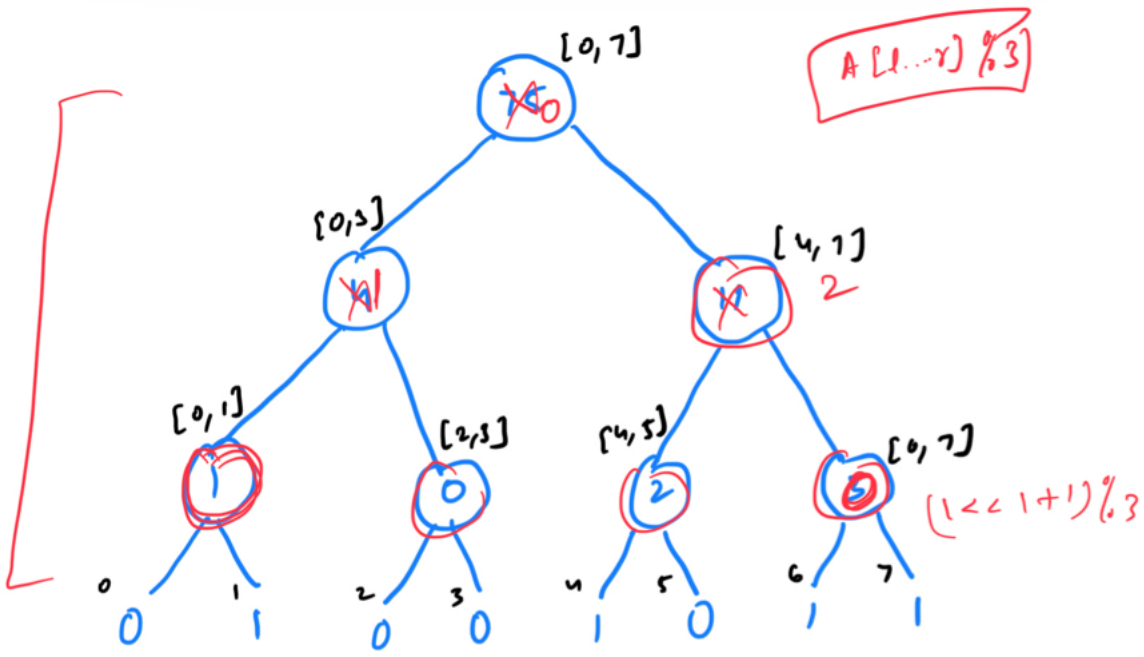
if (high-mid) is even

1

else

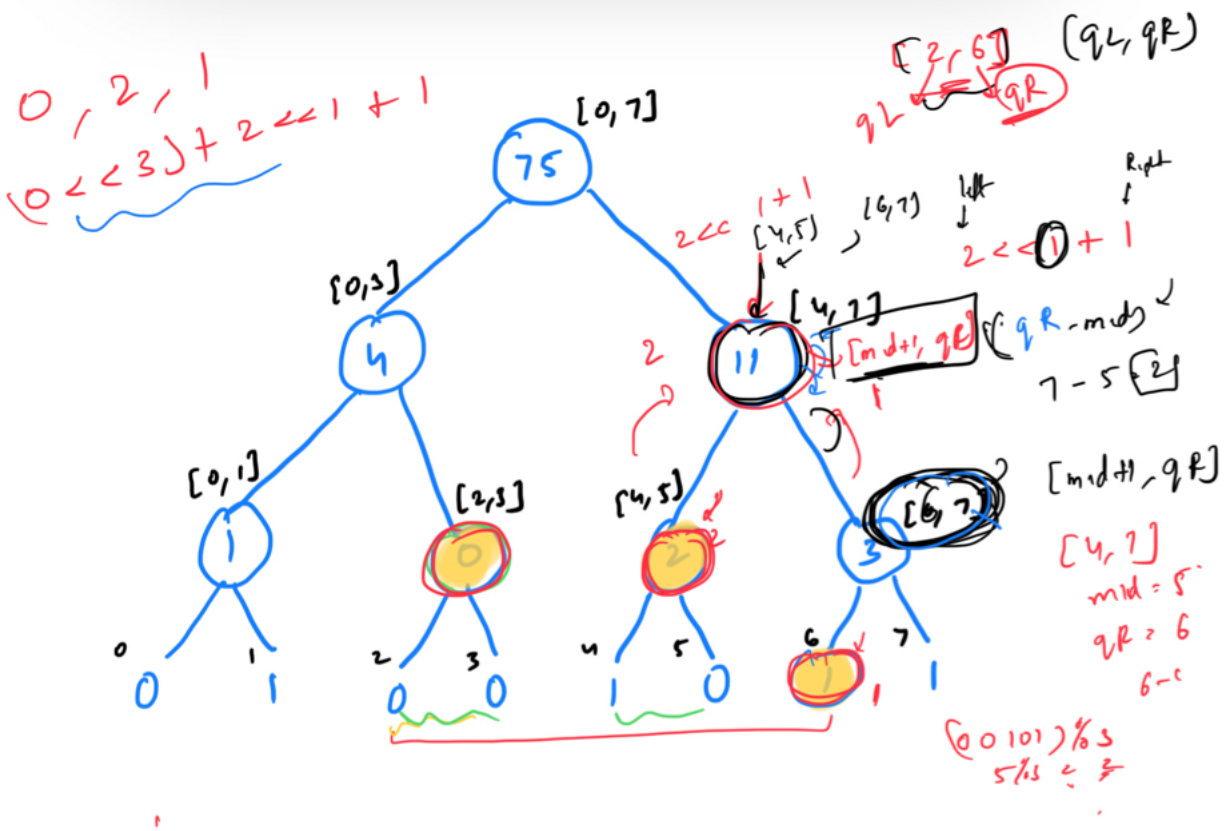
2.

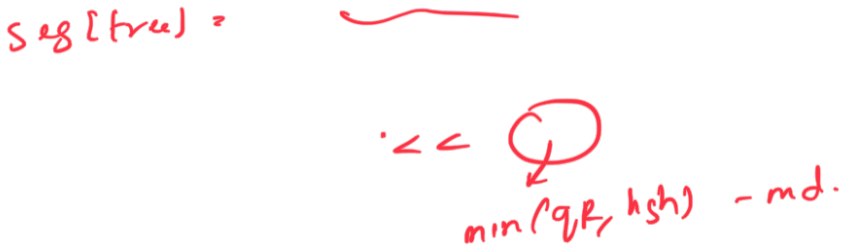
A = 0 1 0 0 1 0 1 1



Builds → Done.

Range Query:




seg[true] =
<<
min(qR, hsh) - md.