Q. Given an Array of size N. where every no. occurs twice except one. Find that unique no.

A = [4, 5, 5, 4, 6, 1, 6]

Ans = 1

Sol^n 1) Take XOR of all elements.

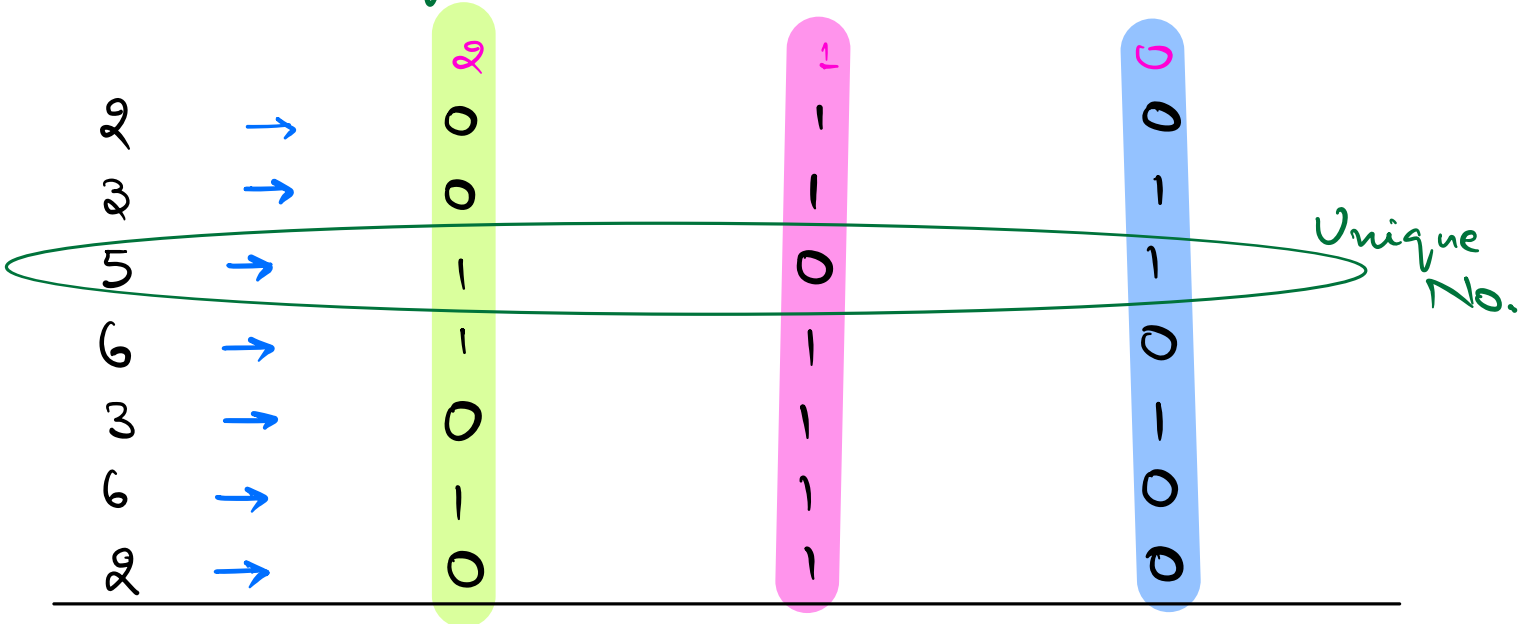
Code

```
ans = 0;
for (i=0; i<N; i++)
    ans = ans ^ A[i];
return ans;
```

T.C. = O(N)
S.C. = O(1)

2) Interesting Sol^n

A = [2, 3, 5, 6, 3, 6, 2]



Ans ⇒ 1 0 1

3 6 (Even ⇒ 2x) 3 (Odd ⇒ 2x+1)

⇓ ⇓ ⇓

Set 1st bit in Unique Element is Not Set

0th bit in Unique no is set.

0th bit : 0^1^1^0^1^0^0 → 1 (# set bits is Odd)

→ 0 (# Set bits is Even)

Freq of element → 2

→ 1 (Only for unique element)

Count of set bit on ith position → 2×(x)

→ 2×(x) + 1

2×(x) : Duplicate Elements

1 : Unique Element.

Code

```
int ans = 0;

for (i=0; i<32; i++) { // Bits
    int count = 0;
```

```
for (j=0; j<N; j++) {  // Array elements
    if ((A[j] & (1<<i)) > 0) {
        count ++;
    }
}
if ((count & 1) == 1) {
    ans = ans | (1<<i);   Set ith bit in ans.
}
}
```

T.C. = O(N)

Given an integer array of size N.

All the no's occurs thrice except one which occurs only once.
find the unique no.

A = [4, 5, 5, 1, 4, 6, 4, 6, 5, 6]

(Ans = 1)

Soln 1) Brute force

∀ elements ⇒ Iterate & find freq.
O(N) O(N)

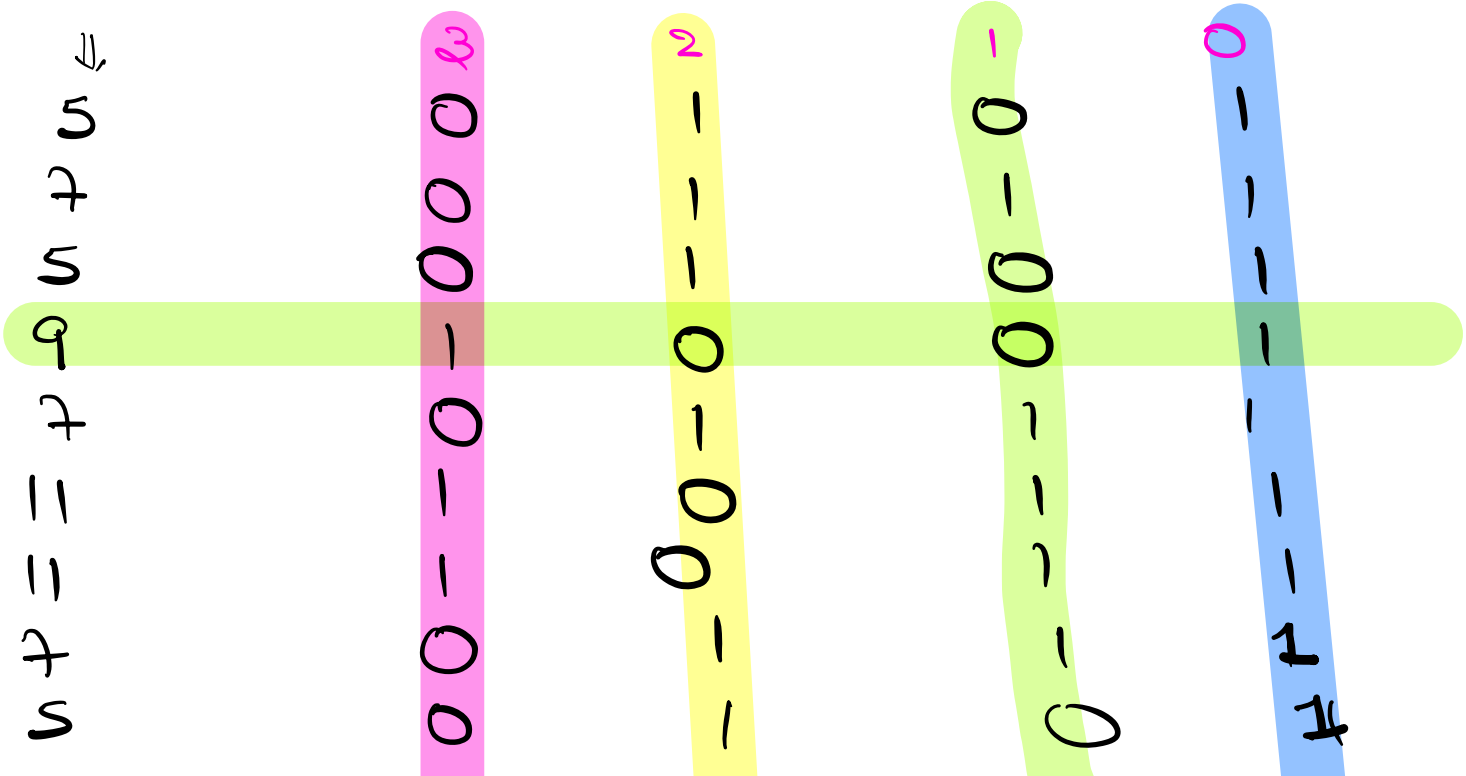
T.C. = O(N²)

2) Using HashMap

T.C. = O(N)
S.C. = O(N) ✗

3) Optimised

A = [5, 7, 5, 9, 7, 11, 11, 7, 5, 11]



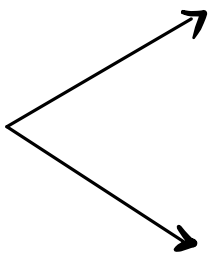
11 | 1 | 0 | 1 | 1

4(3x+1) 6(3x) 6(3x) 10 = (9+1)

9 ⇐ 1 0 0 1

Repeating Elements → 3 times.

Count of set bits on any ith pos.



3x(x + 0 = 3x

3x(x) + 1 = 3x+1

⇓

ith bit of unique no. is set.

Code

```
int ans = 0;

for (i=0; i<32; i++) { // Bits
    int count = 0;

    for (j=0; j<N; j++) { // Array Elements
        if ((A[j] & (1<<i)) > 0) {
            count ++;
```

```
    }
  }
  if(count%3==1){
    ans = ans | (1<<i);
  }
}
```

Set ith bit in ans.

T.C. = O(N)

Given an integer array of size N where every no. occurs twice except two elements. Find those two unique elements.

A = [4, 5, 4, 1, 6, 6, 5, 2]

Ans = [1, 2]

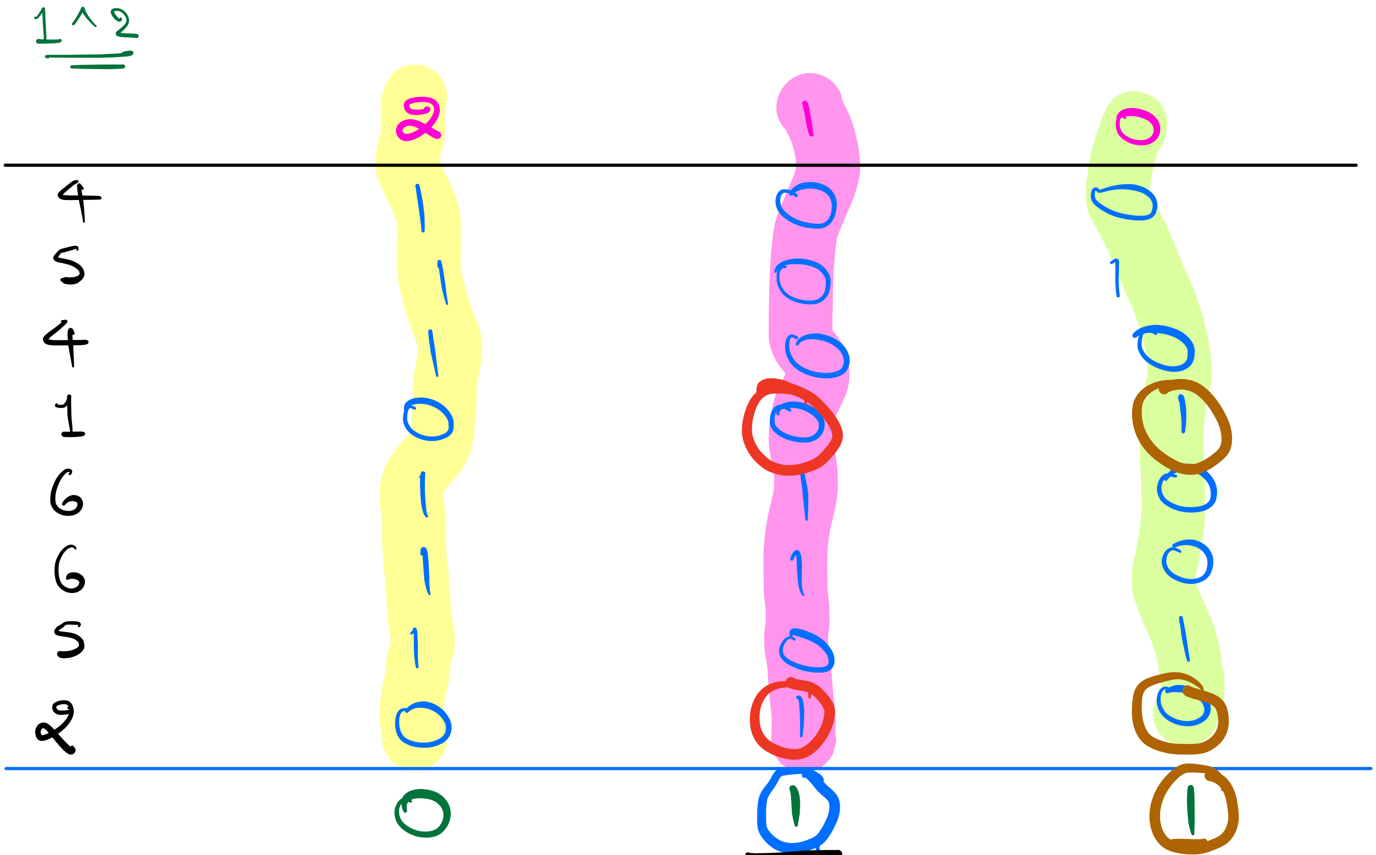
Sol^n) Will the answer be XOR of all elements ??

~~4^5^4^1^6^6^5~~^2 = 1^2

If the XOR of 2 no's is 7. Can we find the 2 no's.

(2^5) , (1^6) , (3^4) = 7

A = [4, 5, 4, 1, 6, 6, 5, 2]



This bit will be different in both unique no's

For 2 different no's a & b

Can a^b ever be 0 ??

If a & b are different no's
Atleast 1 bit in the no.
would be different
&
Hence in the XOR, atleast 1 bit would be

set.

Let's divide the entire array into 2 groups.

Bit 1 is set

{2, 6, 6}

⇓

Unique Element 1 ?!

XOR of all elements

Bit 1 is unset

{1, 4, 4, 5, 5}

⇓

Unique Element 2 ?!

XOR of all elements

Steps

1) XOR of all the numbers.

2) We identify the position of any set bit in the resultant XOR

3) We divide all the numbers into two groups acc to the set bit identified in step 2.

4) Both the unique elements will be a

part of different group

5) Each group now becomes 0 1 of today's class

6) We take XOR of each group individually to find the 2 unique elements.

Code

```
xor_result = 0;
for (i=0; i<N; i++){
    xor_result = xor_result ^ A[i];
}

int index = 0;
while (index <32){
    if((xor_result & (1<<index))>0){
        break;
    }
    else{
        index ++;
    }
}
int xor1 = 0;   { Group where index bit is set }
int xor2 = 0;   { Group where index bit is unset }
```

```
for (i=0; i<N; i++) {
        // Unset
    if((A[i] & (1<<index))==0) {
        xor2 = xor2^A[i];
    }
    else {
        xor1 = xor1^A[i];
    }
}
return {xor1, xor2};
```

T.C. = O(N)

Given an array of size N.
find the two indices (i, j) s.t.

1) $i != j$

2) (A[i] & A[j]) is maximum.

A = { 5, 4, 6, 8, 5 } (indices 0 1 2 3 4)

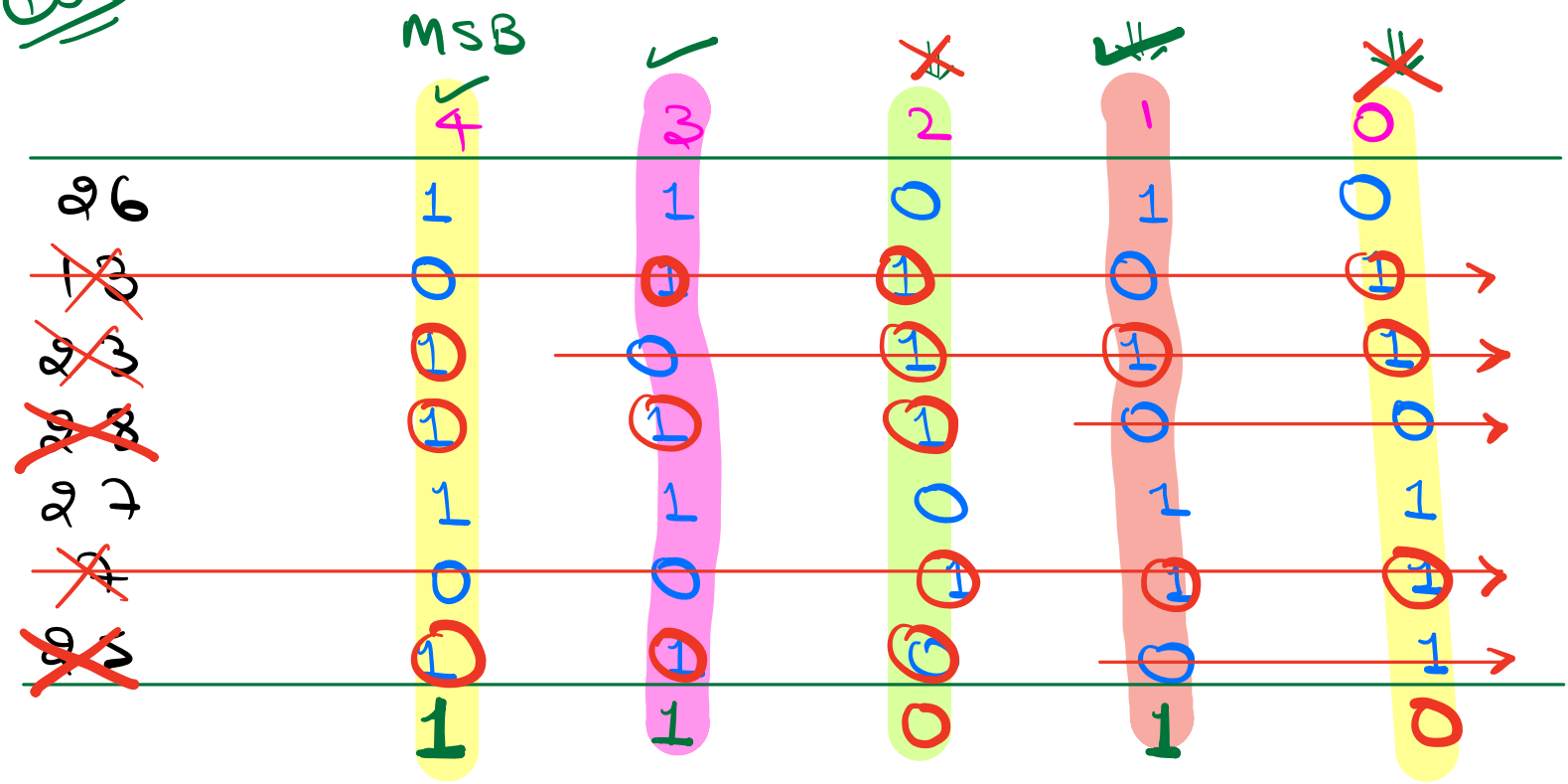
5 & 5

i	j	
0	1	5 & 4
0	2	5 & 6
0	3	5 & 8
.	.	
.	.	

1 2 4 8 6

1 3 4 8 8

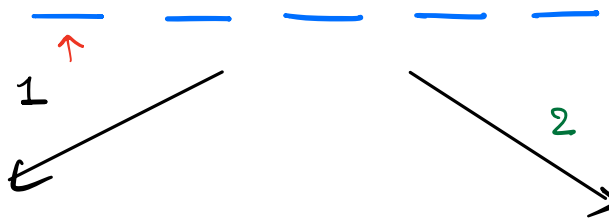
Sol^n $A = [26, 13, 23, 28, 27, 7, 25]$



$a \& b = 1$

Eliminate = Replace with 0

4 → 0



10000 > 01111

2^4

$2^3 + 2^2 + 2^1 + 2^0$

$= 2^4 - 1$

MSB always overpowers all other bits

* If at the end, more than 2 no's are remaining, you can select any 2 no's.

⇓ If x elements are left

Count of pairs whose AND is maximum.

25, 27, 29, 41

(25,27), (25,29), (25, 41)

(27,29), (27, 41), (29, 41)

Count = $^{x}C_2$ = $\frac{(x)(x-1)}{2}$

Code

```
for (i = 31; i >= 0; i -> 4

    int count = 0;
    for (j = 0; j < N; j++) {
        if ((A[j] & (1 << i)) > 0) {
            count++;
```

```
        }
    }
    if (count >= 2) {
        for (j=0; j<N; j++) {
            if ((A[j] & (1<<i)) == 0) {
                A[j] = 0;
            }
        }
    }
}

int i
int j
```

Iterate ⇒ 1st non zero element = i
2nd non zero element = j