421. Maximum XOR of Two Numbers in an Array

Given an integer array `nums`, return *the maximum result of* `nums[i] XOR nums[j]`, where `0 <= i <= j < n`.

```
Input: nums = [3,10,5,25,2,8]
Output: 28
Explanation: The maximum result is 5 XOR 25 = 28.
```

3, 10, 5, 25, 2, 8

```
  0 0 1 1   -> 3
^ 1 0 1 0   -> 10
-----------
  1 0 0 1   -> 9
```

p	q	R : p ^ q
0	0	0
0	1	1
1	0	1
1	1	0

a = 00001010

b = 11110101

M = 11111111

given a, M

↳ Integer. MAX.VALUE

calculate b?

a) 11001001

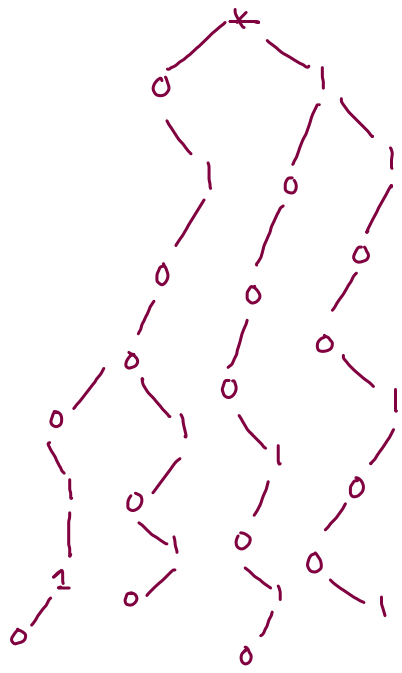
b) 01001010

c) 10001010

d) 01000110

$a \hat{} b = M$

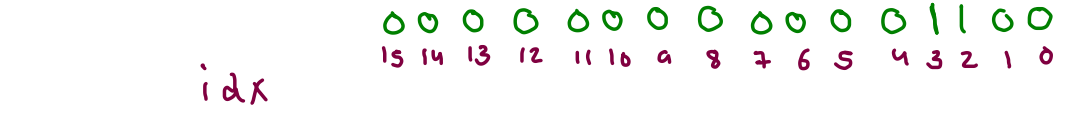
$b = a \hat{} M$



left : 0

right : 1

val = 12



right most bit idx = 30

left most bit idx = 0

n = 27, k = 3

mask = 2^k

= 01000

only kth bit is on

```
  11011  (n)
& 01000  (mask)
-------
  01000
```

```java
public void insert(int val) {
    int idx = 30; // right most bit index
    Node curr = root;

    while(idx >= 0) {
        int mask = 1 << idx;
        int bit = (mask & val) == 0 ? 0 : 1;

        if(bit == 0) {
            if(curr.left == null) {
                curr.left = new Node();
            }
            curr = curr.left;
        }
        else {
            if(curr.right == null) {
                curr.right = new Node();
            }
            curr = curr.right;
        }

        idx--;
    }
}
```

arr:

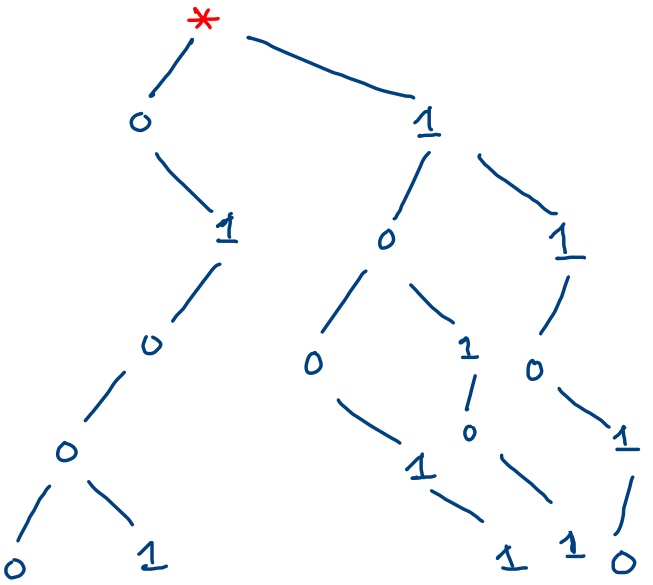
01001

10011

11010

01000

10101



```java
public int search(int key) {
    int idx = 30; //right most bit index
    Node curr = root;
    int ans = 0;

    while(idx >= 0) {
        int mask = (1 << idx);
        int bit = (mask & key) == 0 ? 0 : 1; //de

        //check availability
        if(bit == 0) {
            if(curr.left != null) {
                //desired bit is available
                curr = curr.left;
            }
            else {
                //desired bit is not available
                ans = ans | mask;
                curr = curr.right;
            }
        }
        else {
            if(curr.right != null) {
                //desired bit is available
                ans = ans | mask;
                curr = curr.right;
            }
            else {
                //desired bit is not available
                curr = curr.left;
            }
        }

        idx--;
    }

    return ans;
}
```

```java
Node root;
public int findMaximumXOR(int[] nums) {
    root = new Node();

    for(int val : nums) {
        insert(val);
    }


    int max = 0;
    for(int i=0; i < nums.length;i++) {
        int a = nums[i];
        int b = a ^ Integer.MAX_VALUE;
        int res = search(b);
        int ans = a ^ res;
        max = Math.max(max,ans);
    }

    return max;
}
```

a = 01101

b = 10010

ans = 10011

01001

10011

11010

01000

10101