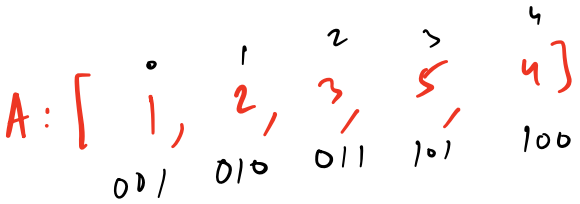
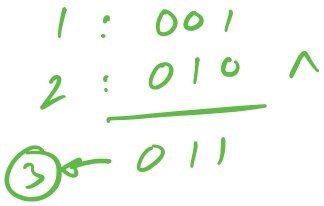
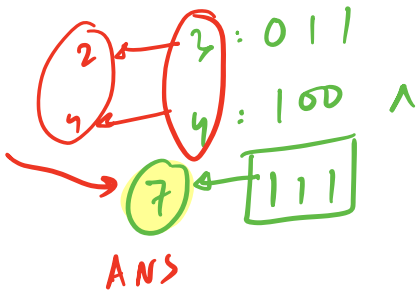
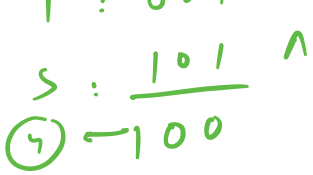
Q Given an array of integers.
Find out the MAX XOR pair.
(i, j) : $A_i \wedge A_j$ is MAX !







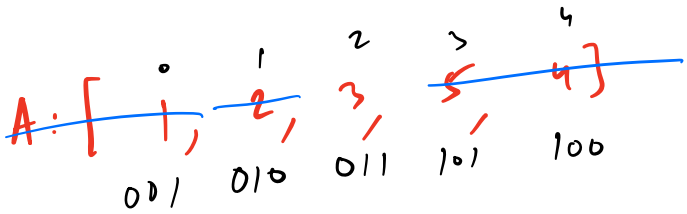


1) BF ANS = 0

```
for ( i : 0 ⟶ N-1 ) {
    for ( j = i+1 ⟶ N-1 ) {
        ANS = max ( ANS, A[i] ^ A[j] );
    }
}
return ANS;
```

TC : $O(N^2)$

SC : $O(1)$



MSB LSB

2 1 0

1 : 0 0 1

x : 1 0 0 → 4

^

MAX → 1 0 1 → (5)

2 : 0 1 0

x : 1 0 1 → 5

^

MAX → 1 1 1 → (7)

N^2

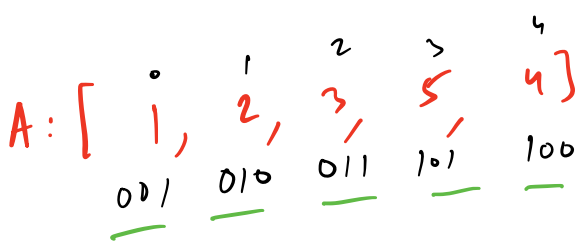
4 : 1 0 0

x : 0 1 1 → 3

^

MAX → 1 1 1 → (7)

Trie →

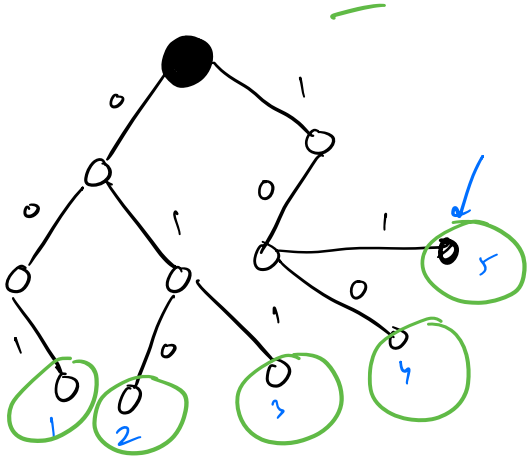


2 1 0

2 : 0 1 0

x : 1 0 1 → 5

1 1 1 → (7)



1 : 0 0 1

x : 1 0 0 → 4

^

MAX : 1 0 1 → (5)

1. find MAX no.
   → find the no. of bits → max_bits → N + log(Max(A))

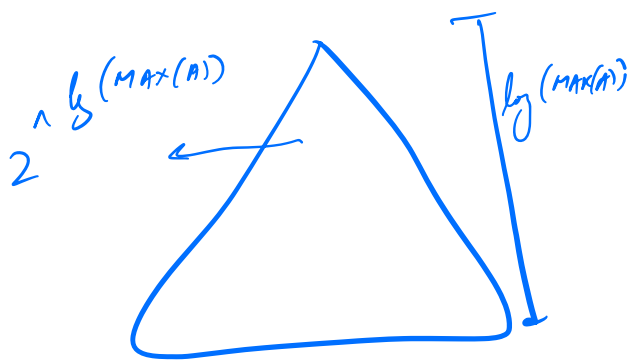
2. Create a trie [binary] → B.T
   Using the arr.
   N x log(Max(A))

3. for element
   → find the best pair → MAX
   N x log(Max(A))

TC = O(N.log(Max(A)))

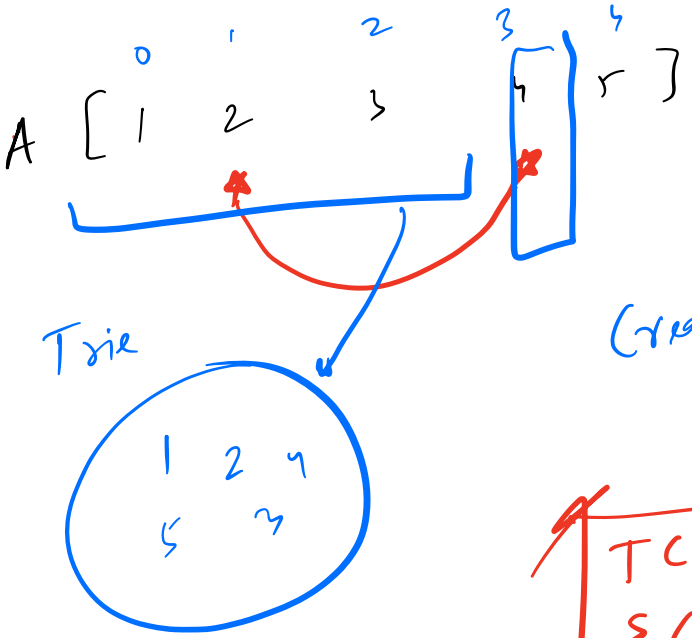
SC = O(N.log(Max(A)))

MIN(2^ ,)



Same ques → find MIN XOR PAIR (i != j)

3 : 0 1 1
n : 0 1 1 → 3 ✗
MIN : 0 0 0



Create trie on the fly!

TC → SAME AS PREV
SC

// Pseudo code

```
Trie.insert(A[0]);
ANS = 00;
for(i = 1 → N-1) {
    find best XOR in trie for A[i] → X
    ANS = MIN(ANS, X);
    Trie.insert(A[i]);
}
```

Q Given an Arr. Find the MAX XOR subarray!

A : [1, 2, 5, 3, 1, 6]

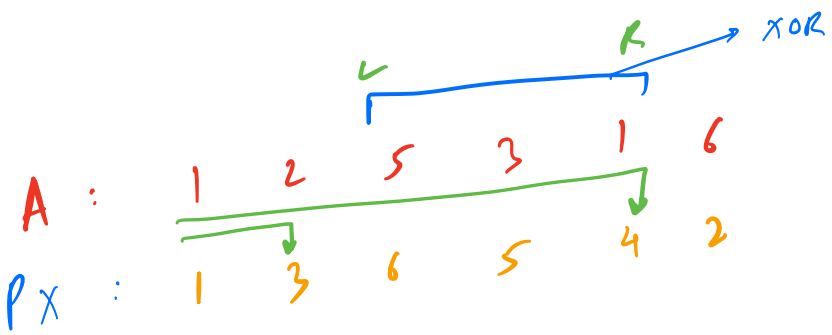
5 101
^ 3 011
^ 1 001
111 → 7

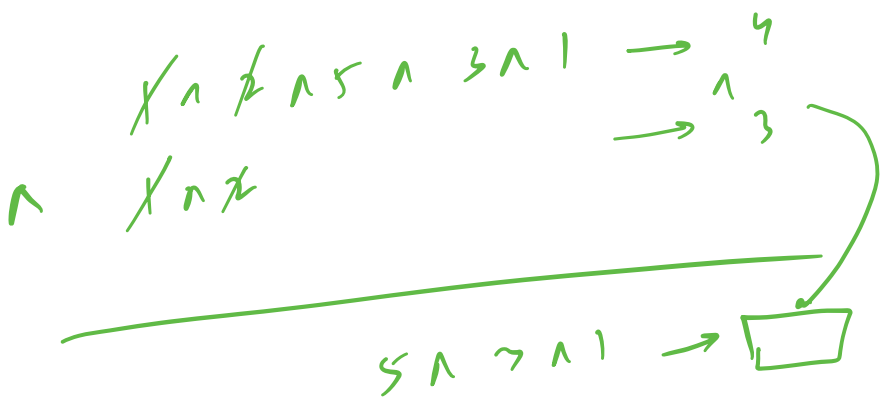
Ans

1) BF

Carry forward

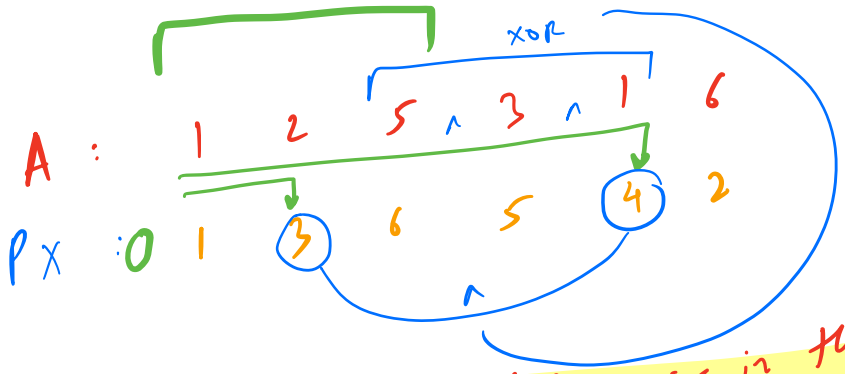
TC → O(N²)
SL → O(1)





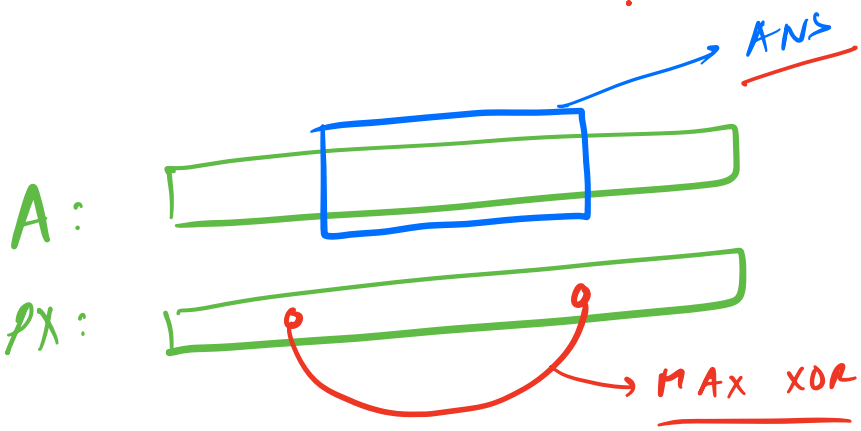
$$XOR(L,R) = PX[R] \wedge PX[L-1]$$

$L-1=0$ → 0



Idea: find the MAX XOR pair in the prefix XOR arrg!

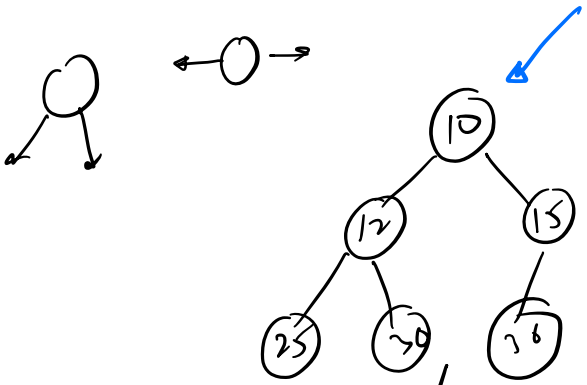
→ ANS!



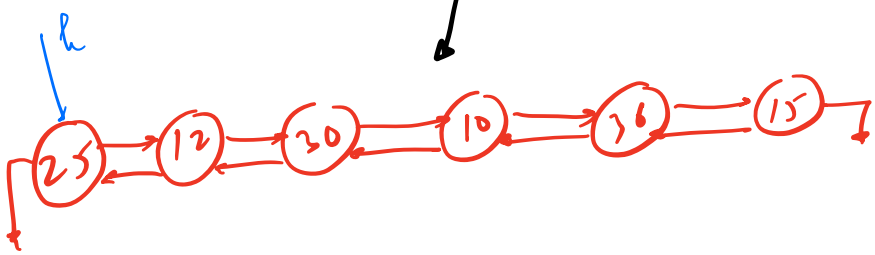
TC = SAME

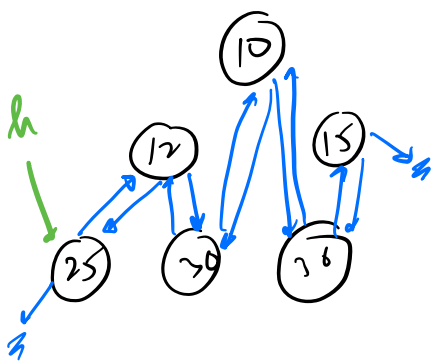
SL = SAME : don't need the PX[]

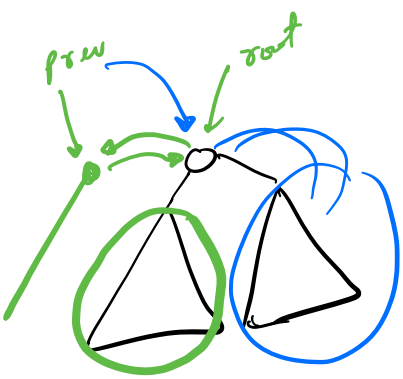
Q Given a B.T. Convert it to a DLL in INORDER!

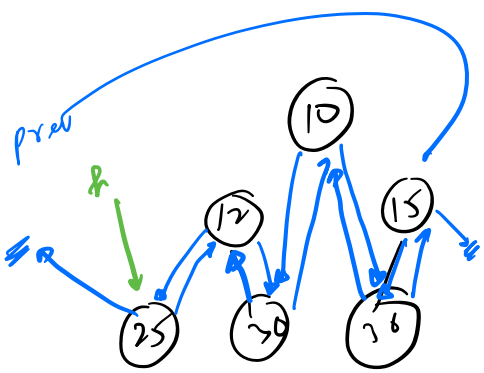


BTNode { DLL {
left —— prev
Right —— next
} }



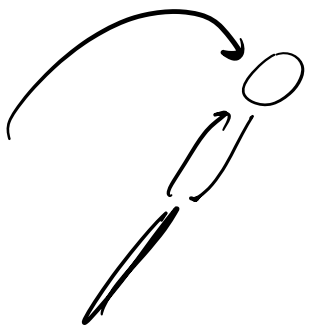






```
Node prw = NULL;
Node h = NULL;

void BT2DLL(Node root) {
    if (root == NULL) ret;
    BT2DLL(root.left);
    if (prw == NULL) {
        head = root;
    }
    else {
        root.left = prw;
        prw.right = root;
    }
    prw = root;
    BT2DLL(root.right);
}
```

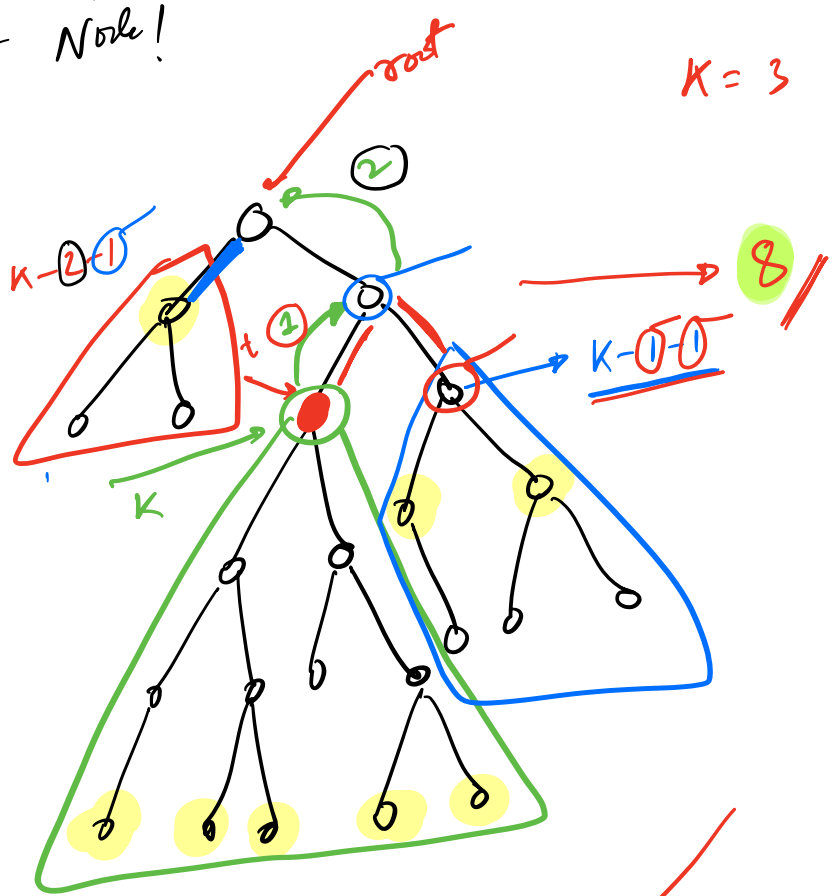


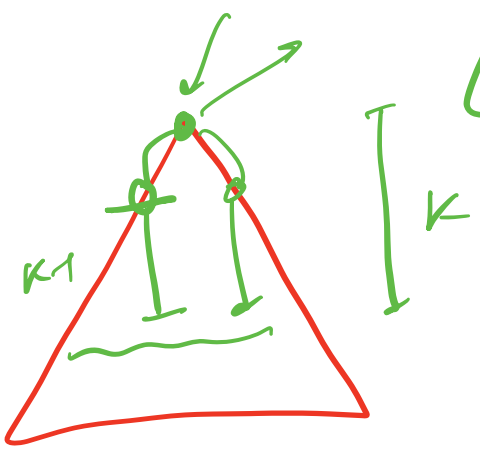
TC = O(N)

SC = O(H)

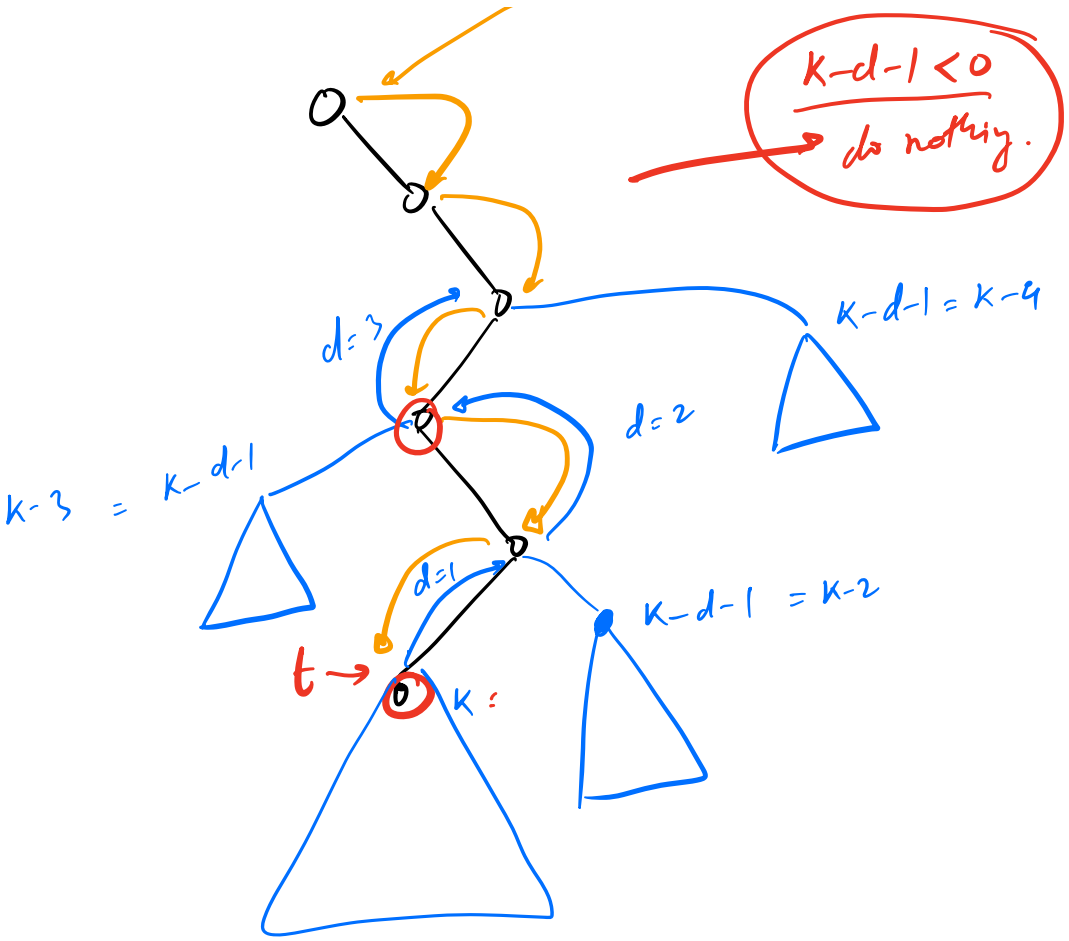
HW → Given a BT.
Convert it to Circular DLL!
inorder!

Q Given a B.T. & K & a target node. Find the no. of nodes at K distance from target Node!






```
int find ( root, K) {
    if (root == NULL) ret 0;
    if (K == 0) ret 1;
ret find (root.left, K-1);
  + find (root.right, K-1);
}
```



2 functions →

TC = O(N)

SC = O(H)