Q. Given an Array of N elements, find the max XOR value of any pair.

$A[i] \wedge A[j] \Rightarrow$ Max

$i \neq j$

	0	1	2	3
A :	4	3	2	7

$A[0] \wedge A[1] = 4 \wedge 3 = 7$ ✓

$A[0] \wedge A[2] = 4 \wedge 2 = 6$

$A[0] \wedge A[3] = 4 \wedge 7 = 3$

$A[1] \wedge A[2] = 3 \wedge 2 = 1$

$A[1] \wedge A[3] = 3 \wedge 7 = 4$

$A[2] \wedge A[3] = 2 \wedge 7 = 5$

Idea 1 :- Brute force

TC : $O(N^2)$

SC : $O(1)$

↓

A = 10000 B = 01111

MSB is set.

A > B

Ex :- N = 9

A : { 22 , 61 , 38 , 27 , 21 , 34 , 42 , 37 , 43 }
(0) (1) (2) (3) (4) (5) (6) (7) (8)

	5	4	3	2	1	0
22:	0	1	0	1	1	0
× 61:	1	1	1	1	0	1
38:	1	0	0	1	1	0
27:	0	1	1	0	1	1
21:	0	1	0	1	0	1
× 34:	1	0	0	0	1	0
42:	1	0	1	0	1	0
× 37:	1	0	0	1	0	1
✓ 43:	1	0	1	0	1	1

A = 22
Find max value of A^B.

	MSB 5	4	3	2	1	0
A:	0	1	0	1	1	0
B:	1	0	1	0	1	1
A^B:	1	1	1	1	0	1

↑ Max

A^B ⇒ 61
A = 22, B = 43

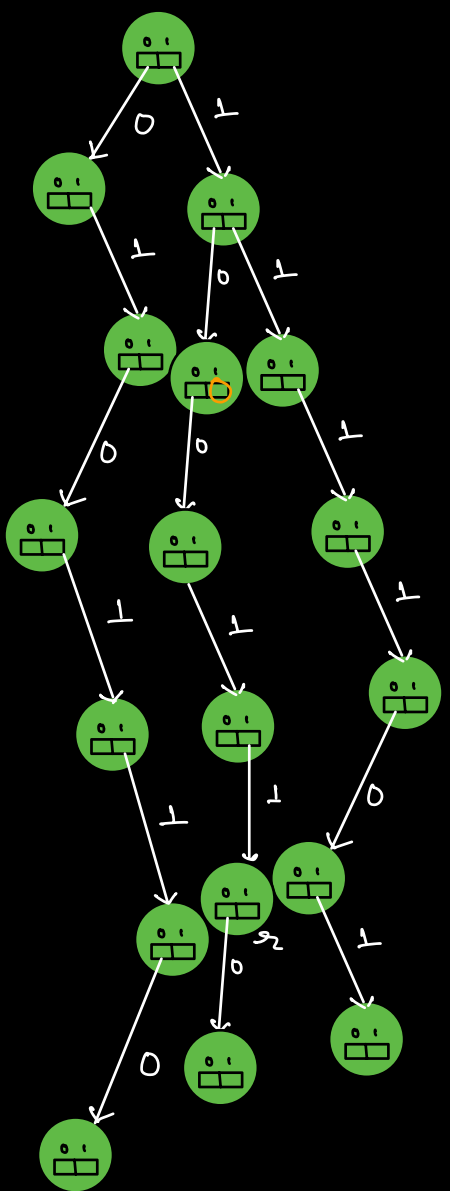
Overall TC : N * (N × 31)

: O(N²)

⇒ Prefix Search can be done optimally using Trie DS. (Binary Trie)

⇒ for all the no's, we need to insert same no. of bits in Trie.

⇒ Get Max of Array & find the index of last set bit.



```
Class Node {
    Node C[2];
    Node() {
        C[0] = Null;
        C[1] = Null;
    }
}

int maxXOR( int arr[], int N ) {
    int mx = max(arr);
    int b = maxSetBit(mx);
    Node root = new Node();
    for( i = 0; i < N; i++) {
        insert(root, arr[i], b);
    }
    ans = 0
    // fix every element & get MAX XOR.
    for( i = 0; i < N; i++) {
        ans = max(ans, query(root, arr[i], b)
    }
    return ans;
}
```

N x b

N x b

(A)

```
void insert(root, ele, b) {
    for(int i = b; i >= 0; i--) {
        // ith bit in ele.
        e = checkBit(ele, i);
        if (root.C[e] == Null) {
            root.C[e] = new Node();
            root = root.C[e];
        }
        else {
            root = root.C[e];
        }
    }
}

int query(root, ele, b) {
    ans = 0
    for(i = b; i >= 0; i--) {
        e = checkBit(ele, i);
        // If e -> 1, We need 0
        // If e -> 0, We need 1
        // e -> 1-e
        if (root.C[1-e] != Null) {
            // 1-e is present
            // Set ith bit in ans.
            ans = ans + 2^i
            root = root.C[1-e];
        }
    }
}
```

0
1

$2^i \rightarrow (1<<i)$

ans |= (1<<i)

```
        else{
            root = root.C[c];
        }
    }
    return ans;
}
```

Overall TC: $O(N \times b)$ where $b <= 31$

↑ max Set bit pos^ in the Array elements.

$\Rightarrow \log_2(\max(Arr))$

SC: $O(N \times b)$