Bitwise Operators (&, |, ^, ~, <<, >>)

bit → 0/1

a	b	a & b	$a \mid b$	a ^ b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Same Same puppy shame.

int A = 10
int B = 4
int C = (A&B)
0 ←

```
  1010
& 0100
  ----
  0000
```

Basic properties

And properties

1) A & 1 = 1 (if A is odd)
→ 0 (if A is even)

```
  x x x x x x (-)
& 0 0 0 0 0 0 1
  -------------
  0000000 1
          or
          0
```

2) A & 0 = 0

```
& x x x x x x  ← A
  0 0 0 0 0 0
  -----------
  0 0 0 0 0
```

3) A & A = A

```
  A→ 1010
& A→ 1010
     ----
A →  1010
```

OR properties

1) $A | 0 = A$

$$\begin{array}{rl} A \rightarrow & 1\ 0\ 1\ 0 \\ \text{OR } 0 \rightarrow & \underline{0\ 0\ 0\ 0} \\ A \rightarrow & \underline{1\ 0\ 1\ 0} \end{array}$$

2) $A | A = A$

$$\begin{array}{rl} A \rightarrow & 1\ 0\ 1\ 0 \\ \text{OR } A \rightarrow & \underline{1\ 0\ 1\ 0} \\ A \rightarrow & \underline{1\ 0\ 1\ 0} \end{array}$$

XOR properties

1) $A \wedge 0 = A$

$$\begin{array}{rl} A \rightarrow & 1\ 0\ 1\ 0 \\ \wedge\ 0 \rightarrow & \underline{0\ 0\ 0\ 0} \\ & \underline{1\ 0\ 1\ 0} \end{array}$$

2) $A \wedge A = 0$

$$\begin{array}{rl} A \rightarrow & 1\ 0\ 1\ 0 \\ \wedge\ A \rightarrow & \underline{1\ 0\ 1\ 0} \\ & \underline{0\ 0\ 0\ 0} \end{array}$$

Commutative

// order of operands doesn't matter

$A \& B = B \& A$

$A | B = B | A$

$A \wedge B = B \wedge A$

Associative Properties

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

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

$(A \wedge B) \wedge C = A \wedge \underline{(B \wedge C)}$

$(A \& B) \mid C \neq A \& (B \mid C)$

Evaluate the expression: a ^ b ^ a ^ d ^ b

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

$\downarrow$

$(\underbrace{a \wedge a}_{0} \wedge b \wedge b \wedge d)$

$0 \quad b \quad 0 \quad \underline{d}$

Ans $\Rightarrow$ $\underline{d}$

Evaluate the expression: 1 ^ 3 ^ 5 ^ 3 ^ 2 ^ 1 ^ 5

1 ^ 1 ^ 3 ^ 3 ^ 5 ^ 5 ^ 2

0 → 3 → 0 → 5 → 0 → 2

Ans 2

Left shift operator

(<<)

(8 bit)

a = 10 0 0 0 0 1 0 1 0

		7	6	5	4	3	2	1	0	
a<<0	=	0	0	0	0	1	0	1	0	
discard										
a<<1	=	0	0	0	1	0	1	0	0	20
discard										
a<<2	=	0	0	1	0	1	0	0	0	40
a<<4	=	1	0	1	0	0	0	0	0	160
discard										
a<<5	=	0	1	0	0	0	0	0	0	64).

not 320

↳ overflow.

$$\boxed{\begin{aligned} a << n &= a * 2^n \\ 1 << n &= 2^n \end{aligned}}$$

[illegible].

→ Provided no overflow

Right shift operator >>.

7 6 5 4 3 2 1 0

a = 20 => 0 0 0 1 0 1 0 0 → discarded

10 a >> 1 => 0 0 0 0 1 0 1 0

5 a >> 2 => 0 0 0 0 0 1 0 1 → discarded

2 a >> 3 => 0 0 0 0 0 0 1 0

1 a >> 4 => 0 0 0 0 0 0 0 1

0 a >> 5 => 0 0 0 0 0 0 0 0

$$\boxed{\begin{array}{l} a >> n = \dfrac{a}{2^n} \\[2ex] 1 >> n = \dfrac{1}{2^n} \end{array}}$$

Ques) $1 << 3$

↳ $2^3 \Rightarrow 8$

IST

Break 7:56Am - 8:06Am

Power of Left Shift operators

↳ OR (|)

Set a bit

```
      5 4 3 2 1 0
N =   1 0 1 1 0 1          ith bit ( 1st bit)
```

(1<<1)

```
1 →          0 0 0 0 0 0 1
1 << 1 →     0 0 0 0 0 1 0
```

N = N | (1<<1)

```
          5 4 3 2 1 0
N =       1 0 1 1 0 1
1 << 2 →  0 0 0 0 1 0
          -----------
          1 0 1 1 1 1
```

```
         7 6 5 4 3 2 1 0
    N =  1 0 1 0 1 1 1 1
OR 1<<4 = 0 0 0 1 0 0 0 0
         ---------------
         1 0 1 1 1 1 1 1

1 →     0 0 0 0 0 0 0 1
1<<4 →  0 0 0 1 0 0 0 0
```

```
          7 6 5 4 3 2 1 0
N =       1 0 1 1 1 1 1 1
1<<4 =    0 0 0 1 0 0 0 0
          ---------------
          1 0 1 1 1 1 1 1
```

N = N | (1<<i)   (sets my ith bit in N)

toggle the bit

		7	6	5	4	3	2	1	0
	N =	1	0	1	0	1	1	1	1
xor	1<<4 =	0	0	0	1	0	0	0	0
		1	0	1	1	1	1	1	1

7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1
0	0	0	1	0	0	0	0
1	0	1	0	1	1	1	1

N = N ^ (1<<i) (toggles ith bit)

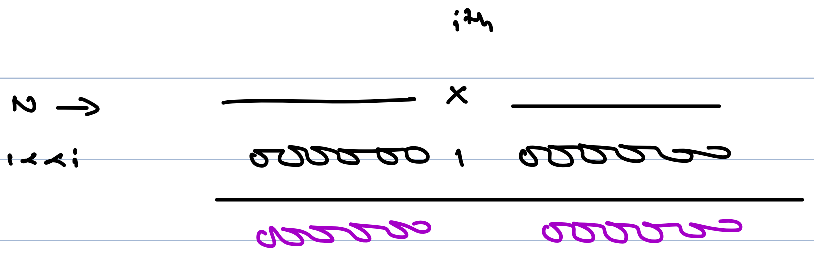
N = N & (1<<i) (to check ith bit is set or not)

		7	6	5	4	3	2	1	0	
	N =	1	0	1	0	1	1	1	1	
and	1<<4 =	0	0	0	1	0	0	0	0	
		0	0	0	0	0	0	0	0	=> 0

		7	6	5	4	3	2	1	0	
	N =	1	0	1	1	1	1	1	1	
and	1<<4 =	0	0	0	1	0	0	0	0	
		0	0	0	1	0	0	0	0	=> non zero.

if n & (1<<i) → 0 (ith bit was 0 in N)

→ ≠ 0 (ith bit was not 0 in N)



Ques) check ith bit is set or not

```
func check (n, i) {
    if ((n & (1<<i)) == 0) {
        return false;
    }
    else {
        return True;
    }
}
```

T.C → O(1)

S.C → O(1).

Ques) Given N, count no. of set bits in N.

N = 12, (1 1 0 0) Ans → 2

N = 8, 1000 , Ans → 1 .

Approach 1 :-

```
int ans = 0;
for (i=0; i < 32; i++) {
    if (check(N, i)) {
        ans++;
    }
}
return ans;
```

T.C → O(1).

Approach 2

N = 10, 1010

```
N   = ________ 0/1
& 1 = 0000000 1
      ---------
      000000 0/1
           ↓
      checking 0th
             bit.
```

```
           7 6 5 4 3 2 1 0
N =        0 0 0 0 1 0 1 0   & 1  =>  0
           7 6 5 4 3 2 1 0
N>>1 =>    0 0 0 0 0 1 0 1   & 1  =>  1
           7 6 5 4 3 2 1 0
N>>2 =>    0 0 0 0 0 0 1 0   & 1  =>  0
           7 6 5 4 3 2 1 0
N>>3 =>    0 0 0 0 0 0 0 1   & 1  =>  1
           7 6 5 4 3 2 1 0
N>>4 =>    0 0 0 0 0 0 0 0   }
```

		7	6	5	4	3	2	1	0		
17	N =	0	0	0	1	0	0	0	1	& 1	=> 1
8	N>>1 =>	0	0	0	0	1	0	0	0	& 1	=> 0
4	N>>2 =>	0	0	0	0	0	1	0	0	& 1	=> 0
2	N>>3 =>	0	0	0	0	0	0	1	0	& 1	=> 0
1	N>>4 =>	0	0	0	0	0	0	0	1	& 1	=> 1
0	N>>5	0	0	0	0	0	0	0	0		

```
                       ↪ 17
func  countbit (n) {
    int  ans = 0;
    while (n>0) {
        if ( (n&1) = = 1) {
              ans++;
        }
        n = N>>1;
    }
    return ans;
}
```

T.C → O(log N)

	7	6	5	4	3	2	1	0	
N=>	1	1	1	1	1	1	1	1	=> 255

Ques) unset ith bit

$$N = 12, \Rightarrow \begin{array}{cccc} 3 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{array} \qquad i = 2$$

↓

$$8 \leftarrow \begin{array}{cccc} 3 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{array}$$

Step 1) . check ith bit

↓

if it's already unset
do nothing

if it's set .

toggle bit .

N = N ^ (1 << i);

$$\begin{array}{lccc} N = & \_\_\_\_\_\_ & \overset{0/1}{x} & \_\_\_\_\_\_ \\ \wedge \quad 1 = & \_\_\_\_\_\_ & 1 & \_\_\_\_\_\_ \\ \hline & \_\_\_\_\_\_ & 1/0 & \_\_\_\_\_\_ \end{array}$$

Ques) $B = 3, C = 2$ → form a no' in which B no. of set bits are there, followed by C no' of unset bits.

28 <— 11100

Ans.

Let's just say $B=3, C=2$

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

$B=2, C=5$

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

→ cth idx and go till $B+C-1$

```
int ans=0;
for (i= C; i <= B+C-1; i++) {
        set ith bit
}
```

Soln :- $((1<<B)-1)<<C$ ← O(1)

next doubt session

-ve no's

→ 2's Complement

n → 0 0 0 1 0 0 0

1's comp → 1 1 1 0 1 1 1

2's comp → + 1

1 1 1 1 0 0 0

N = -10

10 ⇒ 0 0 0 0 1 0 1 0

1's ⇒ 1 1 1 1 0 1 0 1

2's + 1

1 1 1 1 0 1 1 0

~ → 1 to 2^{30}

31 30 0

No. of doubt One

↳ help on the whatsapp group

next presentation

N = 11

1st appr 1101

1 2 3 4 5 6 7 8 9 10 11

bitwise operation