Tries + Trees

~~~~~~~
Agenda
~~~~~~~

1. Trie of Bits
2. Maximum XOR of two elements
3. Maximum XOR of a subarray
4. Flatten a Tree into Linkedlist
5. Swapped Nodes on BST

Hello Everyone
Very Special Good Evening
to all of you 😊😊😊
We will start session
from 9:06 PM

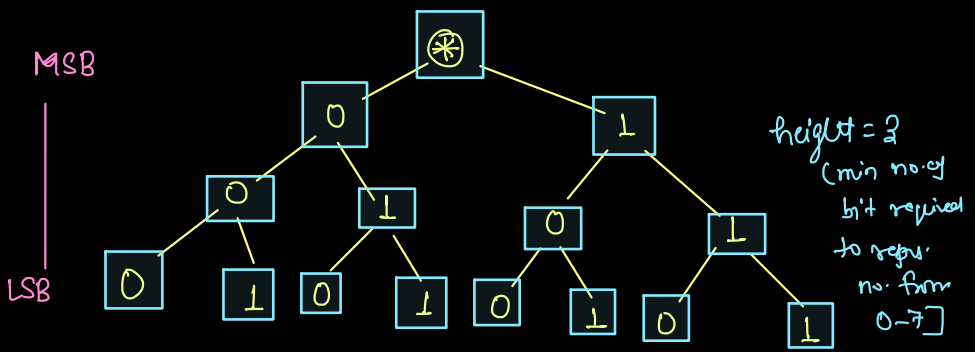
~~~~~~~~~~~~
Trie of Bits
~~~~~~~~~~~~

→ Heirarchal Data Structure
→ prefix Tree

Trie on Bit Representations

Numbers from 0 to 7:
→ min. no. of bits Required to store 0 to 7 = 3 bits

MSB LSB
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



	Range	height
0-7	2^3	3
0-15	2^4	4

0-30 → 0 to 2^5-1 → height = 5

max = N are available, height = $\log_2 N$

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

Maximum XOR of two elements

Given an integer array A, find the maximum value of A[i]^A[j] for all (i, j) pair

arr: [9, 8, 10, 7]
0 1 2 3

"Same same puppy shame"

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

Bruteforce Approach: Explore all possible pair & find max result

```
9  → 1001        9  → 1001        9  → 1001
8  → 1000        10 → 1010        7  → 0111
     0001             0011           1110

8  → 1000        8  → 1000        10 → 1010
10 → 1010        7  → 0111        7  → 0111
     0010             1111           1101
                      ↑ max
```

ans=15

Optimised Approach:

```
A → 1 0 1 1 1 0 1
B → 0 1 0 0 0 1 0
max XOR = 1 1 1 1 1 1 1
(Best possibility)
```

```
A → 1 0 1 1 1 0 1
B → 1 1 0 0 0 1 0
    0 1 1 1 1 1 1
    2^6 2^5 2^4 2^3 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
```

→ 2^6

$2^6 > 2^6 - 1$ ∴ MSB matters a lot

Second Number 25 10 20

arr[]: [5 , 20 , 15 , 10 , 25 , 30]

0 1 2 3 4 5

XOR → 28 30 27

ele → binary Representation

5 → 0 0 1 0 1

20 → 1 0 1 0 0

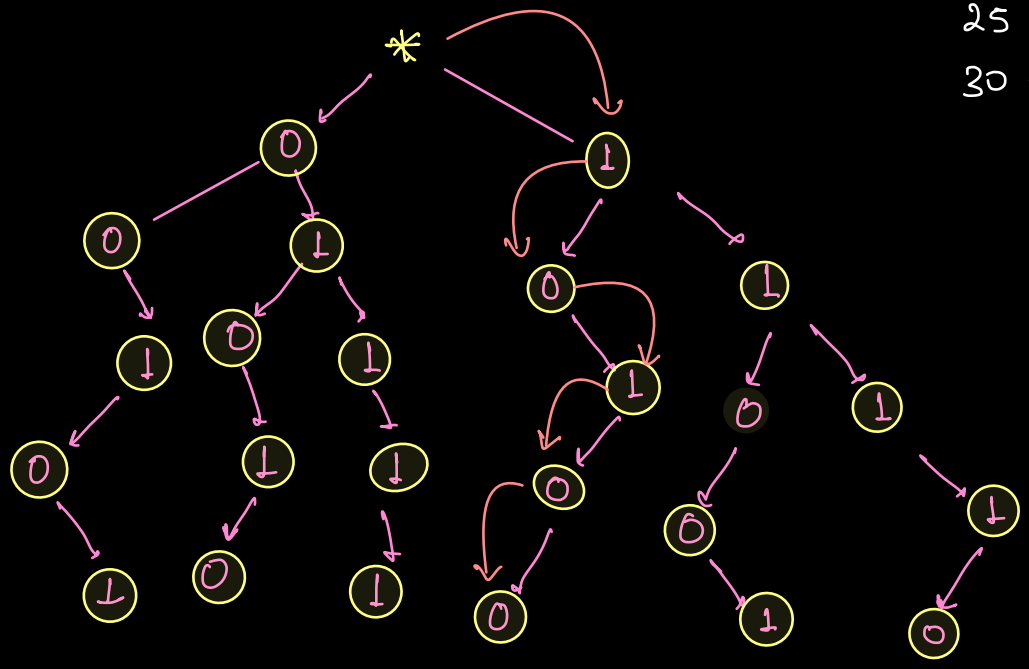
15 → 0 1 1 1 1

10 → 0 1 0 1 0

25 → 1 1 0 0 1

30 → 1 1 1 1 0

Prepare Trie of these number.



5 → 0 0 1 0 1

demand → 1 1 0 1 0

1 1 1 0 0

20 → 1 0 1 0 0

demand → 0 1 0 1 1

1 1 1 1 0

15 → 0 1 1 1 1

demand → 1 0 0 0 0

1 1 0 1 1

- - - - and so on. -

ToDO: Complete the dry Run ...

pseudo Code:

1. find max element of given array.
2. find "min count of bits" Required to make that number

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

③
```
Node root = new Node();
for(int ele: arr){
    insert(root, ele, count);   ⟶ TODO:
}
```

④
```
int max=0;
for(int ele: arr){
    max = Math.max(max, bestXORWithEle(root, ele, count))
}
return max;
```

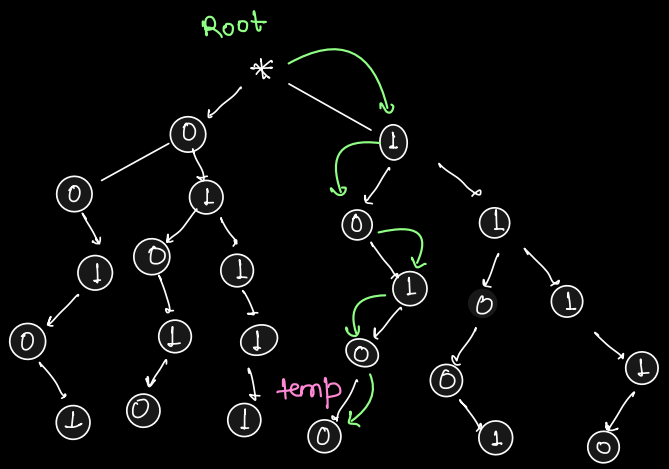
```
bestXORWithEle(Node root, int ele, int bits){
```

xor=0

ele = 15 bits = 5

ele → 0 1 1 1 1 (4 3 2 1 0)

xor → 1 1 0 1 1



bit index	bit	required	avl?	
4	0	1	T ✓	* Set the bit in xor * move in that node
3	1	0	T ✓	
2	1	0	False ✗	
1	1	0	T ✓	
0	1	0	T ✓	

```
bestXORWithEle(Node root, int ele, int bits){
    int xor=0; , Node temp=root;
    for(int bi= bits-1; bi>=0; bi--){
        // find bit at bit index in ele
        int bit = (ele & (1<<bi) != 0) ? 1 : 0;   // bit required (rb)
        if(temp.child[rb] != null){
            xor= (xor | (1<<bi));   ⟶ set the bit
            temp= temp.child[rb];
        }else{
            temp= temp.child[1-rb];
        }
    }
    return xor;
}
```

T.C: O(n * x)

S.C: O(n * x)

x → no. of bits required in max element

Maximum XOR of a subarray

Given an integer array A, find subarray with maximum XOR value

arr[]: [1, 4, 3]
index: 0 1 2

ele	bits
1	001
4	100
3	011

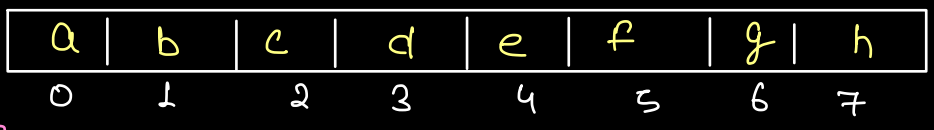
all subarrays:

- [1] → 1
- [1,4] → 5
- [1,4,3] → 6
- [4] → 4
- [4,3] → 7 max: 7, ans!
- [3] → 3

#Bruteforce: consider all possible subarrays & solve it for start to end

T.C: $O(n^3)$

consider all possible subarray :→



prefixXOR[5] → a^b^c^d^e^f

prefixXOR[2] → a^b^c

XOR from 3 to 5 → prefixXOR[5] ^ prefixXOR[2]

→ ~~a^b^c~~d^e^f ^ ~~a^b^c~~

→ d^e^f

iterate on all possible values of i, j & calculate max XOR result

XOR from (i) to (j) → pxor[j] ^ pxor[i-1] * i-1 ≥ 0

T.C: $O(n^2)$

S.C: $O(n)$

Optimised Approach:

arr: [a, b, c, d, e, f]

prefix XOR → pxor[]: [e1, e2, e3, e4, e5, e6]

XOR b/w i, j → pxor[j] ^ pxor[i-1]

max → all possible pair of (i,j) → max XOR pair in prefix of XOR array.

pseudocode:

```
int n= arr.length;
int[] pxor= new int[n];
pxor[0]= arr[0];
for(int i=1; i< n; i++) {                    ——— n
    pxor[i]= pxor[i-1] ^ arr[i];
}
```

Now use maxXOR pair fnction → previous problem and pass array as pxor:

T.C: O(n*x)
S.C: O(n*x);

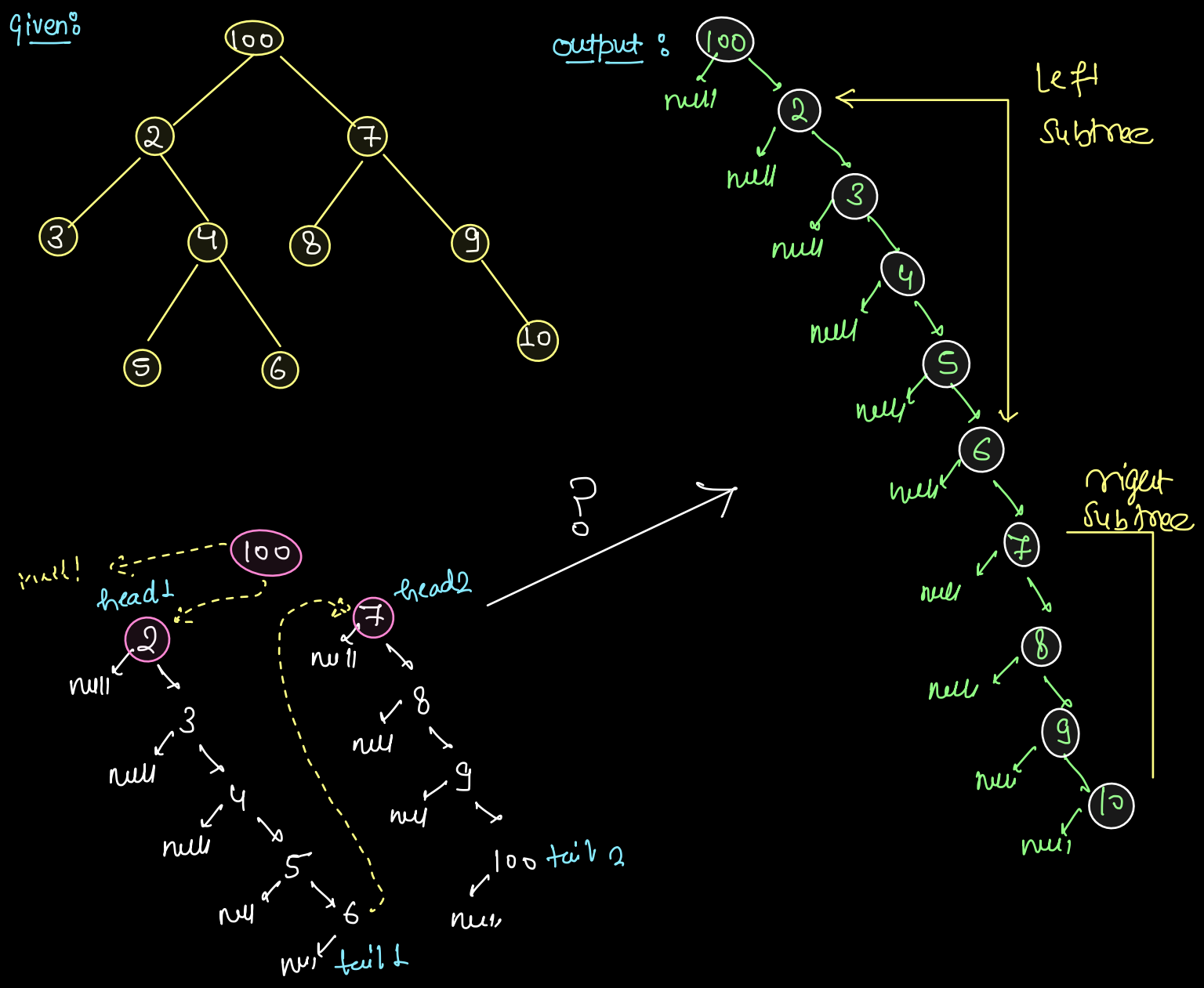
x → max bit count in pxor array.

10:44 – 10:50 pm

Break

Flatten a Tree into Linkedlist

Flatten the given binary tree to linked list in a preorder manner such that right child will become next and left child for all nodes should be null

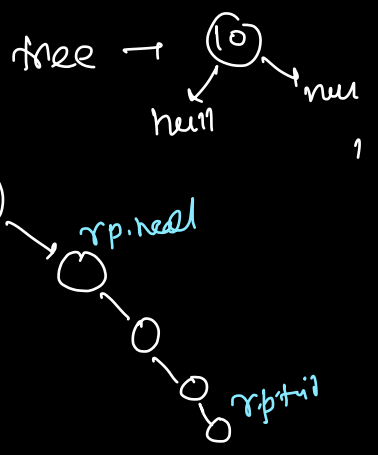


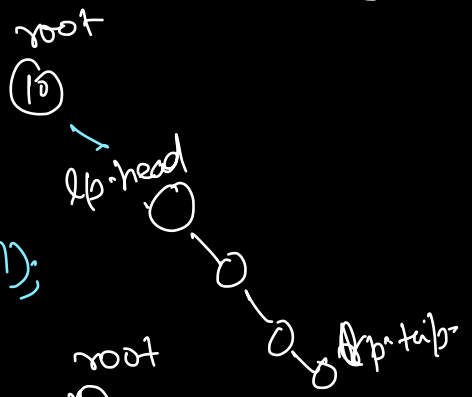

```
tail1.right = head2;
root.right = head1;
root.left = null;
```

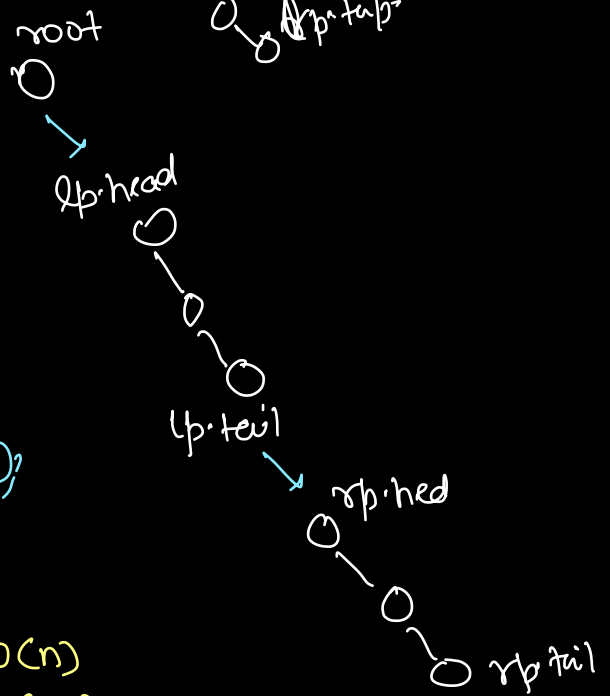
return type from traversal ⟶ pair ⇒ class pair {
Node head;
Node tail;
}

```
pair flatten (Node root) {
    if( root == null) {                    head  tail
        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.right = null ;
        lp.tail.right = rp.head ;
        root.left = null;
        root.right = lp.head;
        return new pair( root, rp.tail);
    }
}
```







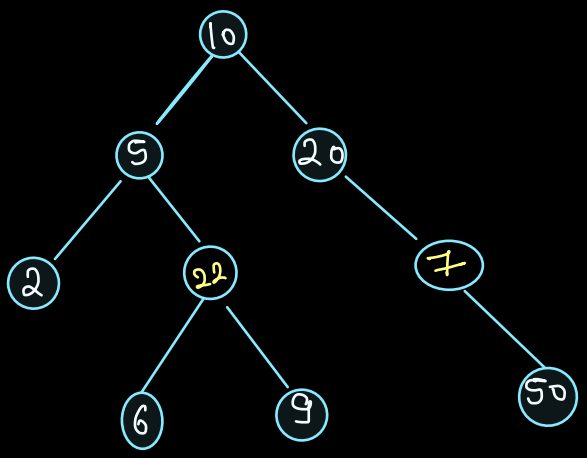
TODO: Dry Run

T.C.: O(n)

S.C.: O(n+): Recursive space + inplace changes in Tree

Swapped Nodes on BST

Given a BST where exactly 2 nodes are swapped find the two nodes (distinct nodes)



inorder is sorted.

inorder → [2, 5, 6, 22, 9, 10, 20, 7, 50]

first

second

→ inorder traversal,

→ prev pointer

→ curr pointer.

Inorder using morris traversal

Already covered in Tree section:

→ Revise it from there: