quiz

$(45)_{10} = (101101)_2$

2	45	
2	22	1
2	11	0
2	5	1
2	2	1
2	1	0
	0	1

Bitwise operators => &, |, ^, ~ → invert

& → bitwise AND

| → bitwise OR

^ → XOR

<< , >>

sopln (13|10); 15

```
  1 1 0 1
OR 1 0 1 0
-----------
  1 1 1 1   → 15
```

soPln (13 & 10); 8

```
  1 1 0 1
AND 1 0 1 0
-----------
  1 0 0 0   ⟶ 8
```

soPln (13 ^ 10); 7

```
  1 1 0 1
XOR 1 0 1 0
-----------
  0 1 1 1   ⟶ 7
```

soPln (15 << 3); 120

a << N = (2^N) a

1 << N = 2^N

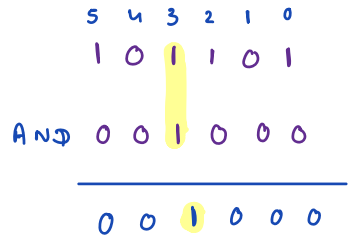
soPln (35 >> 3); 4

a >> N = $a / 2^N$

Some applications of left shift

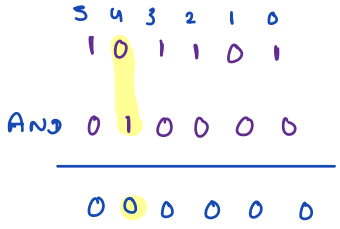
n = 45

1 << 3



n = 45

1 << 4



ON / set → bit is 1

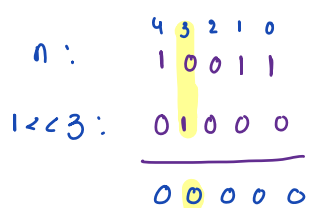
OFF / unset → bit is 0

```
boolean checkbit ( int n, int i )
{
    // check whether ith indexed bit in n is on or not

    if ( ( n & ( 1 << i ) ) == 0 ) {

        return false;

    }

    else {

        return true;

    }

}
```

TC: O(1)

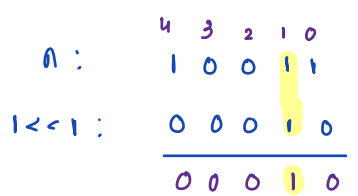
n = 19

i = 3



n = 19

i = 1



n = 45

	5	4	3	2	1	0
	1	0	1	1	0	1
1 << 3 OR	0	0	1	0	0	0
	1	0	1	1	0	1

n = 45

	5	4	3	2	1	0
	1	0	1	1	0	1
1 << 4 OR	0	1	0	0	0	0
	1	1	1	1	0	1

n | (1 << i) ⟶ set the i^{th} indexed bit in n.

n = 9 , i = 2

```
int setbit (int n, int i) {
    // set ith bit in n
    n = n | (1 << i);
    return n;
}
```

	3	2	1	0
9 :	1	0	0	1
1 << 2 :	0	1	0	0
	1	1	0	1

n = 45

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

n = 45

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

n ^ (1 << i) = toggle / flip ith bit

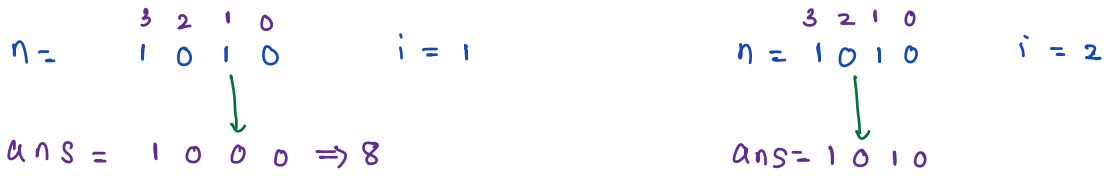
```
int flipbit (int n, int i) {
    // in n flip bit at ith index
    n = n ^ (1 << i);
    return n;
}
```

n = 10

i = 2

	3	2	1	0
10 :	1	0	1	0
1 << 2 :	0	1	0	0
	1	1	1	0

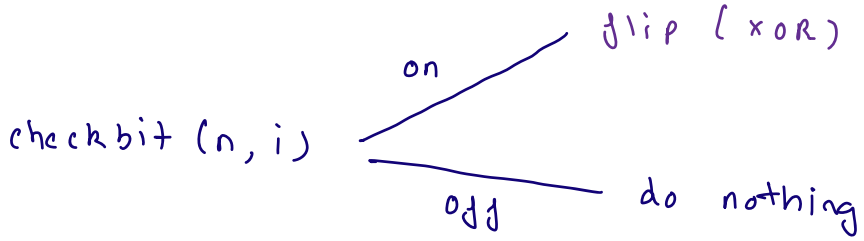
Q.1 Given an integer n and i, unset the ith indexed bit in n number.



n & (1<<i) → check bit at i^{th} index in n

n | (1<<i) → set bit at i^{th} index in n

n ^ (1<<i) → flip bit at i^{th} index in n



Q.2 Given a no. n, count total no. of **set bits**.

bits which are 1.

n	ans
10 (1010)	2
0 (0000)	0
17 (10001)	2
13 (1101)	3

TC: O(1)

int : 4 bytes (1 byte = 8 bits)

int : 32 bit

idea: go on every bit from i=0 to i=31,

then checkbit(n,i) == true

↳ cnt++;

```
int countSetBits(int n) {
    // return total no. of on / set bit in n
    int count = 0;
    for(int i=0; i<=31; i++) {
        if(checkbit(n,i) == true) {
            count++;
        }
    }
    return count;
}
```

itr : 32

TC: O(1)

n = 45

1 0 1 1 0 1

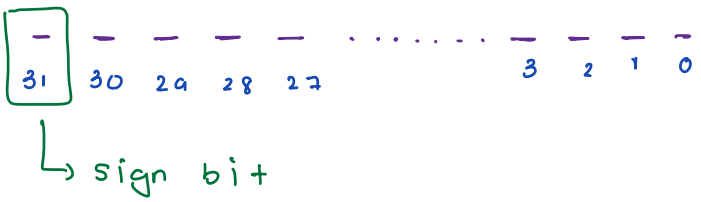
5 4 3 2 1 0

i	cnt
0	1
1	1
2	2
3	3
4	3
5	4
:	:
31	4

Dealing with negative numbers

int : 32 bits

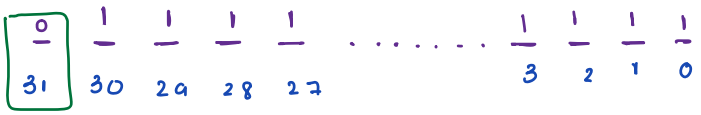
long : 64 bits



Sign bit is 1 (for -ve no.)

Sign bit is 0 (for +ve no.)

max +ve value



$$2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^{30}$$

GP

$a = 1$

$r = 2$

$t = 31$

$$S_t = \frac{a(r^t - 1)}{r - 1}$$

$$= \frac{1(2^{31} - 1)}{1} = 2^{31} - 1$$

int (roughly) $\Rightarrow$ -10^9 to 10^9

int (accurate) $\Rightarrow$ -2^{31} to $2^{31} - 1$

How to convert -ve int no. to binary:

i) convert positive equivalent of the no. to binary.

ii) flip all bits

iii) add a 1.

Note: Let's understand the concept in 8 bits form rather than 32 bits (logic is absolutely same)

n = -13

0 0 0 0 1 1 0 1 → 13

1 1 1 1 0 0 1 0

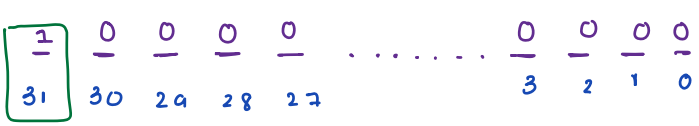
\+ 1

Sign bit 1 1 1 1 0 0 1 1

-128 + 64+32+16 + 2+1

-128 + 64 + 32 + 16 + 2 + 1 = -13

min int value



n = -45

00101101 (45 in 8 bit format)

11010010

\+ 1

11010011

1 1 0 1 0 0 1 1

↓ ↓ ↓ ↓ ↓

$-128 + 64 + 16 + 2 + 1 = -128 + 83$

$= -45$

45: 00000000 00000000 00000000 00101101

-45: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1

↓

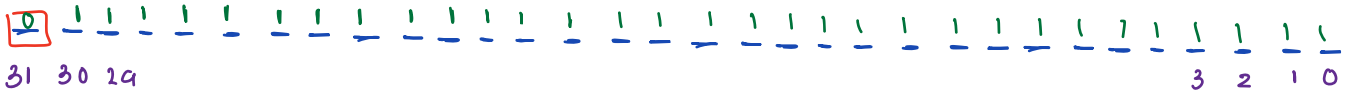
-2^{31}

(32 bit format)

Doubts

=

int : 32 bit



max int val

$$2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^{30}$$

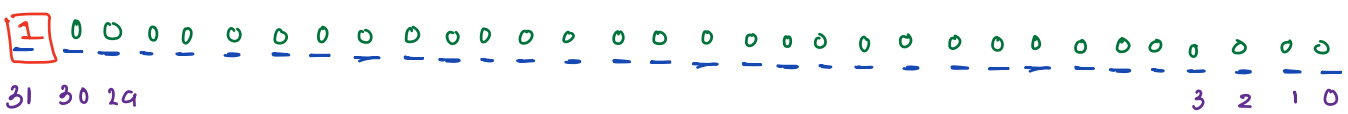
$a = 1$

$r = 2$

$t = 31$

$$\frac{1(2^{31} - 1)}{2 - 1} = 2^{31} - 1$$

min int val



$$-2^{31}$$

$$-2^{31} \quad \text{to} \quad 2^{31} - 1$$