Bit Manipulation

binary

int x = 2;

nibble

2^4

16

1	000	-8
1	001	-7
1	010	-6
1	011	-5
1	100	-4
1	101	-3
1	110	-2
1	111	-1
0	000	0
0	001	1
0	010	2
0	011	3
0	100	4
0	101	5
0	110	6
0	111	7

-5

1011

2's compliment

① ⇒ Compliment bit

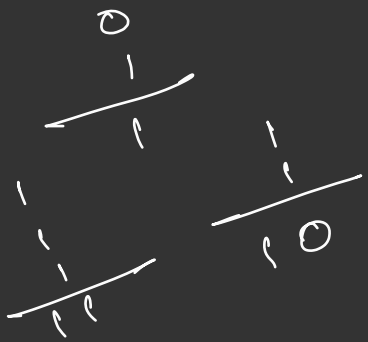
② add 1

```
 0101
-----
 1010
    1
-----
 1011
```

① 0 - is X

② -ve zero

③ MSB - sign



Decimal – binary

base 10

$487 \Rightarrow 4 \times 10^2 + 8 \times 10^1 + 7 \times 10^0$

$101 \Rightarrow 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

+ve

1. Convert
2. Fit to bits

$x = -100$

-ve

1. Convert without sign
2. Fit to bits
3. 2's compliment

---

(2) Binary – Decimal

+ve

1. MSB = 0
2. Convert

-ve

1. MSB = 1
2. 2's compliment
3. Convert
4. -ve sign

Bitwise Operators

① & (AND)

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

a & 1 ⇒ a

a & 0 ⇒ 0

② | (OR)

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

a | 0 ⇒ a

a | 1 ⇒ 1

③ ^ (XOR)

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

a ^ 1 ⇒ ~a

a ^ 0 ⇒ a

a ^ a ⇒ 0

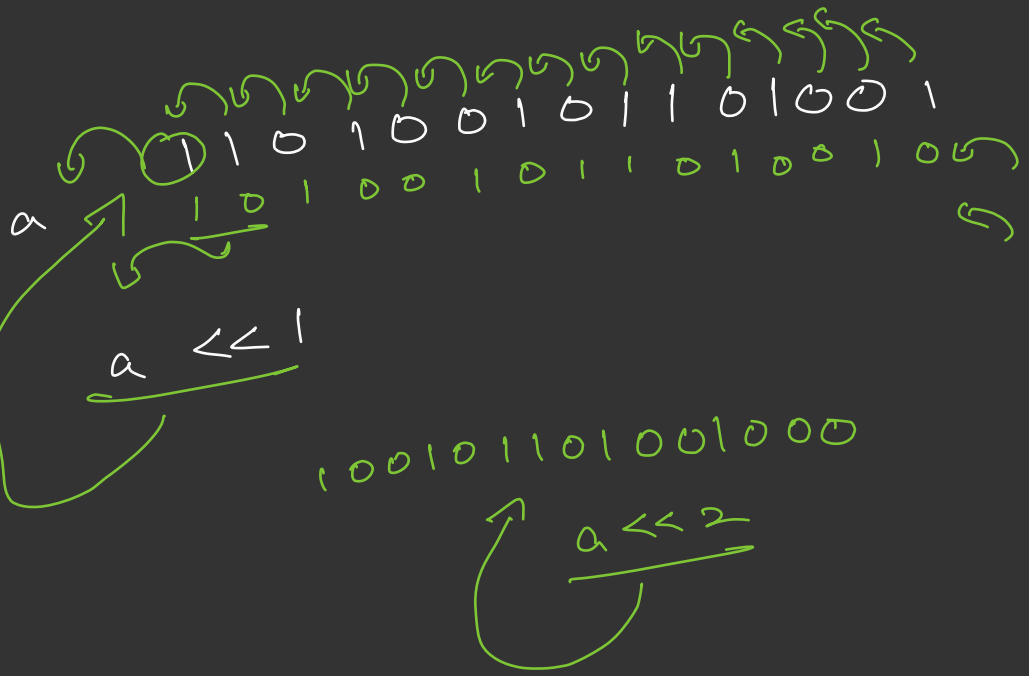
④ ~

~a

~1 ⇒ 0

~0 ⇒ 1

(5) << (Left shift)



0 0 0 0 1 ⇒ 1

0 0 0 1 0 ⇒ 2

1 0 0 ⇒ 4

1 0 0 0 ⇒ 8

1000+0001

1001 = 9

>> (Right Shift)

a = 0 1 0 1 1

0 0 0 1 0

a >> 2

b = 1 1 0 1 1 0 0

b >> 3

1 1 1 1 1 0 1

>>>

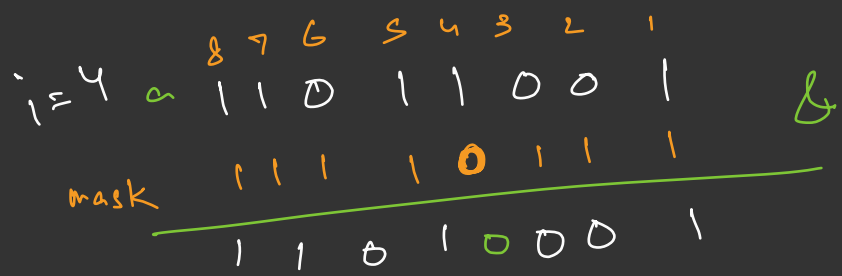
Only adds 0 on Left

b = 1 1 0 1 1 0 0

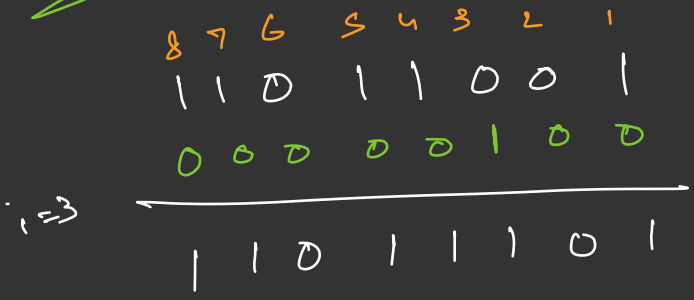
b >>> 3

0 0 0 1 1 0 1

Ques turn i^{th} bit off (convert bit to 0) a & mask



Ques turn ith bit on (convert bit to 1)



OR (1)

Ques toogle the bit

0 0 0 0 0 0 1 << 2

i = 3

1 << (i-1)th

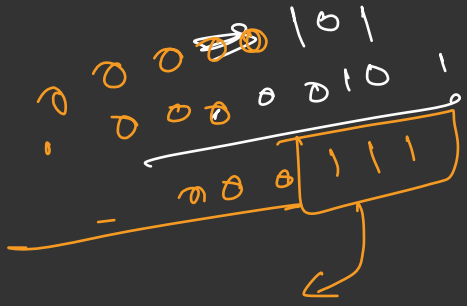
2	5
2	2 1
2	1 0
	0 1

int n = 5

i = 2

1 << 1

0 0 0 0 1 0 ⇒ mask



1 1 1 → 2^0, 2^1, 2^2

4 + 2 + 1 = 7

0 0 0 0 0 0 0 1

0 0 0 1 0 0 0

1 << i-1

ith

i = 6 Turn off bit

12 11 10 9 8 7 6 5 4 3 2 1

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

1 1 1 1 1 1 0 1 1 1 1

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

mask 0 0 0 0 0 0 1 0 0 0 0

~mask

a 0 0 0 0 0 0 0 0 1

b = 1 << 5

b = 0 0 0 0 0 1 0 0 0 0

~b ⇒ 1 1 1 1 0 1 1 1 1

Ques Check ith bit

9 8 7 6 5 4 3 2 1

1 0 1 1 0 1 0 0 1

i = 4 ⇒ true

i = 5 ⇒ false

1 0 0 1 0 1

& 0 0 0 1 0 0

0 0 0 1 0 0

1 0 0 0 1

& 0 0 1 0 0

0 0 0 0 0

Ques Odd Even

n = 6 ⇒ LSB == 0 (even)
== 1 (odd)

Note

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

n ^ n ⇒ 0

n ^ 0

Ques

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

1 → n → L+1

1 no is missing

Find the missing no

Arithmatic

sum = 2+6+1+3+5+7+4

⇒ 28

$\frac{n \times (n+1)}{2} \Rightarrow \frac{8 \times 9}{2} \Rightarrow 36$

⇒ 36 − 28 = 8

res = 0 ^ 2 ^ 6 ^ 1 ^ 3 ^ 5 ^ 7 ^ 4 ^ 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 ^ 8

i = 1 ; i <= 8

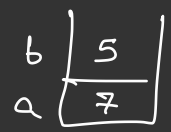
Ques Swap 2 numbers

a, b

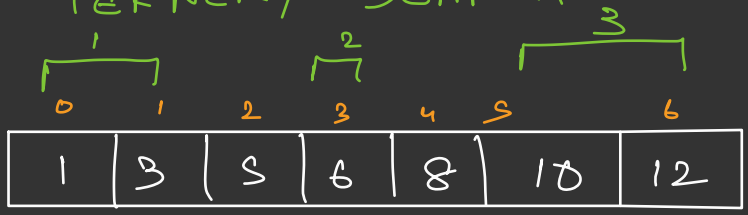
```
int t = a;
a = b;
b = t;
```

```
int a = 5;
int b = 7
```

```
a = a^b;
b = a^b;
a = a^b;
```



TERNERY SEARCH



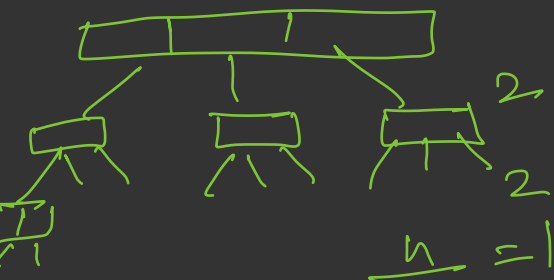
$\frac{r-l}{3}$ l m_1 m_2 r

$m_1 = l + \left(\frac{r-l}{3}\right)$

$m_2 = r - \left(\frac{r-l}{3}\right)$

```
while(left <= right) {
    int m1 = left + (right - left)/3;
    int m2 = right - (right - left)/3;

    if(arr[m1] == val) return m1;
    else if(arr[m2] == val) return m2;
    else if(val < arr[m1]) {
        right = m1 - 1;
    } else if(val > arr[m2]) {
        left = m2 + 1;
    }else {
        left = m1 + 1;
        right = m2 - 1;
    }
}
return -1;
```



$\frac{n}{3^k} = 1$

$\Rightarrow k = \log_3 n \Rightarrow 2\log_3 n$

Binary

$1 \times \log_2 n$

$1 \times 3.1 \ldots$

$\Rightarrow 3.1$

Ternary

$2 \times \log_3 n$

2×2

$\Rightarrow 4$

9