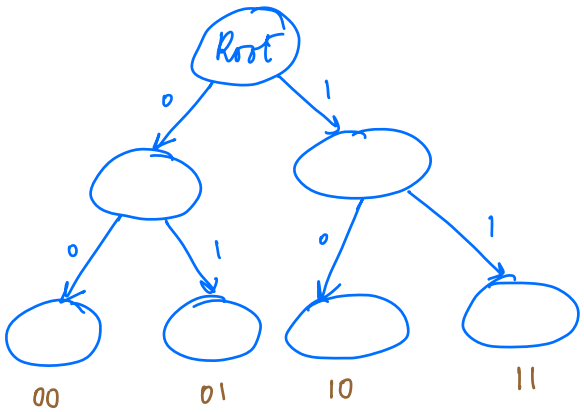
- Trie of Bits
- Maximum XOR Pair
- Maximum XOR Subarray
- Flatten binary tree to linked list
- Find swapped nodes on BST

Trie of Bits

Binary Tree where each edge denotes a 0/1. Each non-leaf node has 2 children.



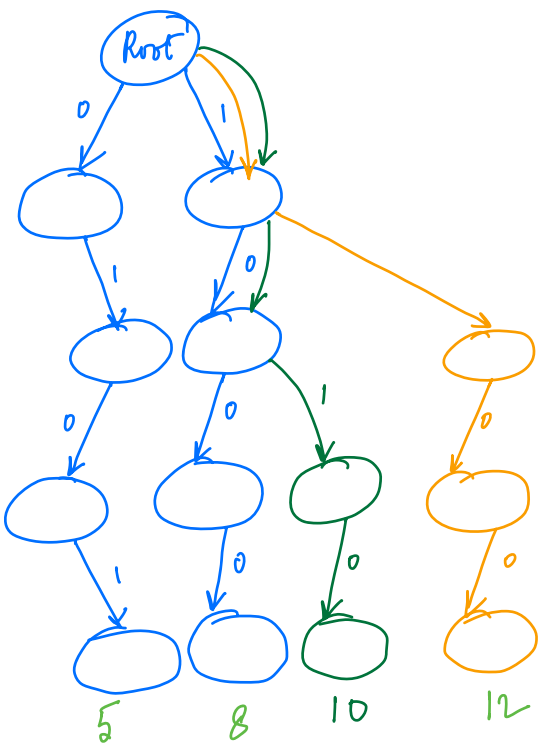
MSB LSB

5 → 0 1 0 1

8 → 1 0 0 0

12 → 1 1 0 0

10 → 1 0 1 0




```
class Node {
    Node children[];
    Node() {
        children = new Node[2];
    }
}
```

Q1) given an integer array A[], find the max value of $(A[i] \wedge A[j])$ (↑ XOR) for all i, j pairs.

$A = \{3, 5, 2\}$

$3 \wedge 5$

$$\begin{array}{r} 11 \\ \wedge\ 101 \\ \hline 110 \end{array} \rightarrow 6$$

$3 \wedge 2$

$$\begin{array}{r} 11 \\ \wedge\ 10 \\ \hline 1 \end{array} \rightarrow 1$$

$5 \wedge 2$

$$\begin{array}{r} 101 \\ \wedge\ 10 \\ \hline 111 \end{array} \rightarrow 7$$

Brute force → $O(n^2)$ T.C.
$O(1)$ S.C.

Optimization

4-bits nos. ⇒

$1000 \rightarrow 2^3 = 8$

$0111 \rightarrow 2^2 + 2^1 + 2^0 = 7$

MSB is more powerful than all the bits to its right taken together.

→ We should try to get a 1 in the MSB (best effort)

→ " " try to get a 1 in the 2nd leftmost bit (best effort)

⋮

A[] → {20, 30, 15, 25, 10, 5}

↳ #bits = 5.

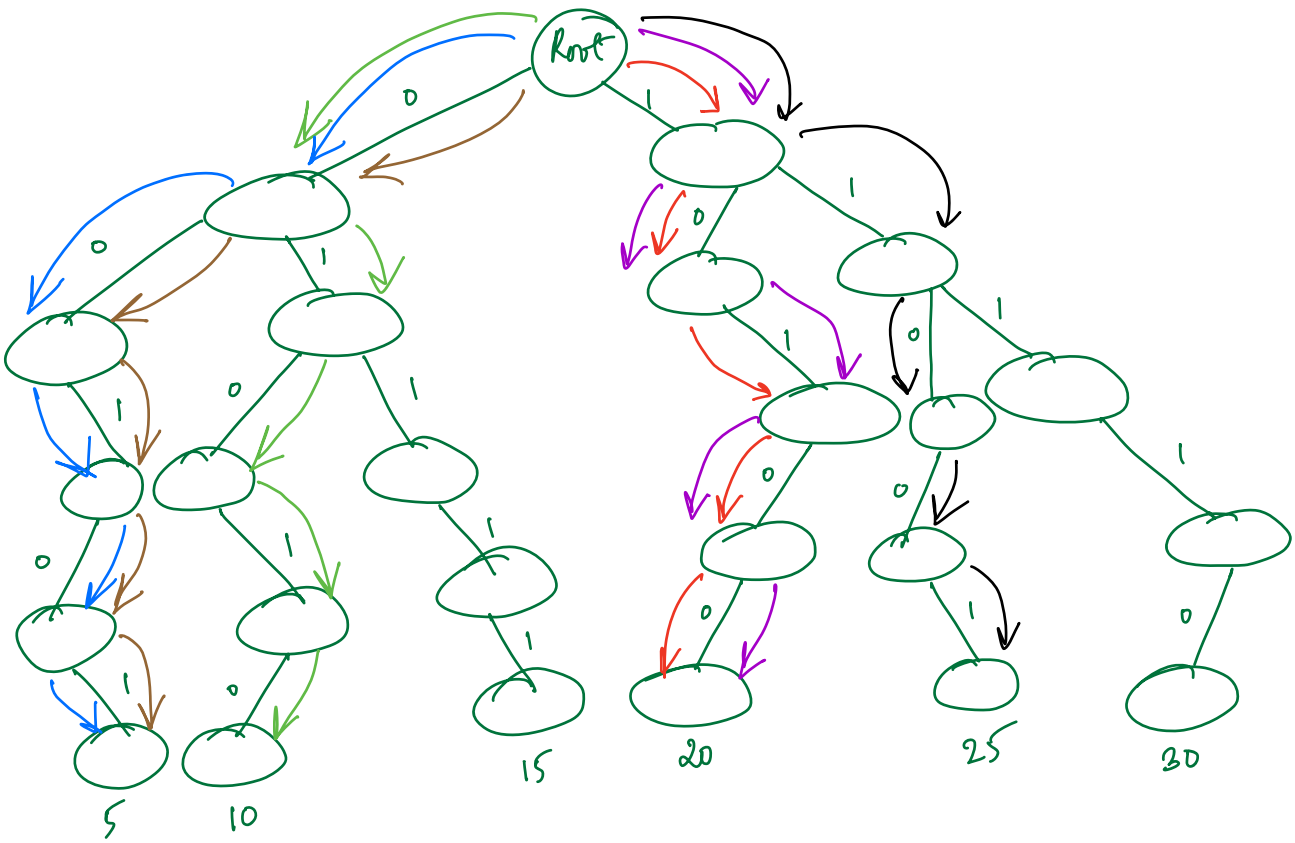
		XOR	
20 →	10100	11110	→ 30
30 →	11110	11011	→ 27
15 →	01111	11011	→ 27
25 →	11001	11100	→ 28
10 →	01010	11110	→ 30
5 →	00101	11100	→ 28

max = 30.

20 → 10100

Best Partner → 01010 XOR → 11110 → 30

01010



20 → 10100

01011 ×

0

11110

30

30 → 11110

00001 ×

1 } 5

11011

27

15 → 01111

10000 ×

1

11011

27

25 → 11001

00110 ××

01

11100

28

10 → 01010

10101

^ 0

11110

30

5 → 00101

11010

^ ^ 01

11100

28

Ans → 30.

```
class Node {
    Node children[];
    Node() {
        children = new Node[2];
    }
}
int findMaxXor(root, x) {
    cur = root; num = 0
    for (i=30; i>=0; i--) {
        bit = ((x >> i) & 1);
        t = (1 ^ bit);
        if (cur.children[t] != null) {
            cur = cur.children[t]; num |= ((1<<i)*t);
        }
        else {
            cur = cur.children[bit]; num |= ((1<<i)*bit);
        }
    }
    return (x ^ num);
}
```

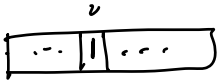
```
mx = 0;
for(i=0; i<n; i++){
    y = findMaxXor(root, A[i]);
    mx = max(mx, y);
}
```

T.C. → n * #bits

= n * 31

= O(n)

SC → O(n).

t = 1 → i 10000..0 OR → 

t = 0 → 00000..0 OR → 

$x = 20$ $\quad$ 5 4 3 2 1 0
1 0 1 0 0

0 0 1 0 1 0 0

$i = 5$

$(x >> i) \& 1 = (20 >> 5) \& 1$

$= (00) \& 1 = 0$

$i = 2$

$(x >> 2) \& 1 = (101) \& 1 = 1.$

1 0 1 0 0

Q2) Find the maximum XOR of subarray, out of all subarrays.

[4 6 1]

4

6

1

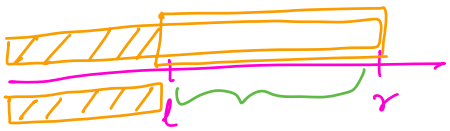
$4 \wedge 6 = 2$

$4 \wedge 6 \wedge 1 = 3$

$6 \wedge 1 = 7$ Best

$sum(A[l \ldots r]) = PSum[r] - PSum[l-1]$

$xor(A[l \ldots r]) = PXOR[r] \wedge PXOR[l-1]$

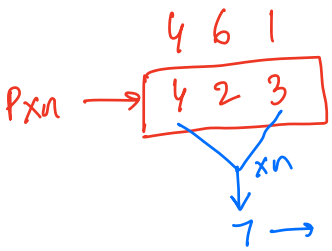


$[i, j] \longrightarrow Pxn[j] \wedge Pxn[i-1]$. if $i > 0$.

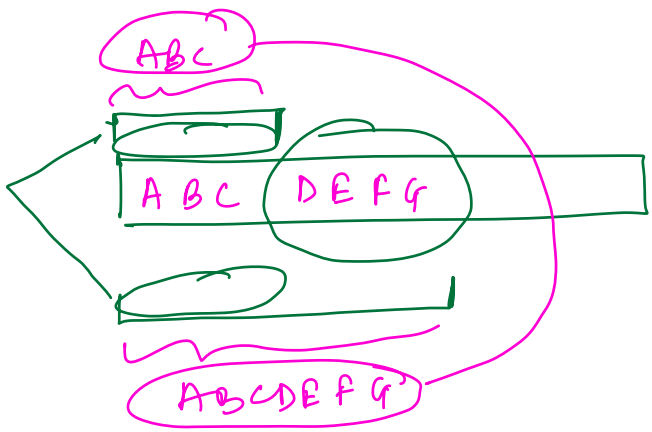
$Pxn[j]$ if $i == 0$.

$\Downarrow$

$(Pxn[j] \wedge 0)$




```
int PXOR[] = new int[n+1];
PXOR[0]=0;
for(i=0;i<n;i++){
    PXOR[i+1]=PXOR[i]^A[i];
}
//Find the max xn pair in Pxn arrag.
...
```



$(A \wedge B \wedge C) \wedge (A \wedge B \wedge C \wedge D \wedge E \wedge F \wedge G)$

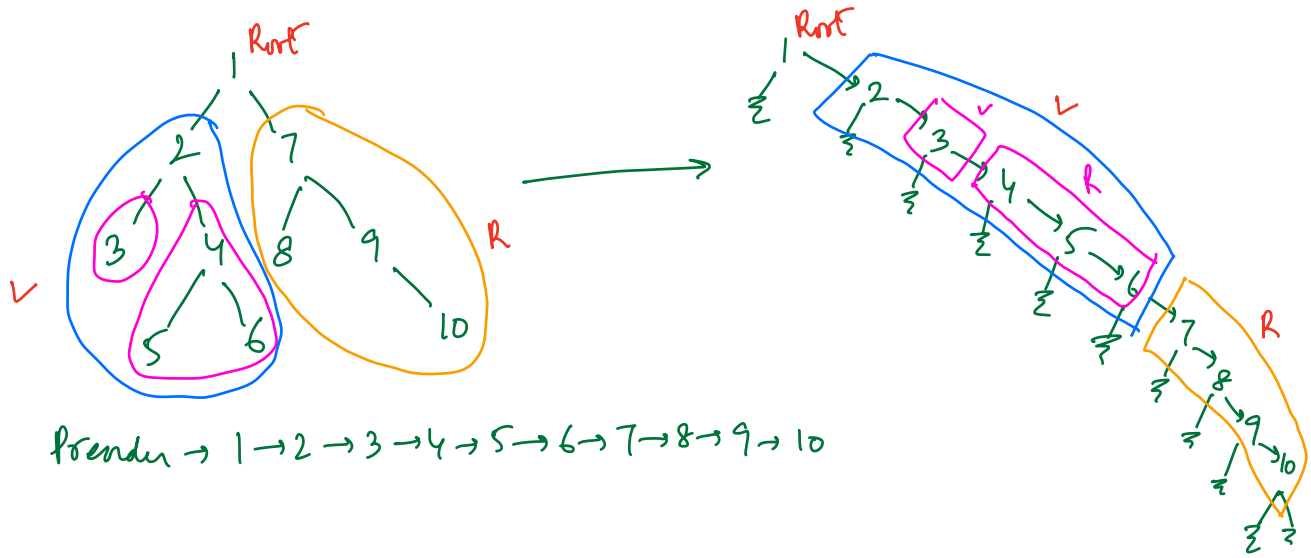
$\downarrow$

O(n) T.C

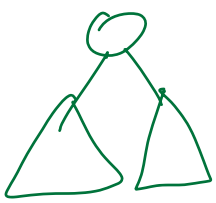
O(n) S.C.

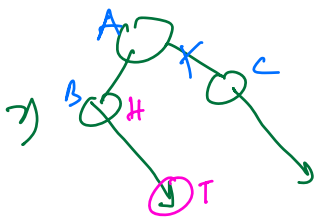
[Break till 10:45PM]

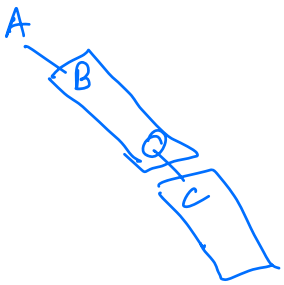
Q3) Flatten the given binary tree in Preorder manner such that right child will become the next node and left child of all nodes is null.



Preorder → 1→2→3→4→5→6→7→8→9→10







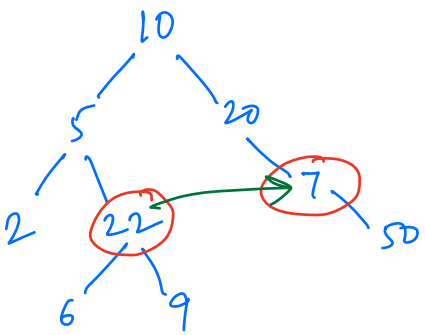

```
Pair flatten(root){
    if(root==null)
        return {null,null};
    L= flatten(root.left);
    R= flatten(root.right);
    if(root.left==null && root.right==null){
        return {root,root};
    }
    if(root.right==null){
        root.right = root.left;
        root.left = null;
        return {root, L.tail};
    }
    else if(root.left==null){
        return {root, R.tail};
    }
```

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

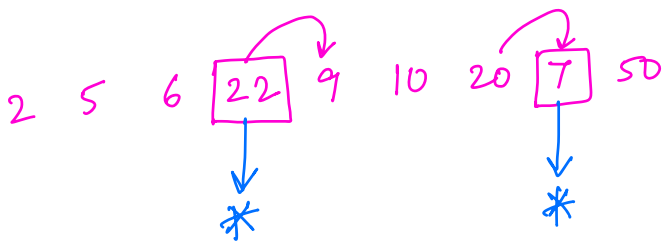
O(n) T.C
O(h) S.C.

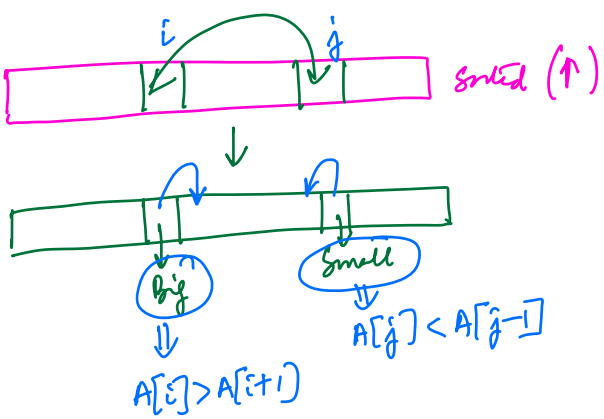
```
else{
    root.right = L.head;
    L.tail.right = R.head;
    root.left = null;
    return {root, R.tail};
}
}
```

Q4) Given a BST, where 2 nodes have been swapped, find the values of the 2 nodes. (All the values in BST are distinct)



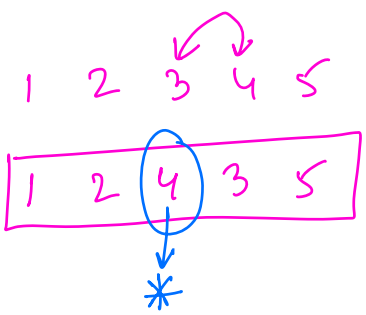
Inorder of given Tree → UNSORTED.

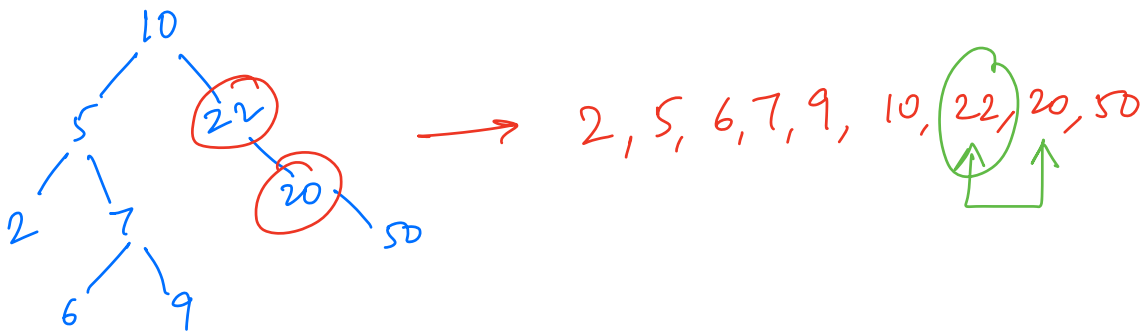




Find out nr. of areas of wrong order

- 2
  - → biggest element of 1st area
  - → smaller element of next area
- 1
  - → biggest element
  - → next element.





O(n) T.C

O(h) S.C ⟶ O(1) S.C.

```
last = null      next = 0
ans1 = 0
ans2 = 0
void inorder ( Node root) {
    if (root == null)
        return;
    inorder (root.left)
    if( last != null && root.data < last.data ) {
        if (ans1 == 0)
            ans1 = last.data , next = root.data
        else
            ans2 = root.data

    }
    last = root;
    inorder(root.right );
}
```

```
main ( ) {
    :
    :
    inorder(root);
    if (ans2 != 0)
        Print(ans1, ans2)
    else Print(ans1, next)
```