Q.1 Given an Array of non negative integers, find the pair with Minimum XOR value Return any pair.

Amazon
Adobe
Booking.com
...

A: { 0 2 7 5 }

0 ^ 2 → 000 ^ 010 = 010 = 2

7 ^ 5 → 111 ^ 101 = 010 = 2 ....

ans = 2 (0^2)

Brute Force:

Find XOR of all possible pairs & find MIN.

TC: $O(N^2)$

SC: $O(1)$

Observation

(a^b) xor → 0 (bits are same)

(a^b) xor → 1 (bits aren't same)

To get the minimum XOR, try to have as many similar bits as possible.

a: 101101
b: 001101
^
x = 100000
MSB LSB

>

a: 101101
b: 111101
^
y = 010000

$$\begin{array}{r} 101101 \\ \wedge\ 001101 \\ \hline c = 100000 \end{array} \quad > \quad \begin{array}{r} 101101 \\ \wedge\ 110010 \\ \hline d = 011111 \end{array}$$

MSB

To get the Minimum XOR of a pair of no's, Numbers should be as close as possible.

a = 100, b = 1 } ①

c = 100, d = 99 } ②

3^7: 011 ^ 111 = 100 = 4

4^7: 100 ^ 111 = 011 = ③

A: { 4, 3, 5, 7, 6 }

↓ Sort

{ 3, 4, 5, 6, 7 }

3^4: $011 \wedge 100 = 111 = 7$

4^5: $100 \wedge 101 = 001 = 1$

5^6: $101 \wedge 110 = 011 = 3$

6^7: $110 \wedge 111 = 001 = 1$

1) Sort the Array

2) Find XOR of adjacent pairs.

TC: $O(N \log N)$

SC: Depends on sorting algo.

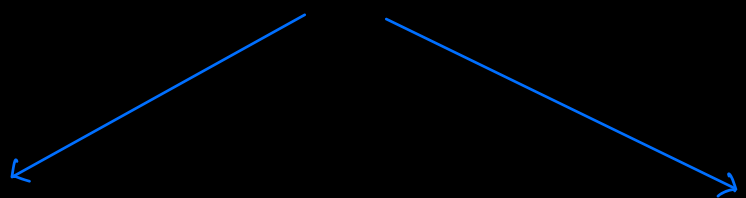
Proof : Adjacent no's will only give the MIN XOR.

$A \geq C \geq B$

A : 1 0 1 1 _ _ _ _ _

C : 1 0 1 40 _ _ _ _ _

B : 1 0 1 0 _ _ _ _ _



A : 1 0 1 1 _ _ _ _ _

C : 1 0 1 0 _ _ _ _ _

B : 1 0 1 0 _ _ _ _ _

$C \wedge B \Rightarrow 0\ 0\ 0\ 0\ \_\ \_\ \_\ \_\ \_$

$C \wedge B \Rightarrow$ Min

A : 1 0 1 1 _ _ _ _ _

C : 1 0 1 1 _ _ _ _ _

B : 1 0 1 0 _ _ _ _ _

$A \wedge C \Rightarrow$ Min

Q.2 Given an array of non negative elements, Calculate the sum of XOR of all possible pairs.

Google

	0	1	2	3	4
A : {	3	2	8	5	6 }

(3,3)	(3,2)	(3,8)	(3,5)	(3,6)
(2,3)	(2,2)	(2,8)	(2,5)	(2,6)
(8,3)	(8,2)	(8,8)	(8,5)	(8,6)
(5,3)	(5,2)	(5,8)	(5,5)	(5,6)
(6,3)	(6,2)	(6,8)	(6,5)	(6,6)

Sum of all pairs = n

XOR = 0

$a \wedge a = 0$

$a \wedge b = b \wedge a$

$ans = 2 * n$

```
for( i = 0; i < N; i++ ){
    for( j = i+1; j < N; j++ ){
        sum += ( A[i] ^ A[j] );
    }
}
return 2 * sum;
```

TC : $O(N^2)$

SC : $O(1)$

Observation

XOR → 1 (Bits are different)

→ 0 (Bits are same)

	0	1	2	3	4
A :	3	2	8	5	6
	011	010	1000	101	110

	2^3	2^2	2^1	2^0
	3	2	1	0
3^2 =	0	0	0	1
3^8 =	1	0	1	1
3^5 =	0	1	1	0
3^6 =	0	1	0	1
2^8 =	1	0	1	0
2^5 =	0	1	1	1
2^6 =	0	1	0	0
8^5 =	1	1	0	1
8^6 =	1	1	1	0
5^6 =	0	0	1	1
	4	6	6	6
	×	×	*	*
	2^3	2^2	2^1	2^0

$32 + 24 + 12 + 6$

$\Rightarrow 74$

ans = $2 \times 74 = 148$.

Idea :-

Count the no. of pairs with i^{th} bit as Set $\rightarrow x$

Contribution = $x * 2^i$

$$\text{Sum} = \sum_{i=0}^{31} x * 2^i$$

$\rightarrow 2 * \text{Sum}$.

Iterations $= N^2 \cdot 32$

$a \wedge b = \_\_\_\_\underset{i^{th}}{1}\_$

$\Rightarrow$ a, b should have diff. bits at i^{th} pos^.

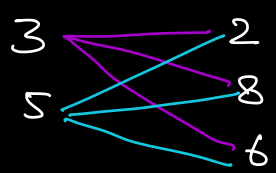
1000

A : {3, 2, 8, 5, 6} $\Rightarrow$ N

011 010 101 110

No's with 0^{th} bit Set $\Rightarrow$ 2 (3, 5)

No's with 0^{th} bit UnSet $\Rightarrow$ 3 (2, 8, 6)



$\Rightarrow 6$

Sum += 6 * 2^0 $\Rightarrow$ Sum = 6

No's with 1^{st} bit Set $\Rightarrow$ 3

No's with 1^{st} bit UnSet $\Rightarrow$ 2

$\rightarrow 6$

Sum += 6 x 2^1 = 18

No's with 2^{nd} bit Set $\Rightarrow$ 2

No's with 2^{nd} bit UnSet $\Rightarrow$ 3

$\rightarrow 6$

Sum += 6 x 2^2 $\rightarrow$ Sum = 42

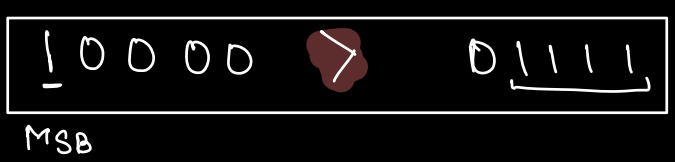
Code

```
for (i = 0; i < 32; i++) {
    n = 0
    for (j = 0; j < N; j++) {
        if ( checkBit(A[j], i) )
            n++;
    }
    Sum = Sum + n * (N - n) * 2^i      (1<<i)
}
return 2 * Sum;
```

O(1)

TC : $O(\log_2(MAX) \cdot N)$

SC : $O(1)$



Q.3 Given an array of non negative elements, Return max '&' value of any pair.

return max(A[i] & A[j]), i != j

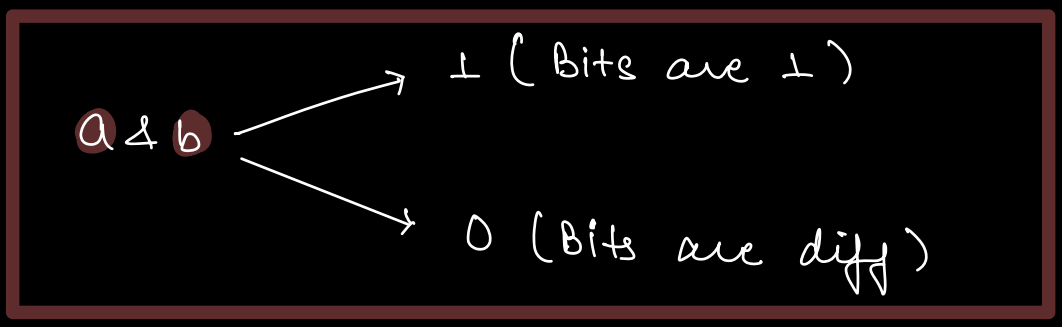
A : { 27, 18, 20 }

27 : 11011
18 : 10010
20 : 10100

27 & 18 :
```
  11011
& 10010
-------
  10010
```
→ 18

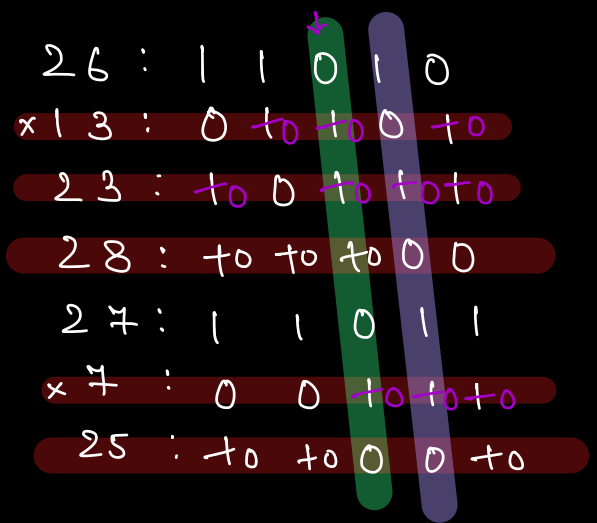
27 & 20 :
```
  11011
& 10100
-------
  10000
```
→ 16

18 & 20 :
```
  10010
& 10100
-------
  10000
```
→ 16



⇒ Ans will be maximum if set bits are present more towards MSB side.

A : [26 , 13 , 23 , 28 , 27 , 7 , 25]



1 1 0 1 0 ⇒ 11010

↑ i

26

⇒ Try to set the MSB first

⇒ if count of elements with i^{th} bit set ≥ 2 ⇒ Set i^{th} bit in ans.

⇒

```
ans = 0
for(i=31; i>=0; i--) {          → log2 MAX
    setBitCount = 0;
    for(j=0; j<N; j++) {
        if(checkBit(A[j], i)) {
            setBitCount++
        }
    }
    if(setBitCount >= 2) {
        ans = ans | (1<<i);
        for(j=0; j<N; j++) {
            if(!checkBit(A[j], i)) {
                A[j] = 0;
            }
        }
    }
}
```

TC: $O(\log_2(Max) \cdot N)$

SC: $O(1)$

↳ If we are updating the given Array.