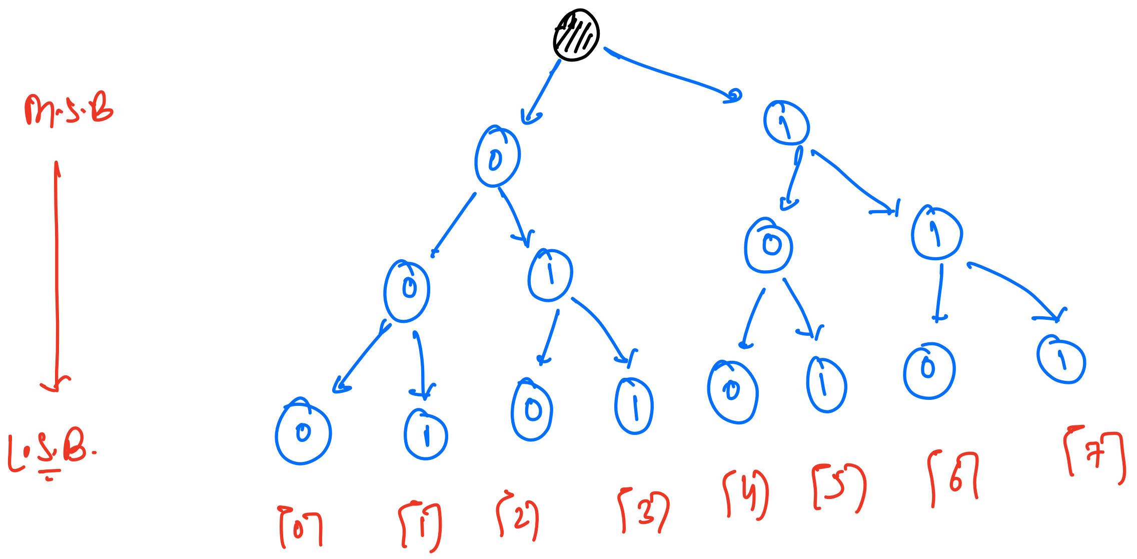
Trie → data structure

→ prefix tree.

Trie on bit representation



0-7.

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

first 2^3 no's [0-7] ⇒ height = 3

first 2^4 no's [0-15] ⇒ height = 4

first 2^5 no's $[0-2^5-1]$ ⇒ height = 5

N no's ⇒ height = $\log_2 N$

3, 6, 4, 12

3 → 0011

6 → 0110

4 → 0100

12 → 1010

```
class Node {
        int val;
        Node children[2];
}
```

children[0]   children[1]

Q: Max value of XOR pair. [A[i] ^ A[j] is maximum]

A → [9 8 10 7]
      0 1 2 3

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

idea 1. Consider all the pairs.

9 → 1001
8 → 1000
0001

9 → 1001
10 → 1010
0011

9 → 1001
7 → 0111
1110

8 → 1000
10 → 1010
0010

8 → 1000
7 → 0111
1111

10 → 1010
7 → 0111
1101

ans → 15.

A → 1 0 1 1 1 0 1
B → 0 1 0 0 0 1 0
xor → 1 1 1 1 1 1 1

A → 1 0 1 1 1 0 1
B → 1 1 0 0 0 1 0
0 1 1 1 1 1 1

2^5 — $- 2^2 + 2^1 + 2^0$

⇒ $2^6 - 1$

A → 1 0 1 1 1 0 1
B → 0 0 1 1 1 0 1
1 0 0 0 0 0 0

2^6

$2^i > 2^0 + 2^1 + 2^2 + — 2^{i-1}$

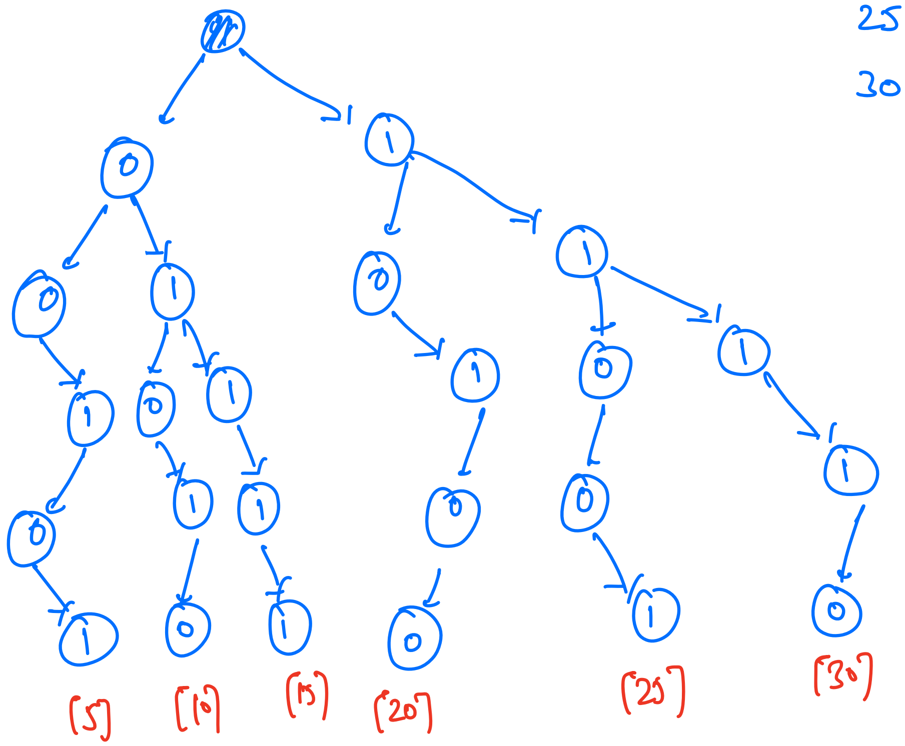
∴ MSB matters.

arr[] → [5, 20, 15, 10, 25, 30]

↓25 ↓10 ↓20 ↓20 ↓5 ↓5

xor value → [28 30 27 30 28 27]

5	→	0 0 1 0 1
20	→	1 0 1 0 0
15	→	0 1 1 1 1
10	→	0 1 0 1 0
25	→	1 1 0 0 1
30	→	1 1 1 1 0



5 → 00101
demand → 11010
✓ ✓ ✓ ✗ ✗
11100

20 → 10100
dimand → 01011
✓ ✓ ✓ ✓ ✗
11110

15 → 01111
dimand - 10000
✓ ✓ ✗ ✓ ✓
11011

10 → 01010
demand - 10101
✓ ✓ ✓ ✓ ✗
11110

#code →

① Find max element in the given arr[].

② Find count of bits in the "max" element.

```
x = 0;
while( max != 0){
        max = (max>>1);
        x++;
}
```

x → no. of bits in largest no.

③

```
Node root = new Node(-1);

for(int val : arr[]){
        insert( root, val);
}
```

④

```
for( i=0; i<N; i++){
    xor = 0 ,  Node curr = root;
    for( j= x-1 ; j >= 0; j--){
        if( checkBit( arr[i], j) == true){
            if( curr.children[0] != NULL){
                //set jth bit in xor-value
                xor = (xor | (1<<j));
                curr = curr.children[0];
            }
            else{
                curr = curr.children[1];
            }
        }
```

```
            else{
                if( curr.children[1] != NULL){
                        //set jth bit in xor-value
                        xor = (xor | (1<< j));
                        curr = curr.children[1];
                }
                else{
                        curr = curr.children[0];
                }
            }
        }
        ans = Max( ans, Xor);
    }
    return ans;
```

T.C → $O(N \times x)$
S.C → $O(N \times x)$

x → no. of bits in largest no.

Maximum Subarray XoR

arr → [1 4 3]

1	4	3
0	1	2

[1] → 1

[1,4] → 5

[1,4,3] → 6

[4] → 4

[4,3] → 7 ← ans.

[3] → 3

Idea.1. → Consider all the subarrays. T.C → $O(N^2)$

a	b	c	d	e	f	g	h
0	1	2	3	4	5	6	7

$(a \wedge b \wedge c \wedge d \wedge e \wedge f) \wedge$

$(a \wedge b \wedge c)$

xor of subarray from 3 to 5 ⇒ pxor[5] ^ pxor[2]

xor of subarray from i to j ⇒ pxor[j] ^ pxor[i-1] *

xor of

idea-2 use pXor[] & find maximum XoR pair value in pXor[].

arr → [1 4 3]
        0 1 2

pXor → [1 5 6]

ans = 7.

1^5	1^6	5^6
4	7	3

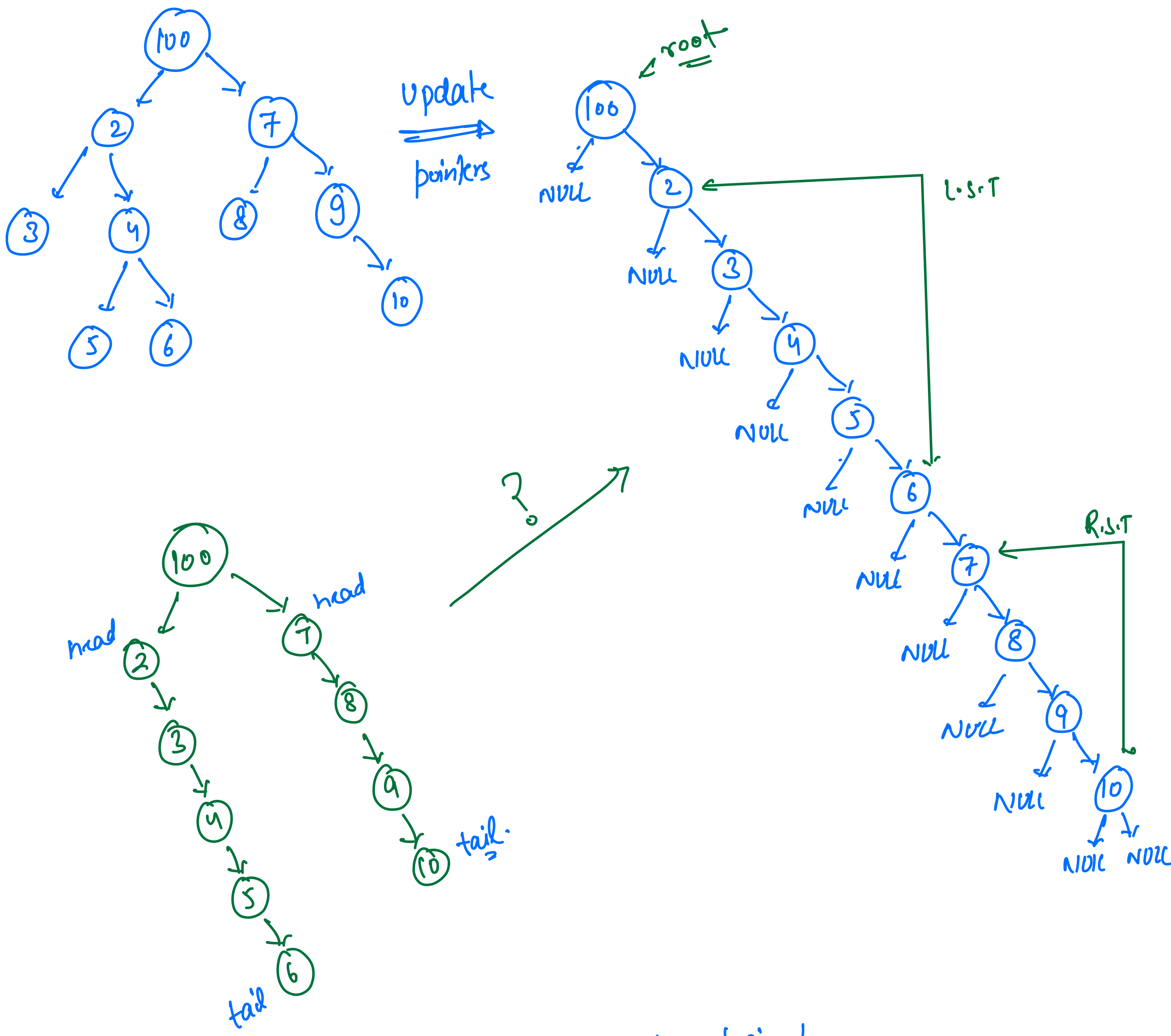
code.

```
ans = 0
for (int val : arr){
        ans = Max( val, ans);          } → subarray of size-1
}

pXor[N];
pXor[0] = arr[0];
for( i=1; i<N; i++){
        pxor[i] = pXor[i-1] ^ arr[i];  } subarray starting
        ans = max( ans, pXor[i]);      } with idx-0
}
```

→ find maximum XoR value pair in the pXor[]; } All other subarrays
(previous question)

T.C → O(N×x)
S.C → O(N×x)

Q1 Convert the given binary tree to linked-list in pre-order manner.



```
class pair {
        Node head;
        Node tail;
}
```

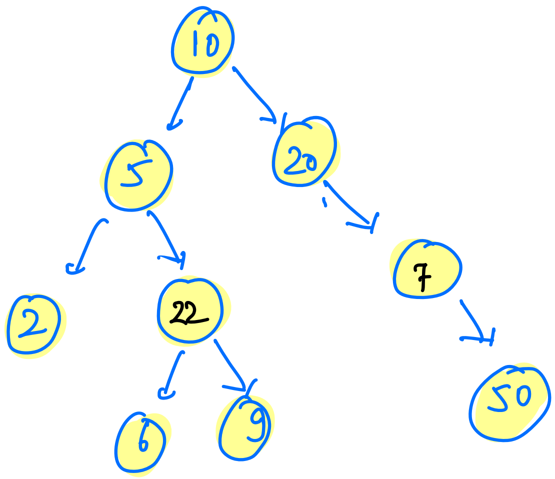
#code →

```
pair flatten( Node root){
    if(root == NULL){ return new Pair(NULL,NULL) }

    pair lp = flatten(root.left);

    pair rp = flatten( root.right);

    if( lp.head == NULL && rp.head == NULL){
            return new Pair( root, root);
    }
    else if (lp.head == NULL){
            return new Pair( root, rp.tail);
    }
    else if ( rp.head == NULL){
            root.left = NULL;
            root.right = lp.head;
            return new Pair( root, lp.tail);
    }
    else{
            root.left = NULL;
            lp.tail.right = rp.head;
            root.right = lp.head;
            return new Pair(root, rp.tail);
    }
}
```

T.C → O(N)
S.C → O(H)

#dry-run (to-do)

Q: Given a B.S.T where exactly 2 nodes are swapped. Find the two nodes. [All nodes are distinct]

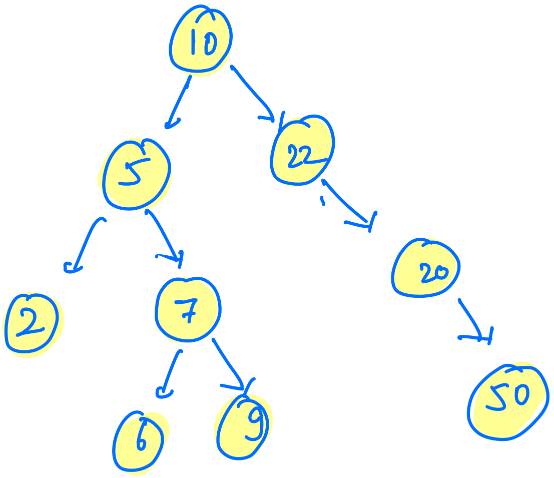


in[] -> [2, 5, 6, 22, 9, 10, 20, 7, 50]

2 dips

1st dip -> first element

2nd dip -> second element.



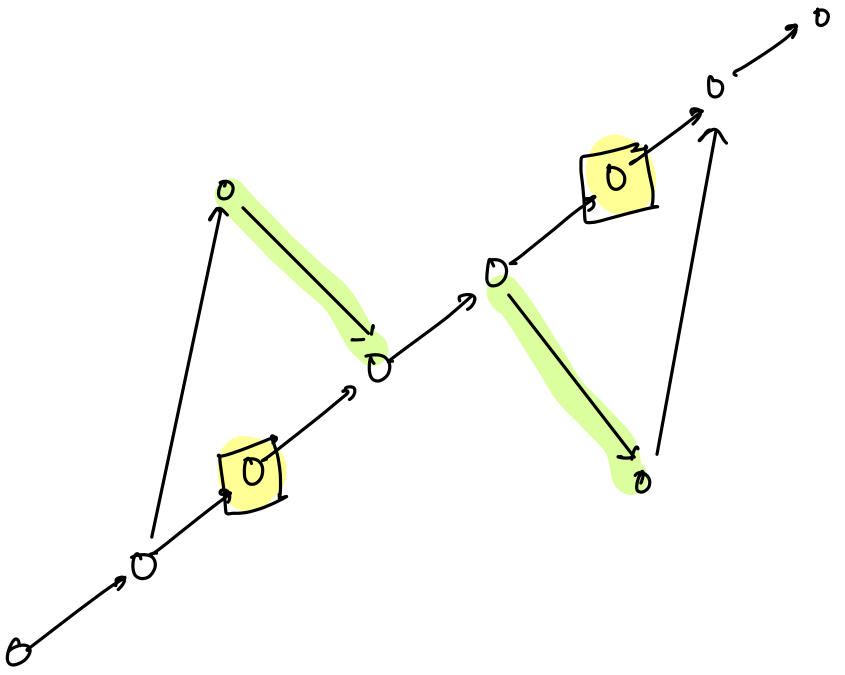
1 dip.

$$
in[] \rightarrow \begin{array}{ccccccccc} [2, & 5, & 6, & 7, & 9, & 10, & 22, & 20, & 50] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ & & & & & & & & \uparrow \end{array}
$$

ele1 -> ~~-1~~ 22

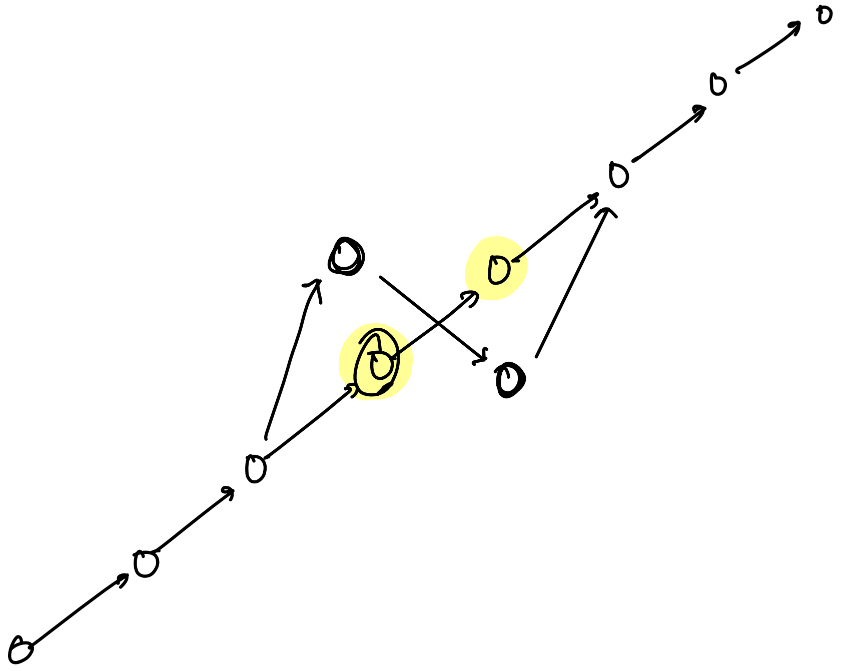
ele2 -> ~~-1~~ 20

count - ~~0~~ 1



in 1st dip ⇒ greater element is the swapped element.

in 2nd dip ⇒ smaller element is the swapped element.



1r

in[] -> [2, 5, 6, 22, 9, 10, 20, 7, 50]
　　　　0　1　2　3　4　5　6　7　8

ele1 -> ~~-1~~ 22

ele2 -> ~~-1~~ ~~9~~ 7

int count = ~~0~~ 1

[T.C -> O(N)
S.C -> O(N) -> O(H) ⟶ O(1)]

updating prev, curr on the go

Morris Traversal

```
int prev = -∞ ; int count = 0;

void traversal (Node root) {
    if (root == NULL) {return}
    traversal (root.left);
    if (root.val < prev ){
    {
        if (count == 0) { count = 1, ele1 = prev, ele2 = root.val}
        else            { ele2 = root.val };
    }
    }
    prev = root.val;
    traversal (root.right);
}
```

X

X