"Everything is easy when you are busy,
but nothing is easy when you are lazy"

– Swami Vivekananda

SCALER

Bit Manipulation - 2

TABLE OF CONTENTS

Given arr[N] where every no. is present two times except one unique no. Find that unique number.

arr[] → [4 , 5 , 5 , 1 , 6 , 4 , 6]

Brute Force :- Take elements one by one and count their frequency.

$O(n^2)$ T.C.
$O(1)$ S.C.

Optimised approach :-

XOR all elements.

4 ^ 5 ^ 5 ^ 1 ^ 6 ^ 4 ^ 6

```
xor = 0
for (i → 0 to n-1) {
    xor = xor ^ arr[i]
}
return xor
```

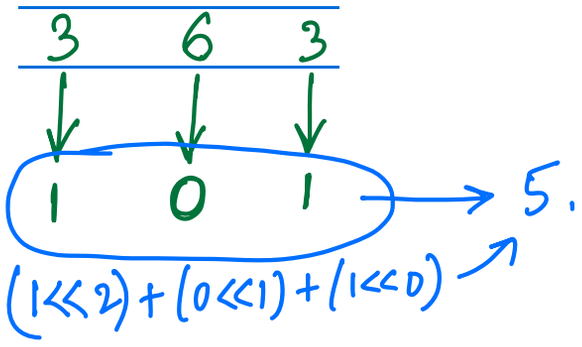
$O(n)$ T.C.
$O(1)$ S.C.

Another Approach

arr[] →

2	3	5	6	3	6	2

	2^{nd}	1^{st}	0^{th}
2 →	0	1	0
3 →	0	1	1
5 →	1	0	1
6 →	1	1	0
3 →	0	1	1
6 →	1	1	0
2 →	0	1	0



Pseudo Code

```
ans = 0
for (i = 0 to 31) {
    cnt = 0
    for (j -> 0 to arr.size()-1) {
        if ((arr[j] & (1 << i)) > 0)
            cnt++
    }
    if ((cnt & 1) == 1)
        ans |= (1 << i)
}
return ans
```

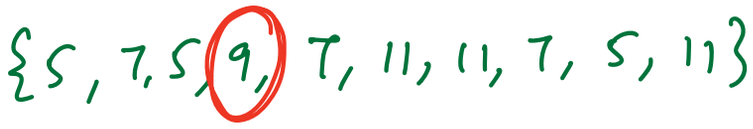
O(n) T.C.
O(1) S.C.

2

Given an integer array of size N, where all the elements occur thrice except one element. Find that unique element.

$(1 \le N \le 10^6)$

arr[] → [4 , 5 , 5 , 4 , 11 , 6 , 6 , 4 , 5 , 6]
0 1 2 3 4 5 6 7 8 9



Idea 1

Two loops to count the freq of each arr[i].

$O(n^2)$ T.C.
$O(1)$ S.C.

Here, xor will not help.

$a \wedge b \wedge a \wedge a \longrightarrow a \wedge b$

Idea 2

arr[] →	[5	7	5	4	7	11	11	9	11	7	5	4	4]
	0	1	2	3	4	5	6	7	8	9	10	11	12

	3	2	1	0
5 →	0	1	0	1
7 →	0	1	1	1
5 →	0	1	0	1
4 →	0	1	0	0
7 →	0	1	1	1
11 →	1	0	1	1
11 →	1	0	1	1
9 →	1	0	0	1
11 →	1	0	1	1
7 →	0	1	1	1
5 →	0	1	0	1
4 →	0	1	0	0
4 →	0	1	0	0
	4	9	6	10
%3	1	0	0	1 → 9

O(n) T.C.
O(1) S.C.

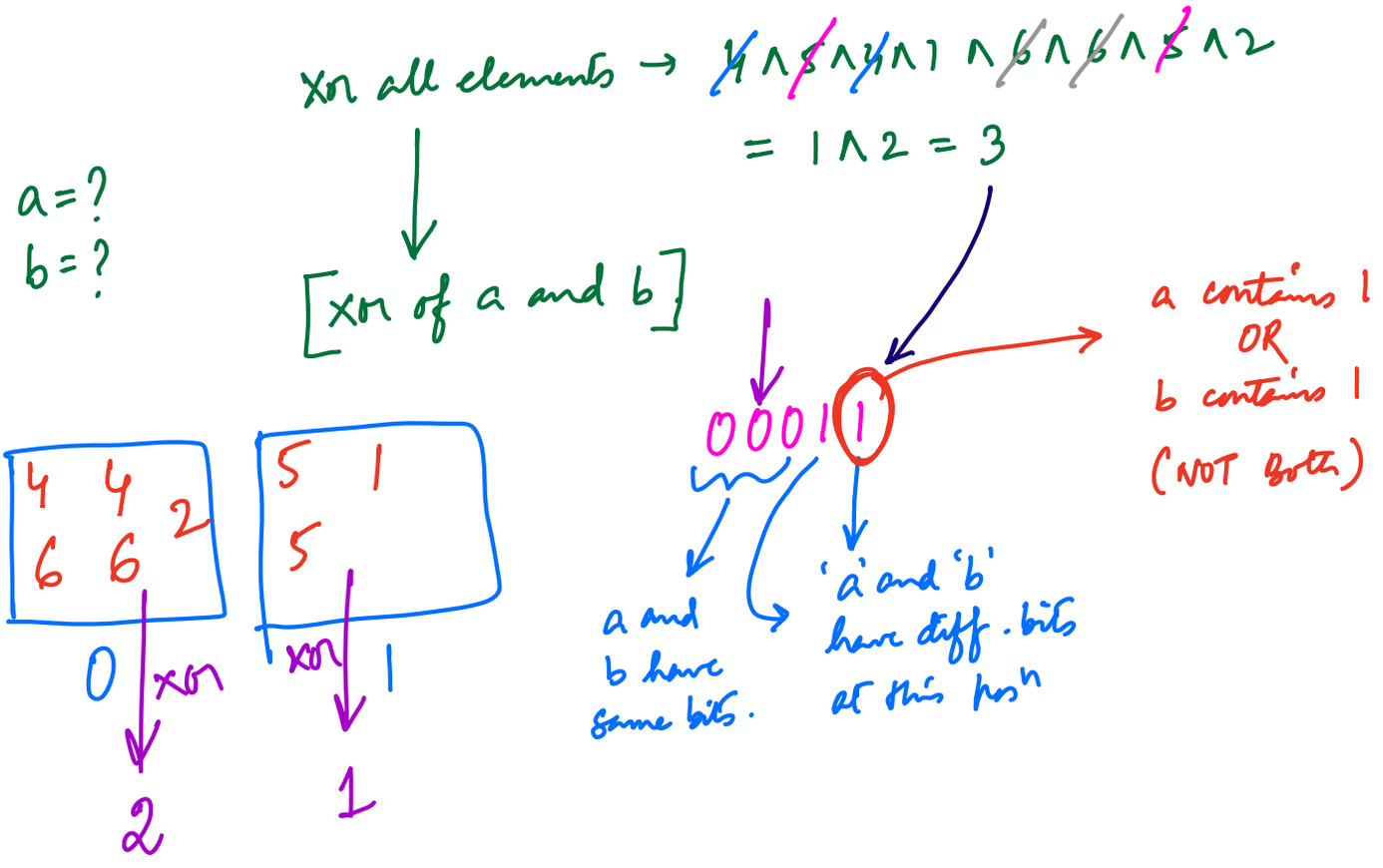
Question

Given an integer array of size N, where all elements repeat twice except two. Find those two elements.

{T.C - O(N)
S.C - O(1)}

arr[] → [4 , 5 , 4 , 1 , 6 , 6 , 5 , 2]

Ans → 1, 2 .



$\{3, 6, 5, 11, 6, 4, 14, 4, 5, 3\}$

Step 1 : Take XOR of all the elements.

$3 \wedge 6 \wedge 5 \wedge 11 \wedge 6 \wedge 4 \wedge 14 \wedge 4 \wedge 5 \wedge 3$

$= 5 \quad (0101)_2$

Step 2 : Find any set bit position in val.

$11 \rightarrow 1011$
$14 \rightarrow 1110$

pos = 2.

Step 3 : Split the array on the basis of posth bit.

(0) $x = 3 \wedge 11 \wedge 3 = 11$

(1) $y = 6 \wedge 5 \wedge 6 \wedge 4 \wedge 14 \wedge 4 \wedge 5 = 14$

Step 4 : Print the unique numbers.

11
14

```
xorall = 0
for (i → 0 to n-1) {
    xorall ^= arr[i]
}
pos = 0
for (pos → 0 to 31) {
    if (checkBit(xorall, pos))
        break;
}
x = 0
y = 0
for (i → 0 to n-1) {
    if (checkBit(arr[i], pos))
        y ^= arr[i]
    else
        x ^= arr[i]
}
print(x)
print(y)
```

O(n) T.C.
O(1) S.C.

[Break till 10:38 PM]

Maximum AND Pair

Given n +ve array elements. Find the maximum value of (arr[i] & arr[j]) where i ! = j (indices must be different for 2 no.)

arr[] → [27 18 20]
0 1 2

27 → 11011
18 → 10010
18 ← 10010

27 → 11011
20 → 10100
16 ← 10000

18 → 10010
20 → 10100
16 ← 10000

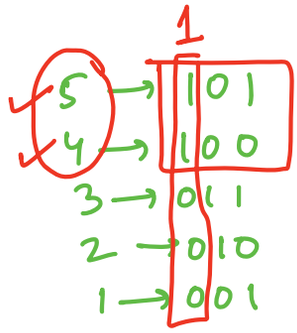
Ans → 18.

{21, 18, 24, 17, 16}

21 & 17 → 17
24 & 21 → 16
17 & 16 → 16
24 & 18 → 16

{5, 4, 3, 2, 1}

5 & 4 → 4
1 & 2 → 0
1 & 4 → 0
4 & 3 → 0



Idea 1

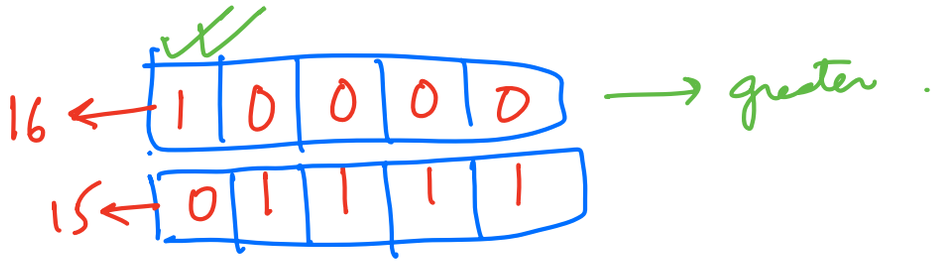
All pairs AND, take max.

$O(n^2)$ T.C.
$O(1)$ S.C.

Idea 2

AND → 1's

1s towards the left are better than the right.
(MSB) (LSB)



{26, 13, 23, 28, 27, 7, 25}

MSB

43210

26 → 11010
13 → 01101
23 → 10111
28 → 11100
27 → 11011
7 → 00111
25 → 11001

1

26 → 11010
13 → 01101
23 → 10111
28 → 11100
27 → 11011
7 → 00111
25 → 11001

110

43210

26 → 11010
13 → 01101
23 → 10111
28 → 11100
27 → 11011
7 → 00111
25 → 11001

11

26 → 11010
13 → 01101
23 → 10111
28 → 11100
27 → 11011
7 → 00111
25 → 11001

1101

26 → 11010
13 → 01101
23 → 10111
28 → 11100
27 → 11011
7 → 00111
25 → 11001

11010 → 26.

Code

```
ans = 0
for (i -> 31 to 0) {
    cnt = 0
    for (j -> 0 to n-1) {
        if ((arr[j] & (1<<i)) > 0)
            cnt++
    }
    if (cnt >= 2) {
        ans |= (1<<i)
        for (j -> 0 to n-1) {
            if ((arr[j] & (1<<i)) == 0)
                arr[j] = 0
        }
    }
}
return ans
```

O(n) T.C.
O(1) S.C

Q5) Calculate the nr. of pairs for which Bitwise AND is maximum.
(Google)

Nr. of ways to choose 2 elements
cnt of C elements.
↳ nr. of remaining elems ≥ ans. in the array.

$$cnt_{C_2}$$

$$= \frac{cnt * (cnt-1)}{2}$$

O(n) T.C.
O(1) S.C.

```
if(ans<=(1<<31)-1 && ans>=(1<<31))
```

$2^{31}-1$

-2^{31}