Bit Manipulation :-

1) And, or & xor properties :-

A	B	A ^ B	A | B	A & B
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	0	1	1

2) Properties :-

a) $A \wedge 0 = 0$

b) $A \wedge A = A$

c) $A \mid 0 = A$

d) $A \mid A = A$

e) $A \,\&\, 0 = A$

f) $A \,\&\, A = 0$

g) Even-odd :-

$n \wedge 1 \longrightarrow 1 \longrightarrow$ odd.

$\longrightarrow 0 \longrightarrow$ even.

ex :-

5 :- 101 ^ 001 = 100

6 :- 110 ^ 001 = 111

h) Commutative :-

$a \wedge b = b \wedge a$

$a \mid b = b \mid a$

$a \,\&\, b = b \,\&\, a$

i) Associative :-

$a \wedge (b \wedge c) = (a \wedge b) \wedge c$

$a \mid (b \mid c) = (a \mid b) \mid c$

$a \wedge (b \wedge c) = (a \wedge b) \wedge c$

3) Left shift & Right shift :-

a) Left Shift :- $1 << n$ → move 1 n bits to left.
$A << n$ → move A, n bits towards left.

$A << 1 = A \times 2^1$
$A << n = A \times 2^n$

$1 << n = 1 \times 2^n = 2^n$

b) Right shift :- $A >> i$ → move A, i bits towards right.

$A >> 1 = A/2$

$A >> n = A/2^n$

Arbitrary left shift → Randomly large #.
Arbitrary right shift → Reaches to zero.

→ Power of Left shift :-

And :- To check if ith bit is set or not :-

$A \& (1 << i)$ → 1 → if ith bit is set.
0 → if unset.

or :- To set the ith bit.

$A | (1 << i)$ = ith bit is always 1.

xor :- To flip the ith bit.

$A \wedge (1 << i) = 1$ → if $A \& (1 << i) = 0$;
0 → if $A \& (1 << i) = 1$;

Q:- Single # 3 :- Find two #s that are not repeated in an array :-
- Rest of the elements occur twice.

1) Find xor of all elements in the array.
Let the two answers be x & y.
2) Set bit in xor, represents bits where x & y are different.
3) Divide the array according to leftmost set bit in xor & find the x & y separately.

Q:- Single # 2 :- Given an array, every element occurs thrice, except for one which occurs just once.
- Find the element.

ex:- a = [2, 2, 2, 3, 5, 5, 5]

```
2  0 1 0
2  0 1 0
2  0 1 0

3  0 1 1

5  1 0 1
5  1 0 1
5  1 0 1
-------
3  4 4
```

```
0 1 0
0 1 1
-----
0 0 1
1 0 1
-----
1 0 0
```

a = [6, 6, 6, 3]

```
6  1 1 0
6  1 1 0
6  1 1 0

3  0 1 1
-------
3  4 1
```

```
1 1 0
0 0 0
-----
1 1 0
0 1 1
-----
1 0 1
```

```
ans = 0;
for(i=0 to 31)
    count = 0;
    for(int num : A)
        if(num & (1<<i)) count++;
    if(count % 3 != 0) ans |= (1<<i);
return ans;
```

Q:- Find nth magic # :-

```
5     0 0 1
25    0 1 0
30    0 1 1
125   1 0 0
130   1 0 1
```

nth magic # =

```
num = 0;
for(i = 0 to 31)
    if(n & (1<<i)) num += 5<<i;
```

Q:- Help from Sam :-

7 → 1 → 2 → 4 → 5 → 6 → 7
(1 +1; 2 +3; 4 +1; 5 +1; 6 +1)

111

5 → 101 : 1 → 2 → 3 → 6 → 7
(1 +1; 2 +1; 6 +1)

6 → 110 : 1 → 2 → 3 → 6
(1 +1; 3 +1)

# of bits in the binary representation of n.

Q-4 Sub-array with odd 1-s

[0, 0, 1, 1, 0]
 0 1 2 3 4   n = 5

	0	1	2	3
0	1	2	3	4

10

# of sub-arrays b/w i & j :=

n = (j - i + 1)

ans = n × (n+1)/2

0 0 1
0 1 0
1 0 0
0 1 1
1 1 0
1 1 1