Welcome ☺

Agenda : BM 2
3 questions.

Q Given an integer array where every number occurs twice except one element. Find the unique element

App1 ⇒ XOR of all the elements

T.C ⇒ O(N)
S.C ⇒ O(1)

App2 ⇒ Use hashset / map and store freq.

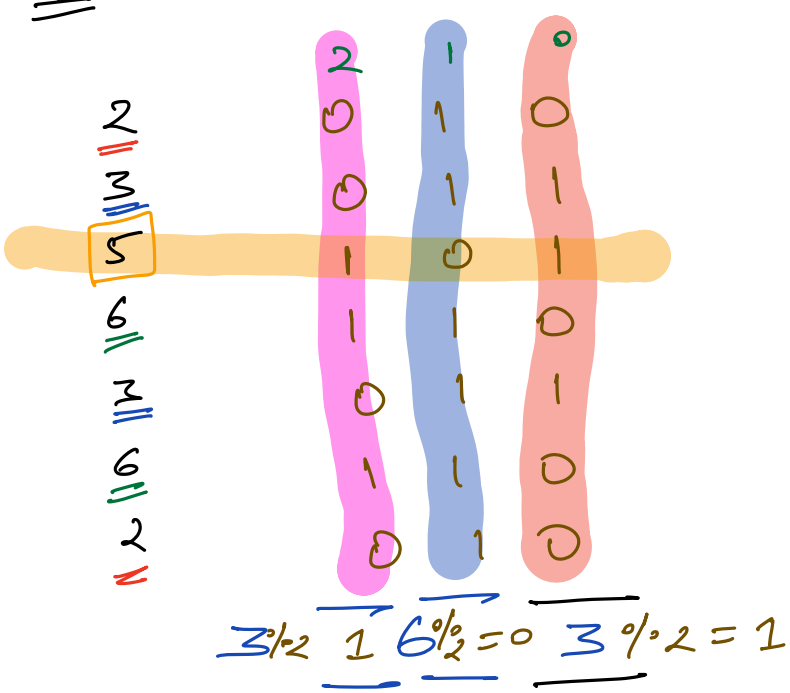
T.C → O(N)
S.C ⇒ O(N)

App3 ⇒ Sort the array and check

T.C ⇒ O(NlogN)
S.C ⇒ O(1)

App4. V. Interesting Soln

eg: A : [2 3 5 6 3 6 2]



1 0 1 = 5

Count of set bits on any pos^n $\Rightarrow$

- Odd $\Rightarrow 2x+1$ → unique element
- Even $\Rightarrow 2x$ → pairs repeating numbers

Code

```
for ( i=0 ; i< 32 ; i++)
{
    count = 0
    for ( j=0 ; j<N ; j++)
    {
        if ( checkbit ( arr[j] , i ) == true )
        {
            count ++
        }
    }
    if ( count % 2 == 1)
    {
        // ith bit of unique element is set
        ans = ans | (1<< i)
    }
}
return ans
```

Q Given an integer array where every number occurs thrice except one element. Find the unique element

eg: [4 5 5 4 1 6 6 4 5 6]

Sol^ Brute force.

→ Use two for loops and count occurence of each number.

T.C O(N^2)
S.C O(1)

Sol^2 XOR of all number

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

1^4^5^6

Sol^2 Use hashset/map and store freq.

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

Sol^3 Sort the array and check

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

Sol^n 4 Since every element occurs thrice.

count % 3 will give us unique element.

```
for ( i=0 ; i<32 ; i++)
{
    count = 0
    for ( j=0 ; j<N ; j++)
    {
        if ( checkbit ( arr[j] , i ) == true )
        {
            count ++
        }
    }
    if ( count % 3 != 0)
    {
        // ith bit of unique element is set
        ans = ans | (1<<i)
    }
}
return ans
```

T.C ⇒ O(N*32)

S.C ⇒ O(1)

Extensions

1) Every ele. occurs 4 times except 1. XOR

2) Every ele. occurs thrice except one ele. which is repeating twice. above sol^n will work.

Google

Q Given N elements where every element occurs twice except 2 unique element. Find the two unique elements.

eg: 3 6 4 4 3 8 o/p → 6, 8.

Sol^ Brute force.

→ Use two for loops and count occurence of each number.

T.C O(N^2)
S.C O(1)

Sol^2 XOR of all number ✗

3 6 4 4 3 8

6 ^ 8

Sol^2 Use hashset/map and store freq.

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

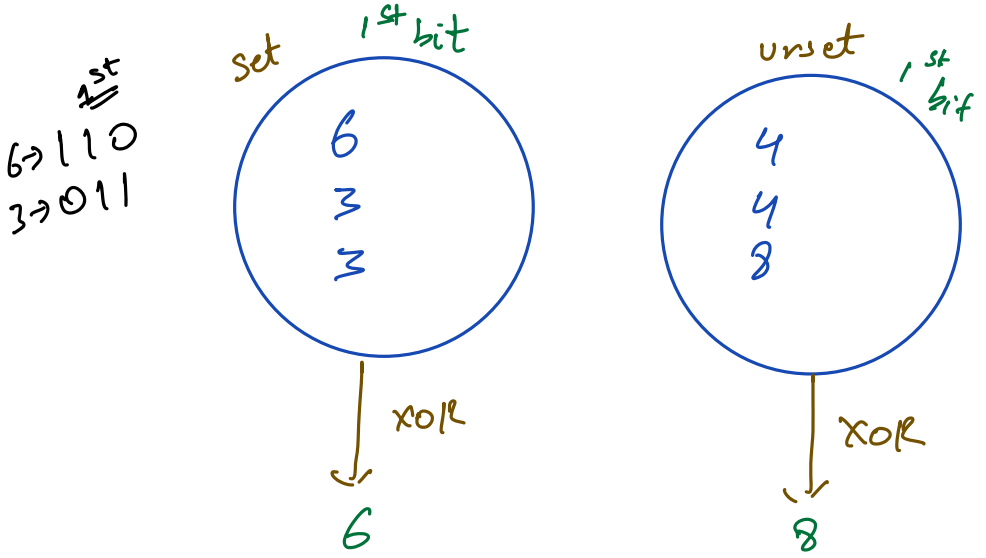
Sol^3 Sort the array and check

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

6 ^ 8

	3	2	1	0	
6	0	1	1	0	
8	1	0	0	0	
	1	1	1	0	= 14

⇒ You can use 1st/2nd/3rd bit to differentiate b/w 2 unique numbers in this case (6^8)



1) XOR of all elements

Code 2) Set bit in XOR ans.

```
posn = -1
for( i=0 ; i< 32 ; i++)
{
    if( checkBit( ans, i) == true)
    {
        posn = i   // differentiator.
        break;
    }
}
```

3) Split array into two baskets using posn

```
set =0      unset =0
for( i=0 ; i<N ; i++)
{
    if( checkBit( arr[i], posn) == true )
    {
        set = set ^ arr[i]
    }
    else {
        unset = unset ^ arr[i]
    }
}
print( set )    print( unset)
```

T.C ⇒ O(N+N)
≃ O(N)

S.C ⇒ O(1)

Q Given an array A of N integers. Find the sum of bitwise XOR of all pairs of numbers in the array.

eg: [1 2 3]

{ 1 2 } → 3

{ 2 3 } → 1

{ 1 3 } → 2

6 ⇐ o/p

1 → 0 1

2 → 1 0

3 → 1 1

^ 0 1

Sol^ Contribution Tech

∀ i^{th} bit

(set ^ unset) = 1 — i^{th} bit

X elements with i^{th} bit set

Y elements with i^{th} bit unset

0 0 0 → 3

0 0 → 2

3 * 2

X * Y pairs whose i^{th} bit is set.

(1<<i == 2^i)

Contribⁿ of i^{th} bit → (X * Y) * (1 << i)

[1 2 3]

1 → 0 1

2 → 1 0

3 → 1 1

			X	Y
0^{th} bit	$\Rightarrow$	$(2*1)*2^0 = 2$	1, 3	2
1^{st} bit	$\Rightarrow$	$(2*1)*2^1 = 4$	2, 3	1

$2 \oplus 4 = 6$

[11 12 13]

		3 2 1 0
11	→	1 0 1 1
12	→	1 1 0 0
14	→	1 1 1 0

			X	Y
0^{th} bit	$\Rightarrow$	$(2*1)*2^0 = 2$	11	12, 14
1^{st} bit	$\Rightarrow$	$(2*1)*2^1 = 4$	11, 14	12
2^{nd} bit	$\Rightarrow$	$(2*1)*2^2 = 8$	12, 14	11
3^{rd} bit	$\Rightarrow$	$(3*0)*2^3 = 0$	12, 14, 11	—

14