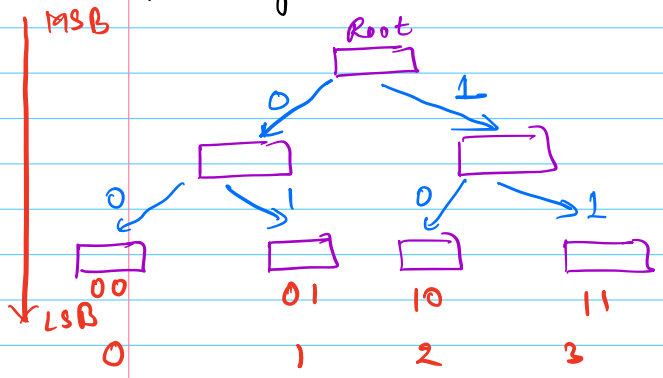
14th august 23

Tries - 2

Trie of bits



Numbers → 0 to $(2^H - 1)$

Numbers → 0 to 15

Height → 4 $(0 - 2^4 - 1)$

Numbers → 0 to 31

Height → 5 $(0 - 2^5 - 1)$

Q) Given an integer array A, find the max. value of A[i] ^ A[j] ∀ (i,j) pairs.

A : [3 (0) 5 (1) 2 (2)]

3 ^ 5 = 011 ^ 101 = 110 → 6

3 ^ 2 = 011 ^ 010 = 001 → 1

5 ^ 2 → 101 ^ 010 = 111 → 7 (Ans)

A = [9 (0) 8 (1) 10 (2) 7 (3)]

8 ^ 7 ⇒ 1000 ^ 0111 = 1111 ⇒ 15 (Ans)

Bruteforce → ∀ i, j check A[i] ^ A[j] & calculate max.

TC: $O(N^2)$ SC: $O(1)$

15 → 1 1 1 1 ^ 7 → 0 1 1 1 = 1 0 0 0 → 8

15 → 1 1 1 1 ^ 11 → 1 0 1 1 = 0 1 0 0 → 4

∴ To maximize the number MSB (leftmost) should be set (1)

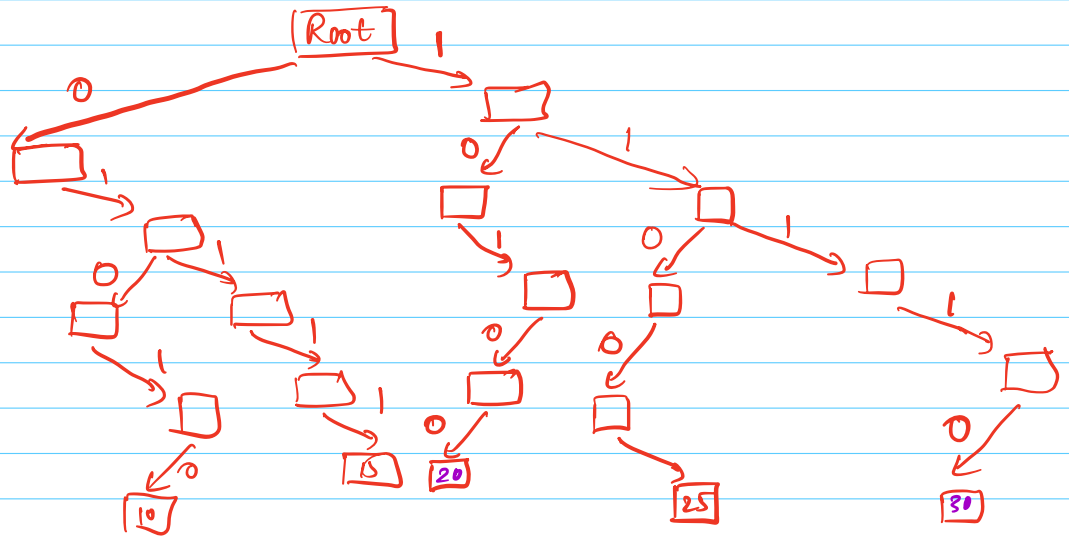
Traversing bits from MSB to LSB → Trie of bits.

A[i] <= 10^9 → Height of trie = 30

$(2^{30}-1) \geq 10^9$

A = [20 (0) 30 (1) 15 (2) 25 (3) 10 (4) 5 (5)]

	4	3	2	1	0	
20 →	1	0	1	0	0	
30 →	1	1	1	1	0	^20 = 01010 ⇒ 10
15 →	0	1	1	1	1	^20 = 11011 ⇒ 27
25 →	1	1	0	0	1	^15 ⇒ 10110 ⇒ 22
10 →	0	1	0	1	0	^20 ⇒ 11110 ⇒ 30
5 →	0	0	1	0	1	^25 ⇒ 11100 ⇒ 28



Steps :> Find max XOR for A[i]
2> Insert A[i]

```
int find (root, X) {
    temp = root;
    for i -> 29 -> 0 {
        b = (X >> i) & 1   // ith bit of X.
        t = 1 - b   // 0 -> 1, 1 -> 0  { 1 ^ b }
        if (temp.child[t]) {
            temp = temp.child[t];
        }
        else {
            temp = temp.child[b];
        }
    }
    return temp.data ^ X;
}
```

TC: O(30) ≈ O(1)

Overall TC: O(N) , SC: O(N * 30) ≈ O(N)

```
main () {
    for ∀ nos {
        cur = find (root, num);
        ans = max(ans, cur);
        insert (root, num);
    }
}
```

Q2 Given an integer array A, find subarray with max XOR value.

A = [4 6 1] (0 1 2)

4	⇒ 4
4^6	⇒ 2
4^6^1	⇒ 3
6	⇒ 6
6^1	⇒ 7 (Ans)
1	⇒ 1

Bruteforce ⇒ # subarrays

calculate XOR value & store max as answer.

TC: $O(N^3) \Rightarrow O(N^2)$, SC: O(1)

a^a = 0

a^b^a = b

subarray XOR ⇒ i-j = A[i]^A[i+1]^A[i+2]^.... ^A[j-1]^A[j]

P[j] = A[0]^A[1]^A[2]^.....A[i-1]^A[i] A[j]

P[i-1] = A[0]^A[1]^...A[i-2]^A[i-1].

A[i]^A[i+1]^... A[j]

subarray XOR i-j = P[j]^P[i-1].

P[i] = P[i-1]^A[i]

A = [1 4 2 6] (0 1 2 3)

P = [1 5 7 1] (0 1 2 3) → max XOR pair ⇒ answer.

subarray XOR i-j = P[j]^P[i-1] , i>0

P[j] , i=0

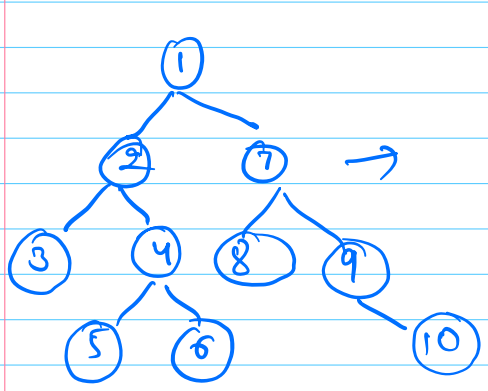
Ans ⇒ max(max of P[i], max xor pair in P[]) → prefix XOR.

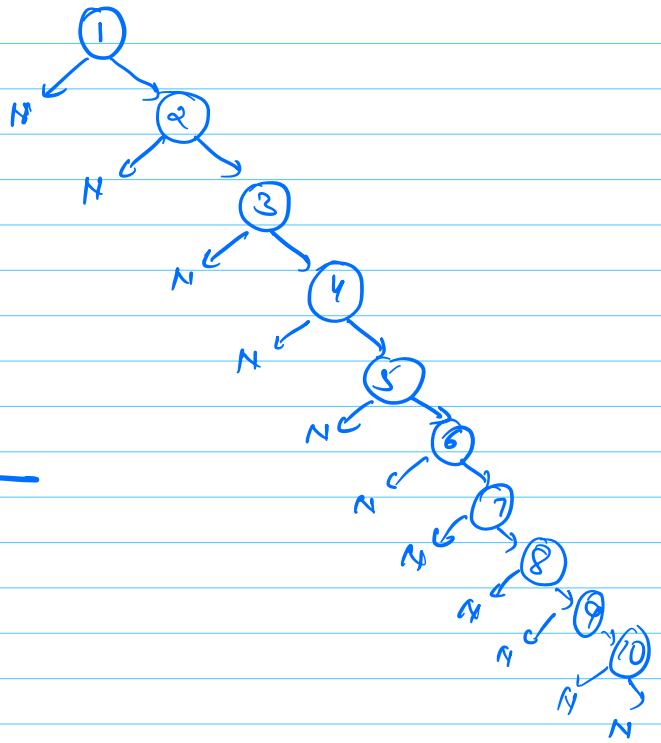
prefix Sum ⇒ P[i] = P[i-1] + A[i]
prefix XOR ⇒ P[i] = P[i-1] ^ A[i]

TC: O(N + N) ≈ O(N)
SC: O(N + N) ≈ O(N)

Q ≡ Flatten the given binary tree to linked list in preorder manner, such that right child becomes next & left child of nodes should be null.

Node
left
right



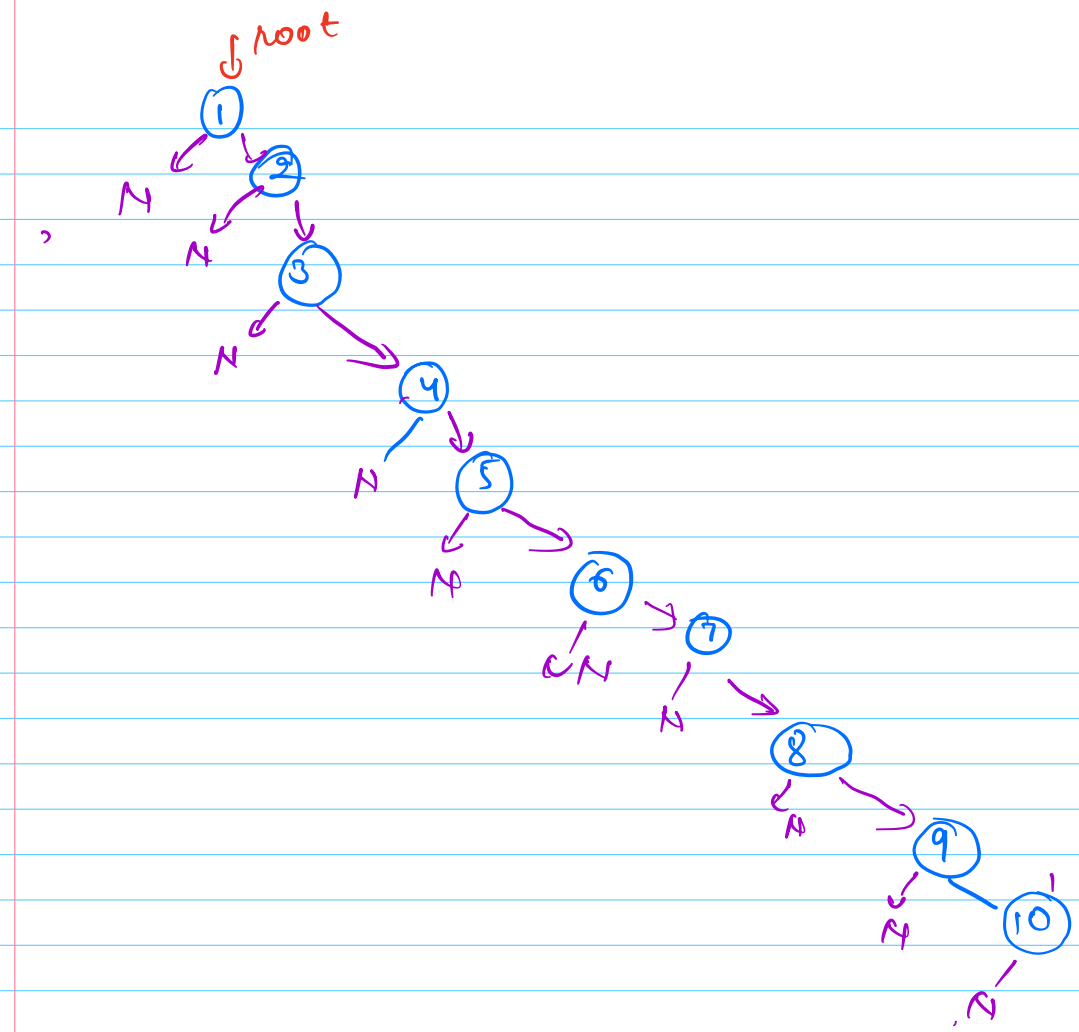


Meet at 8:43 am IST

→ {Head, tail)

```
pair flatten(root) {

    if(root == null) return {null, null};

    L = flatten(root.left);
    R = flatten(root.right);

    root.left = null;

    if (L.head == NULL && R.head == NULL) {
        return {root, root};
    }
    else if (L.head == null) {
        root.right = R.head;
        return {root, R.tail};
    }
    else if (R.head == null) {
        root.right = L.head;

        return {root, L.tail};
    }
    else {
        root.right = L.head;
        L.tail.right = R.head;

        return {root, R.tail};
    }
}
```

TC: O(N)
SC: O(H)


root
1
2
3
4
5
6
7
8
9
10
N

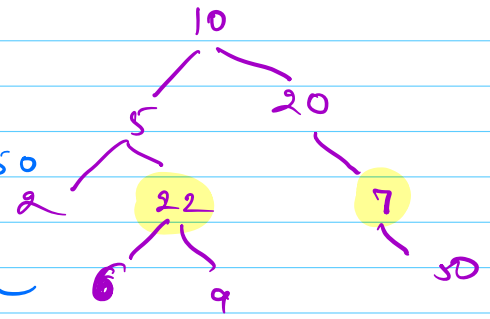
Q = Given a BST where exactly 2 nodes are swapped, find the 2 nodes (distinct nodes)

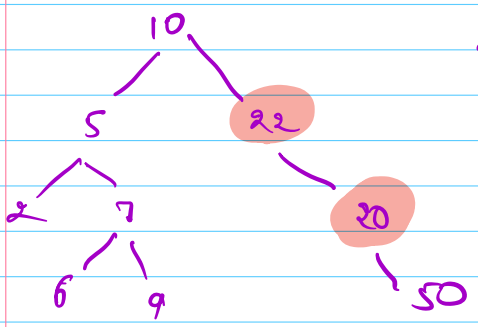
inorder → LNR
inorder traversal is sorted.

2, 5, 6, 22, 9, 10, 20, 7, 50
1 2 3 4

first element of first pair

second element of 2nd pair.





2, 5, 6, 7, 9, 10, 22, 20, 50

ans

TC: O(N), SC: O(H) → O(1)