PSP → 78% → 80% by Thursday

Bit Operations

1 → true, set bit

0 → false, unset bit

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

same same

puppy shame

A	~A (NOT)
0	1
1	0

Q → 45 | 10

45 → 1 0 1 1 0 1

10 → 0 0 1 0 1 0

OR

1 0 1 1 1 1

5 4 3 2 1 0

$2^5 + 2^3 + 2^2 + 2^1 + 2^0 = 47$ (Ans)

Properties

Eg → A = 5 (1 0 1)

1) A | 0 = A

2) A | A = A

3) A ^ 0 = A

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

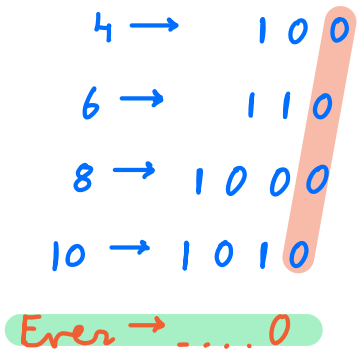
4) A ^ A = 0

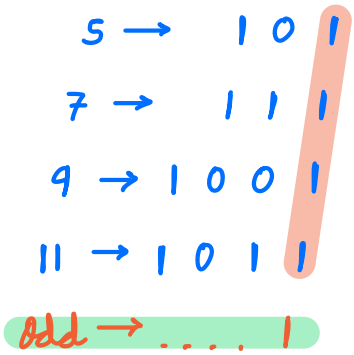
```
  1 0 1
^ 1 0 1
-------
  0 0 0
```

5) A & 0 = 0

6) A & A = A

7) Odd / Even





check last bit →

A & 1 → 0 even number
1 odd number

A = 5

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

A = 20

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

---

8) Commutative Property

A & B = B & A

A | B = B | A

A ^ B = B ^ A

9) Associative Property

(A & B) & C = A & (B & C)

(A | B) | C = A | (B | C)

(A ^ B) ^ C = A ^ (B ^ C)

Q → A ^ B ^ A ^ D ^ B = (A ^ A) ^ (B ^ B) ^ D

= 0 ^ 0 ^ D = 0 ^ D = D (Ans)

Q → 1 ^ 3 ^ 5 ^ 3 ^ 2 ^ 1 ^ 5 = (1 ^ 1) ^ (3 ^ 3) ^ (5 ^ 5) ^ 2

= 0 ^ 0 ^ 0 ^ 2 = 0 ^ 2 = 2 (Ans)

Q → Given an integer array where every number occurs twice except one number. Find that unique number.

$1 \wedge 3 \wedge 5 \wedge 3 \wedge 2 \wedge 1 \wedge 5 = (1 \wedge 1) \wedge (3 \wedge 3) \wedge (5 \wedge 5) \wedge 2$

$= 0 \wedge 0 \wedge 0 \wedge 2 = 0 \wedge 2 = \underline{2}$ (Ans)

Sol → Ans = $\forall i, \wedge A[i]$

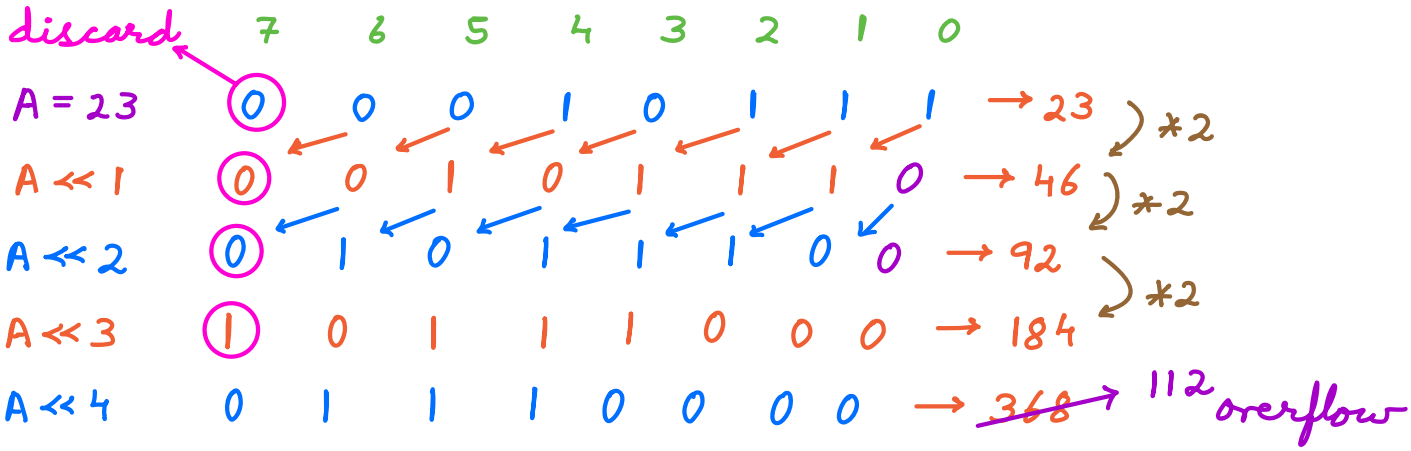
```
ans = A[0]
for i → 1 to (N-1) {
    ans ^= A[i]
}
return ans
```

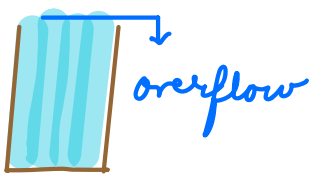
TC = O(N)

SC = O(1)

Left Shift (<<)

For explaining → 8 bit system





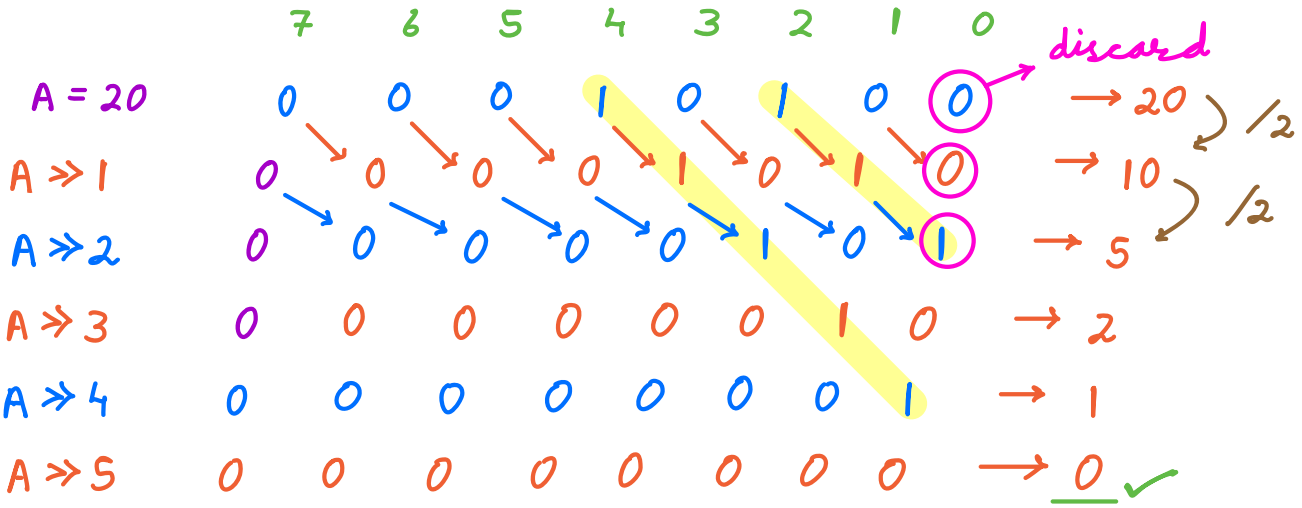
$N << 1 = N * 2$

$N << K = N * 2^K$

$1 << n = 2^n$

$N > \frac{INT\_MAX}{2^K} \Rightarrow$ overflow

Right Shift (≫)

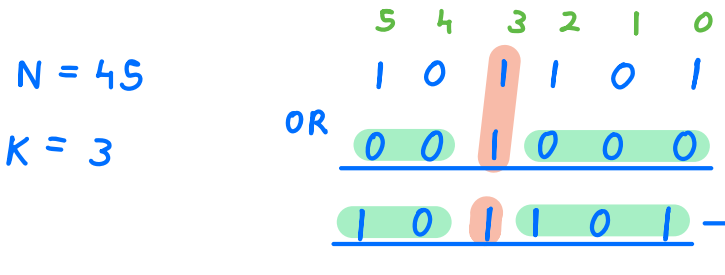


$N \gg 1 = N/2$

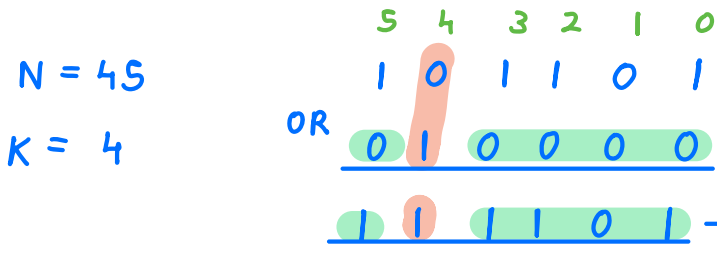
$N \gg K = N/2^K$

$(1 << K) = 2^K$ (only K^{th} bit is set)

OR $N \mid (1 << K)$ → set K^{th} bit of N



(Already 1 ⇒ no change)



(Else ⇒ $N + 2^K$)

2) XOR

$N \wedge (1 << K)$ → toggle K^{th} bit

1 → 0
0 → 1

N = 45
K = 3

	5	4	3	2	1	0
	1	0	1	1	0	1
XOR	0	0	1	0	0	0
	1	0	0	1	0	1

→ 37 (Already 1 ⇒ $N - 2^K$)

N = 45
K = 4

	5	4	3	2	1	0
	1	0	1	1	0	1
XOR	0	1	0	0	0	0
	1	1	1	1	0	1

→ 61 (Else ⇒ $N + 2^K$)

3) AND

N & (1 << K) → 2^K (K^{th} bit is set)
→ 0 (K^{th} bit is unset)

N = 45
K = 3

	5	4	3	2	1	0
	1	0	1	1	0	1
AND	0	0	1	0	0	0
	0	0	1	0	0	0

→ 2^3 (K^{th} bit 1 ⇒ result = 2^K)

N = 45
K = 4

	5	4	3	2	1	0
	1	0	1	1	0	1
AND	0	1	0	0	0	0
	0	0	0	0	0	0

→ 0 (Else result = 0)

Q → For any number N

a) check if K^{th} bit is set →

```
if ( N & (1<<K) == 0) return false
else return true
```

b) unset K^{th} bit →

N = 10
K = 1

3	2	1	0
1	0	1	0

1 0 0 0 → 8

```
if ( N & (1<<K) == 0) return N
else return N ^ (1<<K)   // set ⇒ toggle K^th bit
```

Q → Count the # set bits in N.

integer → 32 bits

```
cnt = 0
for i → 0 to (31) {        // integer; long → 64 bits
    if ( N & (1<<i) > 0)  cnt++
}
return cnt
```

TC = O(1)

SC = O(1)

```
cnt = 0
while (N > 0) {
    if ( N & 1 == 1)   cnt++ // cnt += (N & 1)
    N = (N >> 1)   // N → N/2
}
return cnt
```

TC = O(log (N))

SC = O(1)

$N \to \frac{N}{2} \to \frac{N}{2^2} \to \dots \frac{N}{2^K} = 1 \Rightarrow N = 2^K$

$\Rightarrow K = \log_2(N)$

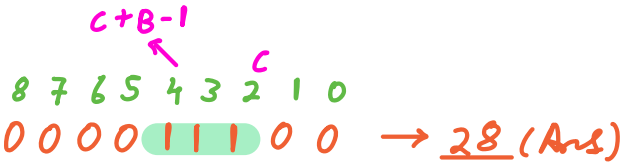
Q → A group of people is working on a project that includes encoding binary numbers. They need to create a binary number with a specific pattern of the project. The pattern requires A 0's followed by B 1's followed by C 0's. A, B & C → inputs

Find the decimal representation of the pattern.

A = 4

B = 3

C = 2



→ 28 (Ans)

A, B, C <= 20

⇒ long

```
ans = 0
for i → C to (C+B-1) {
    ans |= (1<<i)
}
return ans
```

TC = O(B)

SC = O(1)
