Binary number → 1 0 1 0

8 4 2 1 → 8 + 2 = 10

a	b	AND a & b	OR a | b	XOR a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

1 → true/set

0 → false/urset

Basic AND Properties

1) check odd & even →

9 → 1001

13 → 1101

15 → 1111

last bit is 1

8 → 1000

10 → 1010

14 → 1110

last bit is 0

> check last bit
>
> A & 1 → 1 ⇒ odd
>
> → 0 ⇒ even

9 & 1 → 1001 & 0001 → 0001 → 1

10 & 1 → 1010 & 0001 → 0000 → 0

A & 0 = 0

A | 0 = A

A ^ 0 = A

A & A = A

A | A = A

A ^ A = 0

any number

Commutative Property

$A \& B = B \& A$ $\quad A | B = B | A$ $\quad A \wedge B = B \wedge A$

Associative Property

$(A \& B) \& C = A \& (B \& C)$

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

$(A \wedge B) \wedge C = A \wedge (B \wedge C)$

$A = 5 \rightarrow 1\,0\,1$ $\quad (A \& B) = 0$ $\quad 0 | C = \underline{6}$

$B = 2 \rightarrow 0\,1\,0$ $\quad (B | C) = 6$ $\quad A \& 6 = \underline{4}$

$C = 6 \rightarrow 1\,1\,0$

Q → $a \wedge b \wedge a \wedge d \wedge b$

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

$= 0 \wedge 0 \wedge d = \underline{d}$

Q → $1 \wedge 3 \wedge 5 \wedge 3 \wedge 2 \wedge 1 \wedge 5$

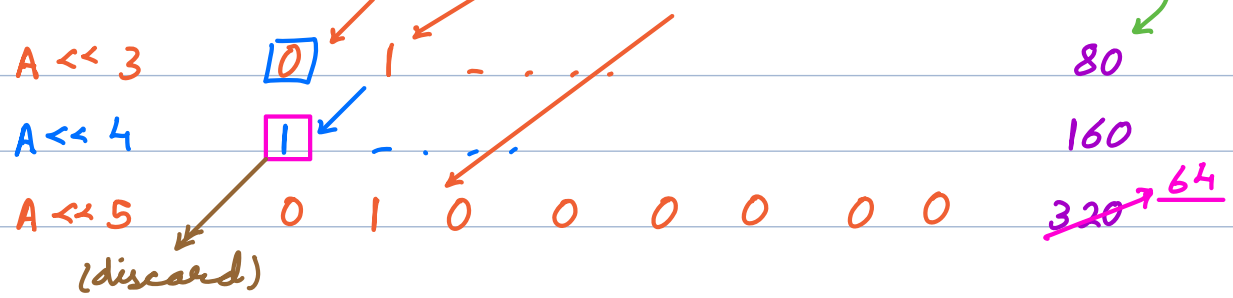
$= (1 \wedge 1) \wedge (3 \wedge 3) \wedge (5 \wedge 5) \wedge 2$

$= 0 \wedge 0 \wedge 0 \wedge 2 = \underline{2}$

Left Shift (<<)

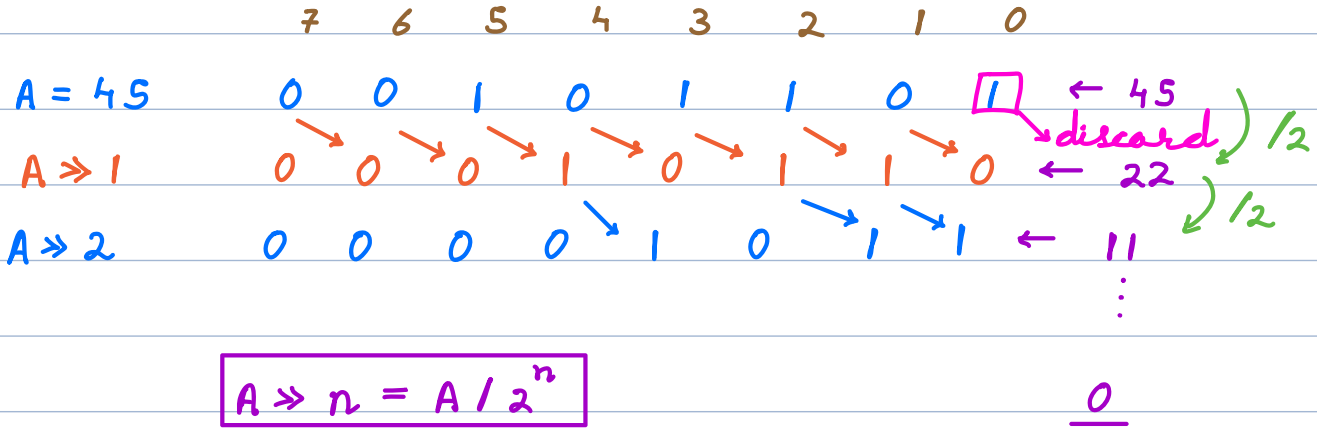
	7	6	5	4	3	2	1	0	
A = 10	0	0	0	0	1	0	1	0	← 10
A << 1	0	0	0	0	1	0	1	0	0 ← 20
A << 2	0	0	0	1	0	1	0	0	0 ← 40

*2 *2



$A << n = A * 2^n$

Right shift (≫)



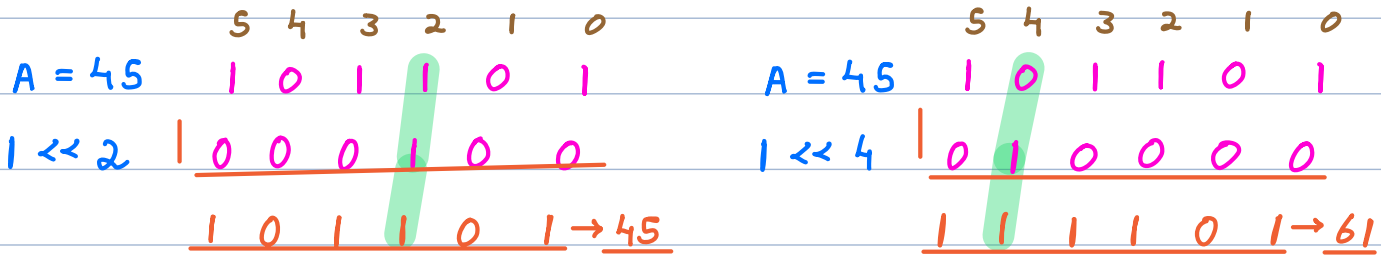
$A \gg n = A / 2^n$

$1 << 3 = 8$

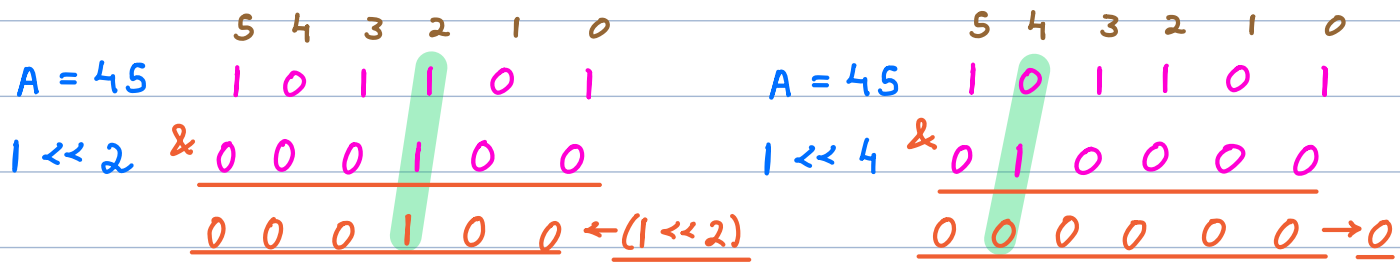
100

$1 << n = 2^n$

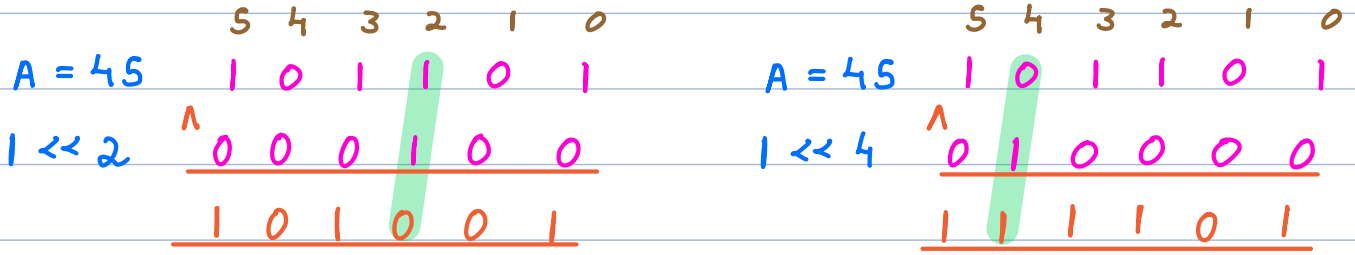
A	(1 << i)



A | (1 << i) → set i^{th} bit (no change if already set)



A & (1 << i) → (1 << i) ⇒ i^{th} bit in A is set
↳ 0 ⇒ i^{th} bit in A is unset



A ^ (1 << i) → toggle i^{th} bit $\begin{pmatrix} 1 \to 0 \\ 0 \to 1 \end{pmatrix}$

Q → check if i^{th} bit in A is set.

return (A & (1<<i)) > 0 TC = O(1)

SC = O(1)

Q → count the number of set bit in a integer A.

32 bits

A = 45 → 0... 1 0 1 1 0 1 Ans = 4

```
cnt = 0
for i → 0 to 31 {
    if ((A & (1<<i)) > 0)
        cnt++
}
return cnt
```

TC = O(32) = O(1) SC = O(1)

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

$A \to A/2 \to A/2^2 \ldots \frac{A}{2^k} = 1$

$\Rightarrow A = 2^k$

$\to TC = O(\log_2(A)) \Rightarrow k = \log_2(A)$

Q → Given an integer N, unset i^{th} bit.

N = 10

i = 3

3 2 1 0

1 0 1 0 → 0 0 1 0 = 2

N = 10 Ans = 10

i = 2

```
if ((N & (1 << i)) > 0)
        N = N ^ (1 << i)
return N
```

Q → A group of CS are working on encoding binary numbers. They need to create a binary number with a specific pattern for their project. The pattern requires A 0's followed by B 1's followed by C 0's. To simplify the process, create a function that takes A, B & C in input & return decimal value of that. (A + B + C <= 32)

A = 4 B = 3 C = 2

8 7 6 5 4 3 2 1 0

0 0 0 0 1 1 1 0 0

$2^4 + 2^3 + 2^2 = 16 + 8 + 4 = 28$

A = 2 ignore

B = 1 C = 1

3 2 1 0

0 0 1 0

$2^1 = 2$

B = 2 C = 4

5 4 3 2 1 0

1 1 0 0 0 0 → 48

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

TC = O(B) SC = O(1)

A = 3 B = 2 C = 4

(B+C+A-1) (B+C) (C-1) 0

8 7 6 5 4 3 2 1 0

1 1 1 0 0 1 1 1 1
