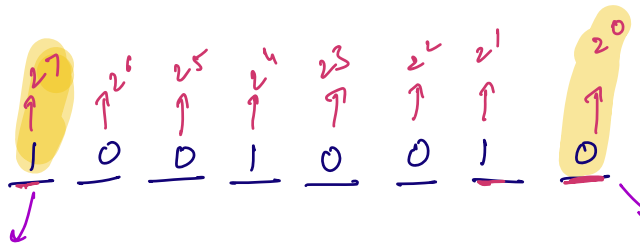
1 : Set bit

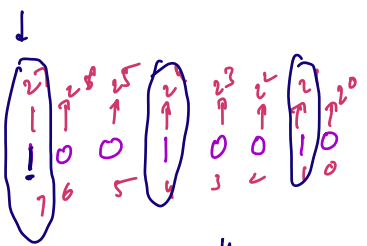
0 : Unset bit



Most Significant bit (MSB)

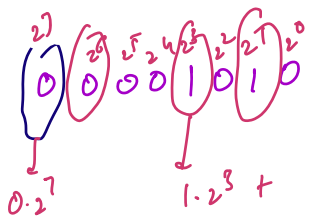
Least Significant bit (LSB)

N =



$1 \times 2^7 + 1 \times 2^4 + 1 \times 2^1$

N =



$0 \cdot 2^7$ $1 \cdot 2^5 + 1 \cdot 2^1$

a & 1 $\Rightarrow$ a & 1 = a $\Rightarrow$ a = 0, 1

a & 0 $\Rightarrow$ a & 0 = 0 $\Rightarrow$ a = 0, 1

a | 1 $\Rightarrow$ a | 1 = 1 $\Rightarrow$ a = 0, 1

a | 0 → a = 0 → 0
a = 1 → 1

⇒ a | 0 ⇒ a ✓

Question: Given 2 the integers N, i, check if i^{th} bit in N is set / unset

N = 10

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

i = 2 ⇒ false
i = 3 ⇒ True
i = 4 ⇒ false
i = 1 ⇒ True
i = 0 ⇒ false

Approach1:

N = 0 0 0 0 1 0 1 0

1 = 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 !

0 0 0 0 1 0 1 0 ← LSB
0 0 0 0 0 0 0 1

x & 1 → LSB → 1 → 1
LSB → 0 → 0

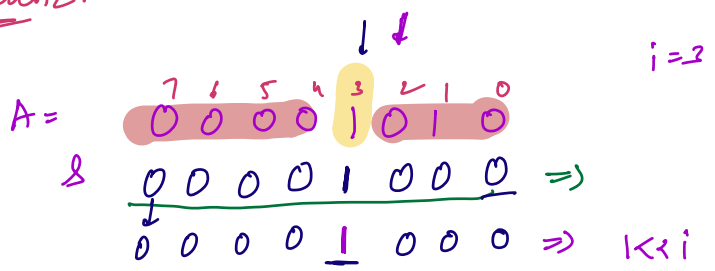
	7	6	5	4	3	2	1	0	
N =	0	0	0	0	1	0	1	0	i = 3
N >> 1	0	0	0	0	0	1	0	1	
N >> 2	0	0	0	0	0	0	1	0	
N >> 3 ✓	0	0	0	0	0	0	0	1	

```
if ( (N >> i) & 1 == 1 ){
    // "ith bit is SET"
    return true;
}
else{
    return false;
}
```

T.C: O(1)

S.C: O(1)

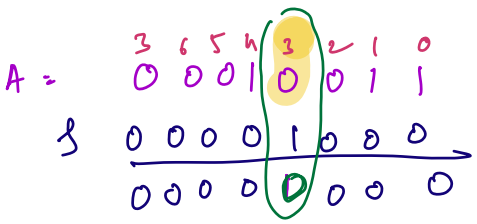
Approach2:



i=3 ⇒ 00001000

7 6 5 4 3 2 1 0

(1<<3)



i=5

i=5 ⇒ 00 100000

7 6 5 4 3 2 1 0

(1<<5)

```
if( N & (1<<i) == 0 ){
    // ith bit is unset
    return false;
}
else{
    return true;
}
```

T.C: O(1)

S.C: O(1)

Question: Given 2 the integers N, i set the i^{th} bit in N and return the answer

N = 4

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

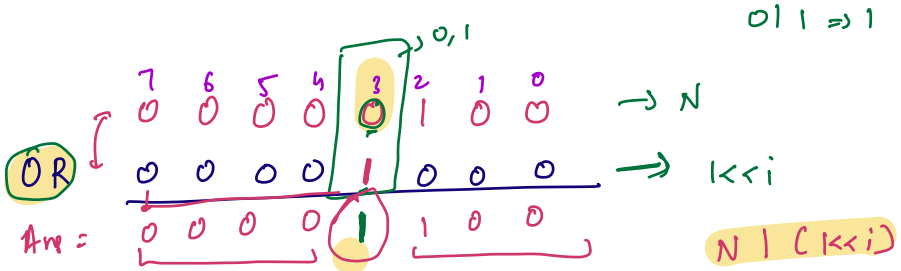
i = 3 → 0 0 0 0 1 1 0 0 ⇒ 12

i = 1 → 0 0 0 0 0 1 1 0 ⇒ 6

i = 2 → 0 0 0 0 0 1 0 0 ⇒ 4

0 | 1 ⇒ 1

N = 4
i = 3



f(0) = 1
f(0) = 1

a | 1 ⇒ 1

a | 0 =

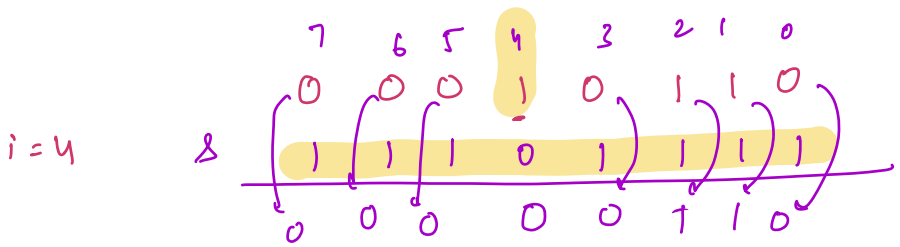
```
int setIthBit(N){
    return N | (1<<i);
}
```

Question: Unset the i^{th} bit

	7	6	5	4	3	2	1	0	
N =	0	0	0	1	0	1	1	0	
i = 4	0	0	0	0	0	1	1	0	=> 6
i = 2	0	0	0	1	0	0	1	0	=> 18
i = 1	0	0	0	1	0	1	0	0	=> 20
i = 3	0	0	0	1	0	1	1	0	=> 22

F(0) = 0

F(1) = 0



a & 1 => a

0 & 1 => 0

1 & 1 => 1

How to get

1 1 1 0 1 1 1 1

0 0 0 1 0 0 0 0 => (1<<i)

~(1<<i) => 1 1 1 0 1 1 1 1

```
int unsetIthBit(N){

    return N & (~(1<<i))

}
```

T.C: O(1)

S.C: O(1)

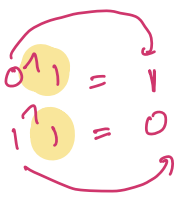
HW: Given N and i, toggle the i^{th} bit

	7	6	5	4	3	2	1	0
N =	0	0	1	0	1	1	0	1
i = 2	0	0	1	0	1	0	0	1
i = 1	0	0	1	0	1	1	1	1

F(0) = 1

F(1) = 0

$x \wedge 1$



$x \wedge 0$

Question: Given N, count no. of set bits

N = 10 1010 => 2 set bits

N = 8 1000 => 1 set bit

N = 7 0111 => 3 set bits

Approach 1: Wrong

int => 32 bits

N = 10

```
int findSetBits(N){
    for(i=0; i<32; i++){
        if( isSetBit(N,i) )
            count++;
    }
    return count;
}
```

C++, Java

32 iterations

~~wrong~~ code

T.C: O(1) ?

int => 32 => 2×10^9

long => 64 => 9×10^{18}

10^{100} ?

T.C is independent of language

In python, the no. of bits used to store an integer is dynamic

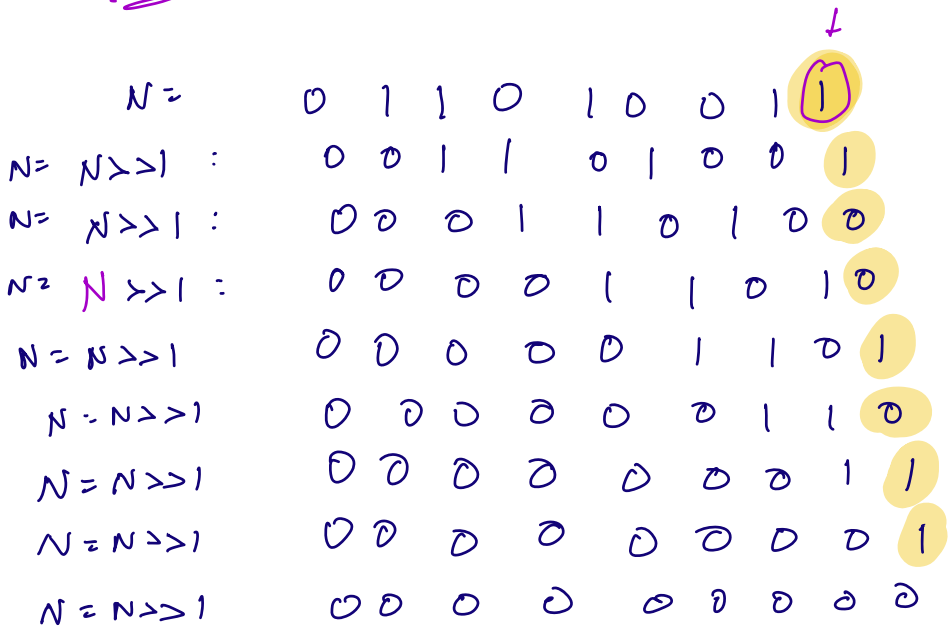
-> we can also store 10^{100} -

As N ↑, No. of bits required to store than number also increases.

#iters => $O(\log_2 N)$

Approach:

Count = 1 + 1 + 1 + 1 + 1




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

T.C: $O(\log_2 N)$

$N \to \frac{N}{2} \to \frac{N}{4} \to \frac{N}{8} \to \cdots\cdots \to 0$

8:36

Ex1 :

1 0 0 0

3 2 1 0

$\Downarrow$

$2^3 = 8$

0 1 1 1

3 2 1 0

$2^2 + 2^1 + 2^0$

$= 7$

Ex2 :

1 0 0 0 0 0 0 0

7 6 5 4 3 2 1 0

$\Rightarrow 2^7 = 128$

0 1 1 1 1 1 1 1

7 6 5 4 3 2 1 0

$\Rightarrow 2^0 + 2^1 + 2^2 + \dots + 2^6$

$= 1 + 2 + 4 + 8 + 16 + 32 + 64$

$= 127$

Eg

1 0 0 0 0 ... 0 0 0

N-1 2 1 0

$\Rightarrow \boxed{2^{N-1}}$

0 1 1 1 ... 1 1 1

N-1 N-2 3 2 1 0

$2^0 + 2^1 + 2^2 + \dots + 2^{N-2}$

$\Downarrow$

$a = 1$ $\{0, 1, \dots N-2\}$

$r = 2$ $[0, N-2)$

$k = N-1$ $N-2-0+1 = N-1$

$S_k = \frac{a(r^k - 1)}{r-1}$

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

$= \boxed{2^{N-1} - 1}$

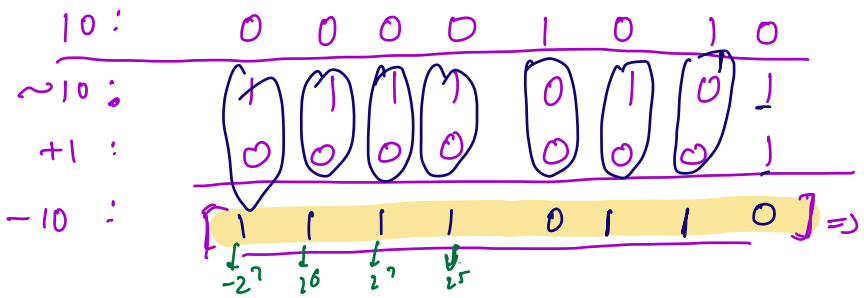
[1 0 0 0 0 0 ... 0 0 0] > 0 1 1 1 1 1 1 1

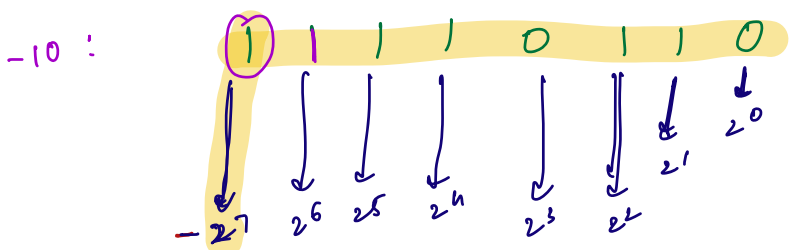
Negative Numbers

Negative number of a is 2's complement of a.

2's complement ⇒ 1's complement + 1

↳ Toggle of a





-2^7

$= 2^7 + 2^6 + 2^5 + 2^4 + 2^2 + 2^1$

$= 128 + 64 + 32 + 16 + 4 + 2$

$= 246$

$1 \times (-2^7) + 2^6 + 2^5 + 2^4 + 2^2 + 2^1$

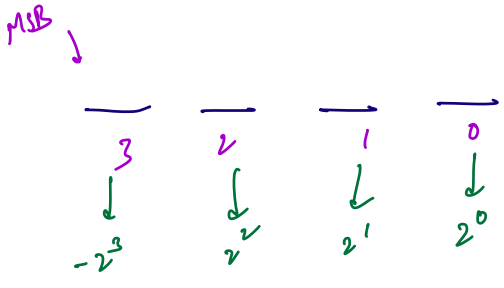
$= -128 + 118$

$= \boxed{-10}$

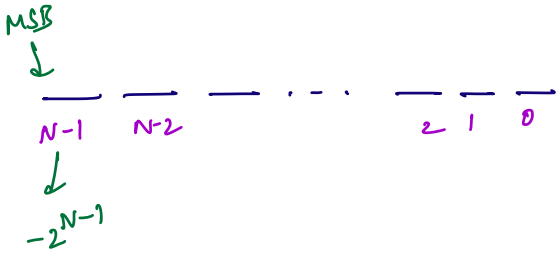
MSB ⇒ 1 [Negative Numbers]

Base value of MSB in

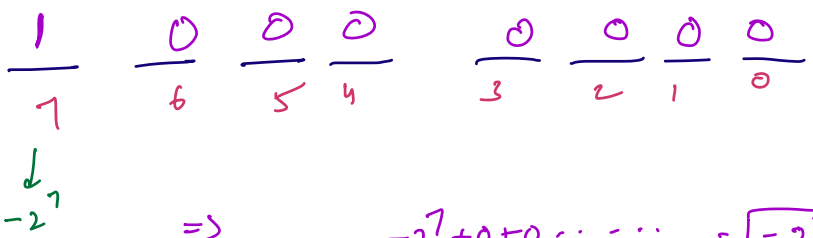
4 bit No.



N bit No

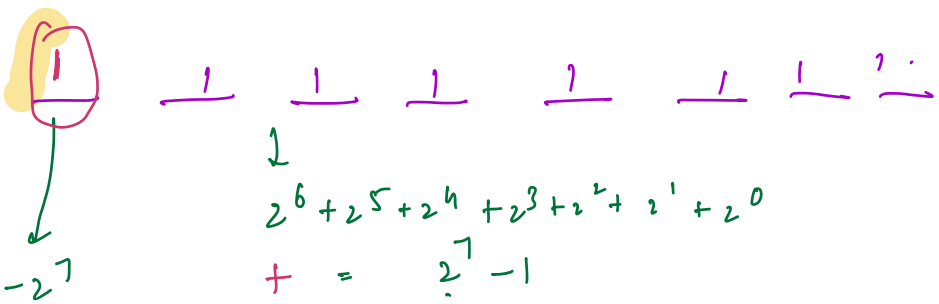


N



$\Rightarrow \quad -2^7 + 0 + 0 \cdots \cdots = \boxed{-2^7}$

N:



$2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

$+ \quad = \quad 2^7 - 1$

$$2^7 > 2^6 + 2^5 + 2^4 + \cdots 2^0$$

Data types

Unsigned Int (+ve Numbers)

Signed Int

(+ve & -ve Numbers)

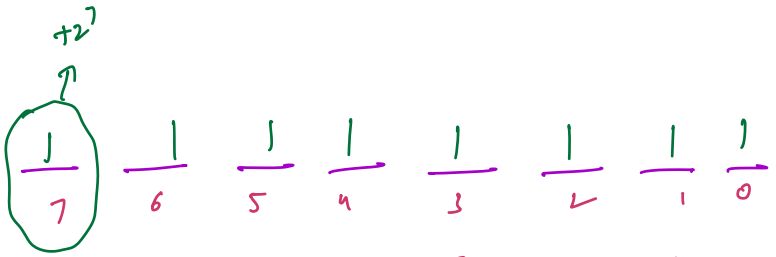
Java / Python => Signed Integer => Base value is -ve

C++ => Unsigned int, int

Ranges:

Unsigned

8 bit: Max:



$2^0 + 2^1 + 2^2 + \dots 2^7 \Rightarrow 2^{8-1} = 255$

Min:

8 bit: [0, 255]

N bit:

Range: $[0, 2^N - 1]$

$2^0 + 2^1 + 2^2 + \dots 2^{N-1}$

Signed Int

4 bits

-2^3

Max: 0 1 1 1 $\Rightarrow$ [-8, 7]

Min: 1 0 0 0 $\Rightarrow$ $[-2^3, 2^3-1]$

-2^3 2^2 2^1 2^0

8 bits

Max: 0 1 1 1 1 1 1 1 $= 2^7-1$

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

Min: 1 0 0 0 0 0 0 0 $\Rightarrow -2^7$

Range: $[-2^7, 2^7-1]$

$[-128, 127]$

N bit

Min: 1 0 0 . . . . . 0 0 0 0 =

(N-1) (3) (2) (1) (0)

-2^{N-1}

Max: 0 1 1 . . . . . 1 1 1 1

(N-1) (3) (2) (1) (0)

$= 2^0 + 2^1 + 2^2 + \cdots 2^{N-2} \Rightarrow 2^{N-1} - 1$

Range of N bit Signed Integer:

$$[\ -2^{N-1},\ 2^{N-1}-1\ ]$$

$N=32 \Rightarrow [\ -2^{31},\ 2^{31}-1\ ]$

$\Rightarrow [\ -2\times10^9,\ 2\times10^9-1\ ]$

$2^{10} = 1024 \simeq 10^3$

$2^{30} = (2^{10})^3 \simeq 10^9$

$2^{31} \simeq 2\times10^9$

long data type

$N=64 \Rightarrow [\ -2^{63},\ 2^{63}-1\ ]$

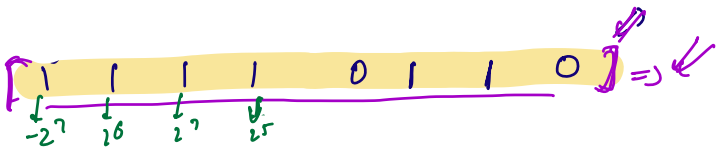
$\Rightarrow [\ -8\times10^{18},\ 8\times10^{18}-1\ ]$

1.5 hr class ⇒ 7am

$-10: \sim 10+1$

[1, 1, 1, 1, 0, 0]

-10 :



$\sim(-10)$ = 0 0 0 0 1 0 0 1

+1

0 0 0 0 0 1 0 1 0

N = -10

Advanced