Bitwise Operators.

{ &, |, ^, ~, <<, >> }

a	b	a & b	a | b	a ^ b (XOR)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Negate (~)

a	~a
0	1
1	0

Properties.

1) $a \& b = b \& a$
   $a \mid b = b \mid a$
   $a \wedge b = b \wedge a$
   } Commutative

2) $a \wedge b \wedge c = a \wedge (b \wedge c) = (a \wedge b) \wedge c = a \wedge (c \wedge b)$
   } Associative (&, |)

3) $a \wedge 0 = a$

$1 \wedge 0 = 1$

$0 \wedge 0 = 0$

$a = 10$

↳ 1010

$a \wedge 0 \Rightarrow 1010$

$\wedge\ 0000$

$1010 \Rightarrow a$

---

4) $a \wedge a = 0$

Quiz $a = 15$

$a << 2 \Rightarrow 15 \times 2^2 = 60$

↳ left shift operator.

$0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \Rightarrow 15$

$<<1$ ↙ ↙ ↙ ↙ ↙ ↙ ↙ ↙

$0\ 0\ 0\ 1\ 1\ 1\ 1\ 0 \Rightarrow 30$

lost ↙ ↙ ↙ ↙ ↙ ↙ ↙ ↙

$0\ 0\ 1\ 1\ 1\ 1\ 0\ 0 \Rightarrow 60$

$a << N \Rightarrow a * 2^N$ [Assuming NO overflow].

$2^N \Rightarrow a = 1$

$1 << N \Rightarrow 1 \times 2^N = 2^N$

$1 << N = 2^N$

↳ pow(2, n)

↳ TC: O(log N)

TC: O(1)

Quiz $a = 29$

$a >> 2 \Rightarrow 29/2^2 = 29/4 = 7$

↳ Right shift operator.

$$a >> N = \frac{a}{2^N}$$ {No Overflow}

$a = 60 \xrightarrow{>>} 30 \xrightarrow{>>} 15 \xrightarrow{>>} 7 \xrightarrow{>>} 3 \xrightarrow{>>} 1 \xrightarrow{>>} 0$

Q.1 Given a number N, check if i^{th} bit is set in N or not.

$i = 2$

	7	6	5	4	3	2	1	0
N ⇒	0	0	1	1	0	1	0	0

⇒ True.

						i		
	7	6	5	4	3	2	1	0
N ⇒	0	0	1	1	0	1	0	0
n ⇒	0	0	0	0	0	1	0	0
&	0	0	0	0	0	1	0	0

$a \& 1 = 1 \Rightarrow a = 1$

$a \& 1 = 0 \Rightarrow a = 0$

$n \Rightarrow 1 << i$

```
bool checkBit (N, i) {
    return N & (1<<i);
}
```

TC: O(1)

```
bool checkBit(N, i) {
    if ((N >> i) & 1)
        return true;
    return false;
}
```

* N < Odd / Even

N = _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

- 1 ⇒ N is ODD
- 0 ⇒ N is even.

even — $2^0 = 1$

if (N & 1) ⇒ N is Odd.

Q.2 Given an Array, where all numbers appears twice except 2 numbers which appears once. Find these 2 single no's.

Amazon MS ...

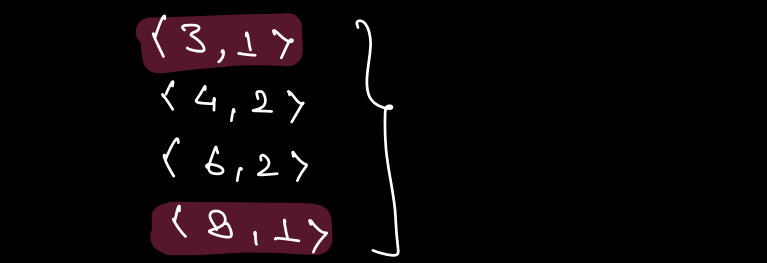
A: {3, 4, 6, 4, 6, 8} ⇒ 3, 8

A: {4, 9, 8, 9} ⇒ 4, 8.

A: {1, 2} ⇒ 1, 2.

1) HashMap.

A : { 3, 4, 6, 4, 6, 8 }



TC : O(N)
SC : O(N)

2) Sorting.

{ 3, 4, 6, 4, 6, 8 }

↓ Sort

{ 3, 4, 4, 6, 6, 8 } ⇒ 3, 8.

i

TC : O(NlogN)
SC : Depends on sorting Algo.

3. A: {3, 4, 6, 4, 6, 8}

↓ XOR

3^8 = 11

3: 0011
8: 1000
^
1011

A: {a, b, c, b, a, S_1, c, S_2}

↓ XOR

$S_1 \wedge S_2$

Quiz can $S_1 \wedge S_2 = 0$? NO

$\Downarrow$

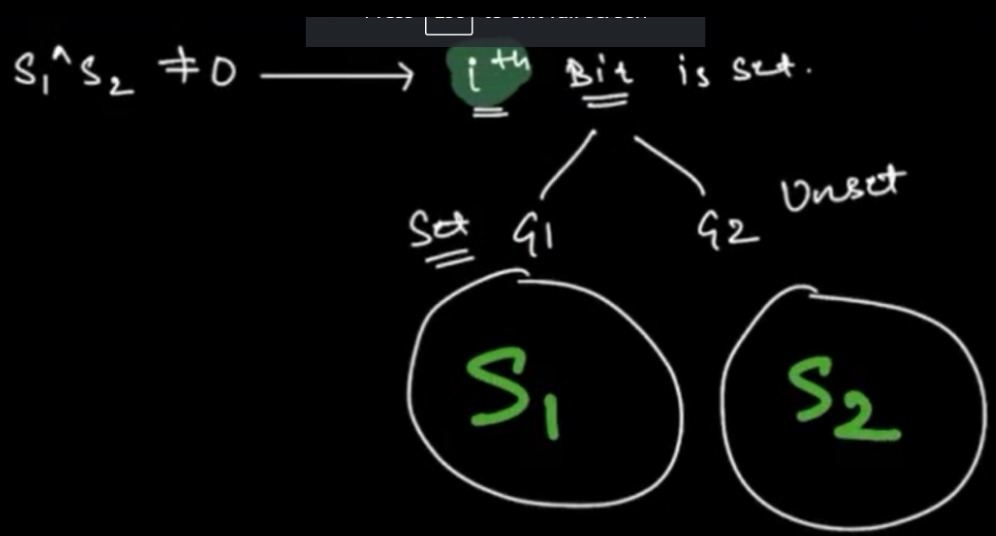
$S_1 \neq S_2$

$\Rightarrow S_1 \wedge S_2$ can't be ZERO

$\Rightarrow S_1 \wedge S_2$ will have At least 1 set Bit.

{3, 6, 4, 4, 6, 8}

3^8 = 0011
1000
^
$S_1 \wedge S_2 \Rightarrow$ 1011



A : { 2, 1, 3, 3, 12, 2 }

⟶

$2 \wedge 1 \wedge 3 \wedge 3 \wedge 12 \wedge 2 = 1 \wedge 12$

$S_1 = 1 \Rightarrow 0001$

$S_2 = 12 \Rightarrow 1100$

1101

3 2 1 0

A : { 2, 1, 3, 3, 12, 2 }

2 → 0010
1 → 0001
3 → 0011
12 → 1100

G_1 (No's with 0^{th} Bit Set)

1, 3, 3

↓ XOR

1

G_2 (No's with 0^{th} Bit UnSet)

2, 12, 2

↓

12

3rd Bit

A : { 2, 1, 3, 3, 12, 2 }

G_1 (No's with 3^{rd} Bit Set)

12

↓

12

G_2 (No's with 3^{rd} Bit UnSet)

2, 1, 3, 3, 2

↓ XOR

1

Ex

A: {10, 8, 8, 9, 12, 9, 6, 11, 10, 6, 12, 17}

$xor_A = 11 \wedge 17$

```
 01011
^10001
-------
 11010
 43210
```

10 → 01010
8 → 01000
9 → 01001
12 → 01100
6 → 00110
11 → 01011
17 → 10001

3rd Bit

G_1 (Nos with 3rd Bit Set)	G_2 (Nos with 3rd Bit UnSet)
ans1 = 0^10^8^8^9^12 ^9^11^10^12 = 11	ans2 = 0^6^6^17 = 17

Steps

1) Take XOR of complete array (xorA)
2) Find any set bit position in xorA (Pos)
3) Based on position of set bit (Pos), split the array elements in two groups. (Using CheckBit fun on Pos)
4) Take XOR of 2 groups separately.
5) Return 2 nos.

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

// find any set bit pos in xorA. → $\log_2(MAX)$

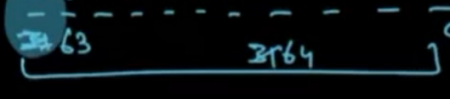
```
pos
s1 = 0
s2 = 0
for(i=0; i<N; i++){
    if(CheckBit(A[i], pos))
        s1 = s1 ^ A[i]
    else
        s2 = s2 ^ A[i]
}
return s1, s2;
```
O(N)

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

SC: $O(1)$

Change in Datatype this can be vary. Generally to store N value , we need N = Log(N) + 1 bits.

// find any set bit pos in xorA. → 32 / 64



Q.3 Given an Array, every element appears thrice except 1 element that appears once. Find the single number.

Google
Amazon
MS|LI

A: {5, 7, 5, 4, 7, 11, 11, 9, 11, 7, 5, 4, 4}

$a \wedge a = 0$

$a \wedge a \wedge a = a$

HINT: Can we find how many times any specific bit is set in array element.

A: {5, 7, 5, 4, 7, 11, 11, 9, 11, 7, 5, 4, 4}

	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	← Single No.
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	

Count of No's with 0^{th} bit set: = 10

Count of No's with 1^{st} bit set:

$9 \% 3 = 0$

3rd Bit

4 9 6 10 → $\% 3 \neq 0 \Rightarrow 0^{th}$ Bit in ans.

(4 % 3 = 1, 9 % 3 = 0, 6 % 3 = 0, 10 % 3 = 1)

ans = 1001 ⇒ 9

Steps

Iterate on all the bit positions :- ($\log_2(\max(Arr))$)

for i^{th} bit position :-

{ iterate on array & check if ith bit is SET or not.
if SET ⇒ count++

if (count % 3 == 0) {
i^{th} bit is NOT set in ans.
}
else {
i^{th} bit is set in ans.
}

⇒ Code

$\log_2(MAX(Arr))$

```
ans = 0
for(i = 0; i < 32; i++) {
    count = 0;
    for(j = 0; j < N; j++) {
        if(checkBit(A[j], i)){
            count++;
        }
    }
    if (count % 3 != 0)
        ans | (1<<i); // Set ith bit in ans.
}
return ans;
```

TC: $O(\log_2(\max(A[i])) * N)$

SC: $O(1)$

Q.4 Given an Array of size N containing all the elements from 1 to N+2, except 2 elements find these 2 elements.

A: [3, 6, 1, 4] ⇒ 2, 5.

N = 4

↓

[1, 2, 3, 4, 5, 6]

A: [3, 6, 1, 4]

[1, N+2] ⇒ [1, 2, 3, 4, 5, 6]

→ Take xor of array elements (xorA)

xor = 3^6^1^4

→ Take xor with all the elements [1, N+2] of xorA.

xor = 2^5

⇒ Apply logic of Q.2.

— * —