Bit magic:

Bitwise AND

x : 0 ... 0011

y : 0 ... 0110

(x & y) : 0 ... 0010

```
public class test {
public static void main(String[] args)
{ int x= 3, y= 6;
  System.out.println(x&y);
}
}
```

```
  011
& 110
-----
  010 = 2
```

OR

```
0 0 — 1
0 1 — 1
1 0 — 1
1 1 — 0
```

OR 101 = 5

Bitwise NOT

x : 000...01 = 1

~x = 111...10 ($2^{32}-1-1$) ($2^{32}-2$)

-2

int x = 1

print (~x);

→ ~1 = -2

In Java, negative numbers are stored in 2's complement rep

Rep of $-x = 2^{32} - x$

Reason

Range of int: -2^{31} to $2^{32}-1$

= 1111...32 times

By 2^3-1 = 111

2^3-1-1 = 110

By int x = 5

print (~x)

($2^{32}-1-5$) = $2^{32}-6$

O/p = -6

∴ Complement of

5*24

Date / / Page 132

$-x = -(x+1)$

Left shift ≪ | Right shift ≫

formula

$x \ll 1$ | $x \gg 1$

$x \times 2^1$ | $\frac{x}{2^1}$

$x \ll 2$ | $x \gg 2 \Rightarrow x/2^2$

$x \ll 1 = 2$

1111 . . . 1

111 10

$\Rightarrow 2^{32} - 1 - 1 = -2$

$\therefore x \ll 1 = -2$

$2^{32} - 1$ {$-x = 2^{32} - x$}

$\Rightarrow$ 111 . . 1 (32 times)

116 . . . 10

$2^{32} - 1 - 1 = -2$

OR

$-1 \ll 1 \Rightarrow -1 \times 2^1 = -2$

Right shift

$x = 33$ $x \gg 1 \Rightarrow \frac{33}{2^1} = 16$

x: +ve (Change leading bits → 0)

$x = 0 \ldots 100001$

$x \gg 1 = 0 \ldots 10000 = 16$

$x \gg 2$ $0 \ldots 10000\,1$

$000 \ldots 1000 = 8$

Change leading bits → 1

x: -ve

$x = -2$

$x \gg 1$ $-2/2^1 = -1$

x: 1111 . . . 10

$11 \ldots 1 = 2^{32} - 1$

$= -1$

$-2 \gg 2$ $-2/2^2 = 0$

x: 1111 . . . 10

1111 . . . 11 $= -1$

So $(-x \gg y) = \{-1 : y >= 1\}$

31 bits 1 + sign.

$2^{32}-2$

Unsigned Right Shift: >>>

$x = -2$

$x >>> 1$

x : 111 ... 10

$x >>> 1$: 0 111 ... 1

$= 2^{31}-1 = 2147483647$

one Zero at leading Bit → +ve number.

$x = -2$ = 111 ... 10

$x >>> 2$ = 00 111 ... 1

$= 2^{30}-1 = 1073741823$

111 = 7

$2^3 - 1$

111 ... 30 times

$2^{30}-1$

Check k^{th} bit is set or not from right side

If $n=5$ $k=1$ → 000 ... 010 (k=1 position)

Op yes.

$n=8$ $k=2$ → 00 ... 1000

No

$n=0$, $k=3$ → 00000 ... 000

No

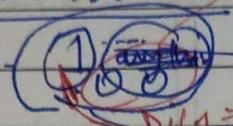
$k \leq$ No. of bits in Binary representation

Left shift 1 with (k-1)s do & with n ⇒ if non zero ⇒ Yes
else ⇒ No.

n = 5, k = 3
Yes.

1 = 000...1
1<<(k-1)
00...100

5 → 00...0101
1<<k-1 → 00...0100



kth bit of n = 5 set bit & ... result in non-zero ⇒ yes.
0 ⇒ result non-zero ⇒ No.

Program

```
if (n & (1<<(k-1)) != 0)
    print("Yes");
else
    print("No")
```

M1 - Left shift

Idea: from a new no, whose kth bit from right is 1, else 0
& Find(n & this obt no) == 1 ret Yes.

0 or 0 1 0

M2 - Right shift

```
if ((n >> (k-1)) & 1) == 1)
    print("Yes")
else
    print("No")
```

Move kth bit of number to last posn & do Bitwise & with 1 ⇒ if 1 ⇒ Yes
0 ⇒ No.

Ex n = 13 k = 3

00...1101

n >> 2 → 0000...11

000001

if == 1 · Yes.

Count set bits.

n = 5 (101)
o/p = 2

n = 7 (111)
o/p = 3

n = 13 (1101)
o/p 3

n>>1 or n=n/2

Pan each bit & perform &1 → if 1 cnt++ else

My

```
{ int n
while(n>0)
{ if(n&1
  count = count + (n&1);
  n>>1
return cnt;
}
}
```

M1 Xlaive : Check if last bit is 1, count it & remove the last bit..

TC: O(Total bits in n)

```
{ int res = 0
while (n>0)
{ if((n&1)==1)        if(n%2==0)        res+= (res&1);
    res++;              res++;
n=n>>1                n=n/2             n=n>>1
}
```

n=5

00 ... 0101 → 5
(a1, x)

00 ... 010* → 2
(30)

0 ... 01 → 1
(31 bit)

00 ... 0 → 0
(32 bits.)

M2 Brian Kernigan's Algorithm.

TC: O(set bit count)

n: 40

Initial: 00 .. 0101000
1st 00. 0100000
2nd 00 .. 0000000

Approach ② last set bit turned off

Subtract by 1 =>

① All 0 after last set bit → 1

&

```
1 0 1 1 0 0 0 0     n = 40
1 0 0 1 1 1 1 1     n-1 = 39
& ---------------
1 0 0 0 0 0 0 0     n = 32
```

```
1 0 0 0 0 0 0     n = 32
0 1 1 1 1 1 1     n-1 = 31
& -------------
0 0 0 0 0 0 0
```

```
int countbits(int n)
{   int res = 0
    while (n > 0)
    {   n = (n & (n-1));
        res++;
    }
    return res;
}
```

Lookup Table Method

for 32 bit number. tab[i] => represent. count of set in number i.

m3 O(1)

int table [256]

0 - 255

0 - 28-1

∴ consider 8 bits as one unit

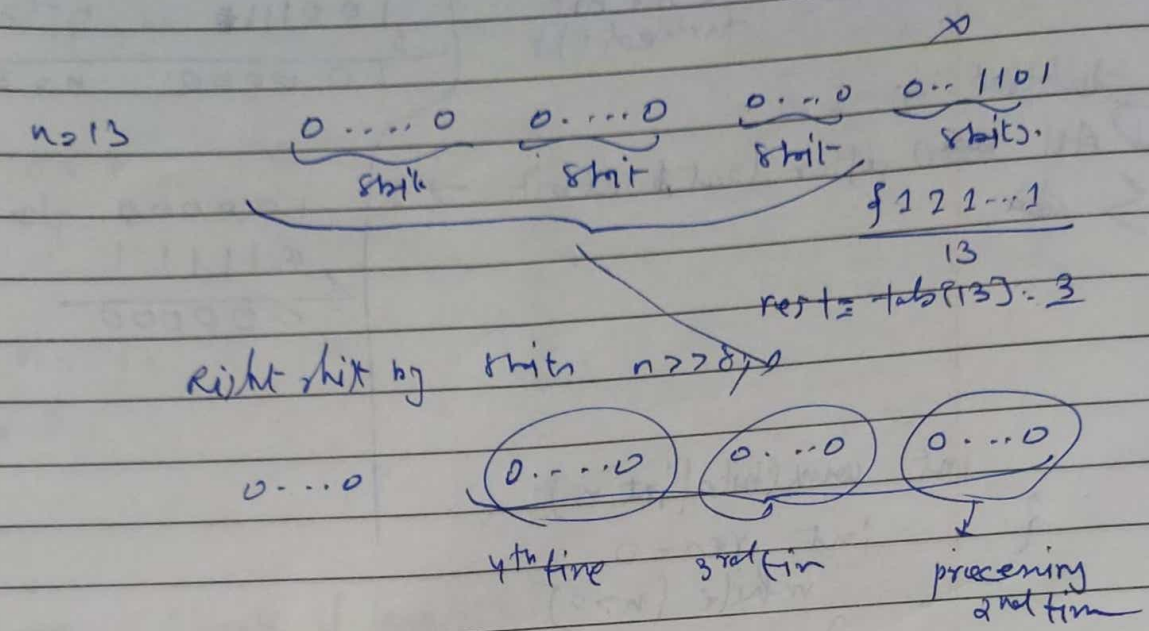
Represent of set bit in a no. i

```
void initialize()
{   table[0] = 0;
    for (int i = 01; i < 256; i++)
        table[i] = ((i & 1) + table[i/2]);
}
```

Recurrence

```
int count (int n)
{   int res = table [n & 0xff];
    n = n >> 8;
    res += res + table[n & 0xff];
    n >> 8;
    res += res + table [n & 0xff];
    n >> 8
    res += res + table [n & 0xff];
    return res;
}
```

Hash rep of 8 set bit


n=13
0....0 0....0 0...0 0..1101
8bits 8bit 8bit 8bits.
§111..1
13
rest = tab[13]: 3
Right shifting 8bits n>>8
0...0
0...0 0...0 0...0
4th time 3rd time processing 2nd time

Bitwise Part 1

AND (&)

```
int x=3, y=6
print (x&y)
```

// 2

x: 0 ... 0011
y: 0 ... 0110
0010

OR (|)

0 0 = 0
1 0 = 1
0 1 = 1
1 1 = 1

(x|y) // 7 ... 0111

XOR (^)

0 0 = 0
1 0 = 1
0 1 = 1
1 1 = 0

x^y = 5 ... 0101

Part 2

$2^{32}-1$ = all 1s = 11 11 ... 11

NOT

x = 000 ... 01
~x = 111 ... 10

-2 ⇐ $2^{32}-2$ ∈ $2^{32}-1-1$ ∈ ~x = 111 ... 10

Java, -ve nos. are stored in 2's complement

$$-x = 2^{32} - x$$

Range of int = -2^{31} to $2^{31}-1$

(2) x=5

x = 000 ... 0101
~x = 111 ... 1010

$2^{32}-1$

what did i subtracted from $2^{32}-1$ it is 0101 = 5

$\therefore \sim x = 2^{32} - 1 - 5 = 2^{31} - 6 \Rightarrow -6$

~5 = -6

$x << y = x \times 2^y$

11.000 <<1

Left shift << (Multiplication)

x = 000...0011 $x=3$

$x<<1$: 000..0110 6 3×2^1

~~$x<<2$~~ :

x : 000..0011

$x<<2$ 000..001100 12 3×2^2

$x<<4$ 0000 110000 48 (32+16) $3\times2^4 = 48$

32 16 8 4 2 1

$x=+1$ $2^{32}-1$ 11111..10 $-x$ 11111..10

$x<<1$ 11111..1..10

$x=-1$, $x<<1$: ?

$x=-1 \Rightarrow 2^{32}-1 \Rightarrow$ 11111..1

11111..10 $\Rightarrow 2^{32}-1-1$

$\Rightarrow 2^{32}-2 = -2$

Signed Right shift

→ +ve : filling leading bits with 0

→ -ve : 1

x : 00...100001 → 33

- x>>1 : 000...010000 → 16

x : 000...100001

- x>>2 : 0000...100000 → 8

- x = -2 ⇒ x>>1 = ? -1

x = -2 ⇒ $2^{32}-2$ ⇒ 111111...10

1 111111...1 ⇒ $2^{32}-1$ ⇒ -1

x>>2 = ? -1

x : 111111...110

11 11111...1 ⇒ $2^{32}-1 = -1$

-2>>y { y any value } = -1

Unsigned Right shift >>>

[+ve, -ve → both filling with 0 leading bits]

x = -2 , x>>>1 = ?

x = -2 = $2^{32}-2$ ⇒ 111...10

01111...1 = $2^{31}-1$

x>>>2 = ?

x → 111...10

001111...1 = $2^{30}-1$