## Right shift (>>)

8 bits → 1 byte
int → 4 bytes
32 bits


$$x >> k = \frac{x}{2^k}$$

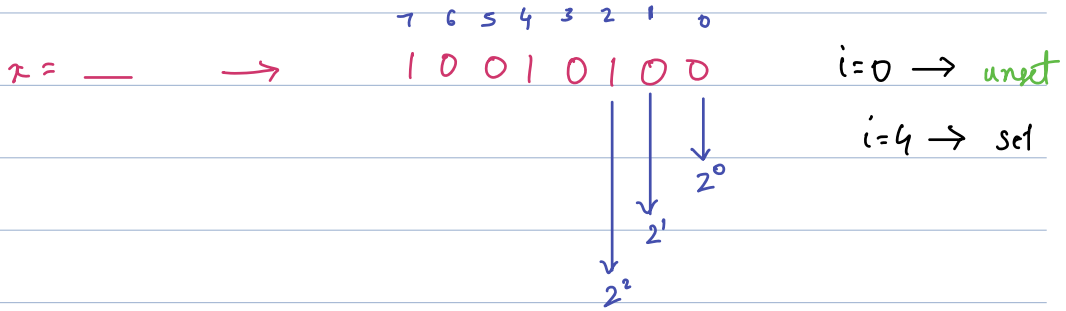
Q1) check if a number is odd / even?

x = _ → Binary 1010**1** ← set → odd

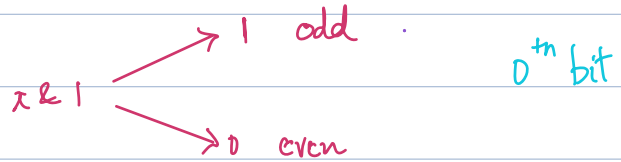
x & 1 → 1 odd
x & 1 → 0 even

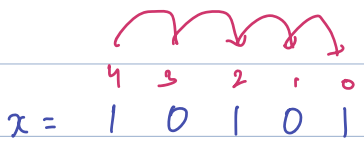
Amazon MCQ

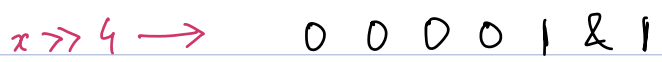
Q) Given a number N and a position i, check if $i^{th}$ (0 based indexed) positioned bit is set or unset?

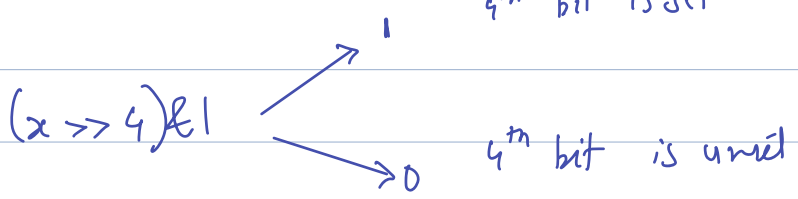






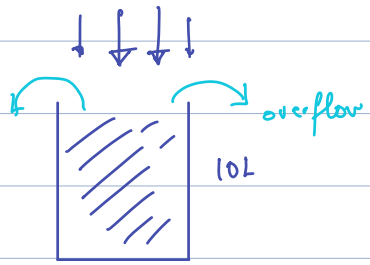

(x >> i) & 1 Ans

# Left shift (<<)

$2^7$ $2^6$ $2^5$ $2^4$ $2^3$ $2^2$ $2^1$ $2^0$ → Base values for $i^{th}$ bits

8 bit

| | | |
|---|---|---|
| x = 1 | 0 0 0 0 0 0 0 1 | → 1 |
| x << 1 | 0 0 0 0 0 0 1 0 | → 2 |
| x << 2 | 0 0 0 0 0 1 0 0 | → 4 |
| x << 3 | 0 0 0 0 1 0 0 0 | → 8 |
| x << 4 | 0 0 0 1 0 0 0 0 | → 16 |
| x << 5 | 0 0 1 0 0 0 0 0 | → 32 |
| x << 6 | 0 1 0 0 0 0 0 0 | → 64 |
| x << 7 | 1 0 0 0 0 0 0 0 | → 128 |
| x << 8 | 0 0 0 0 0 0 0 0 | → 0 |




15L water

bigger bucket

int → 32 bits

↳ long

Microsoft

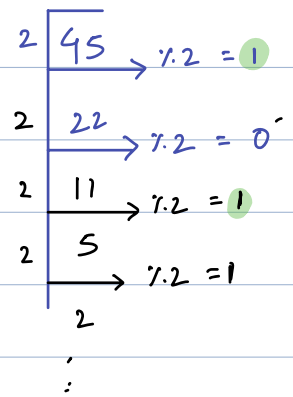
(Q3) Given an number, find total number of set bits in it.

x is positive number

$x = 9$ → binary → 1001

No of set bits : 2

Convert no. to binary and count set bits




```
while (x > 0) {
    if (x % 2 == 1) {
        count++
    }
    x = x/2        // x = x >> 1
}
return count
```

faster

TC: O(log x)

$x >> k = \frac{x}{2^k}$

HFT

x % 2 == 1 odd / even

x & 1

$1 << 0 = 1$

$1 << 2 = 4$

8 bit number

00000001 $<< 2 \Rightarrow$ 00000100

$1 << i = 2^i$

making 000 $\overset{i}{1}$ 0000

$2^n \longrightarrow 1 << n$

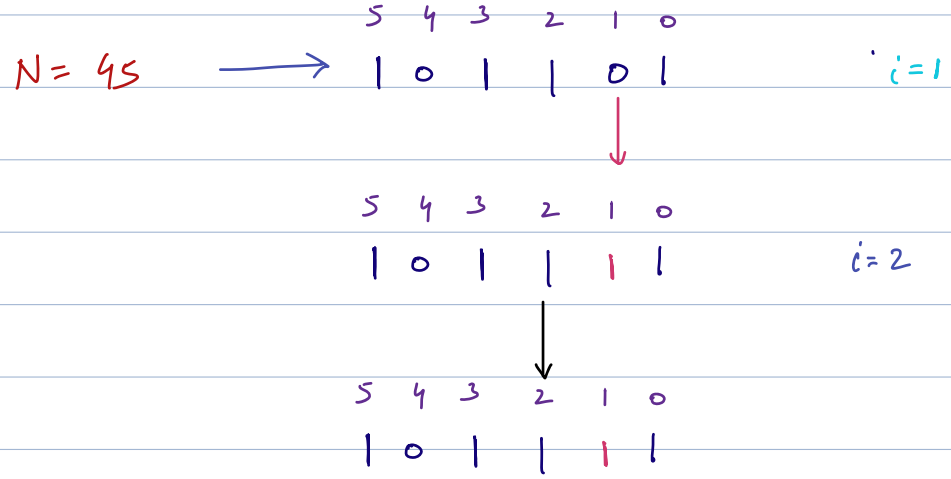
same things

$2^3 \longrightarrow 1 << 3$

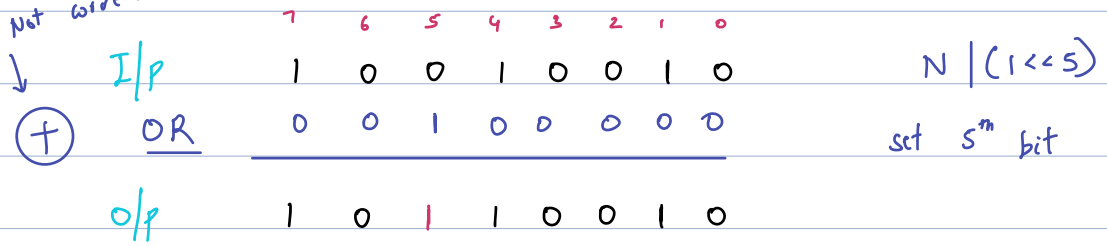
00001000

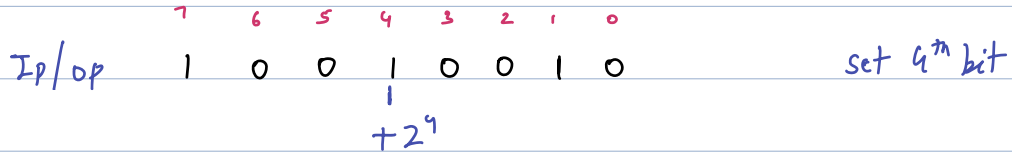
Microsoft MCQ

Q3) Given a number N and a position i Set the $i^{th}$ bit of N




Not correct



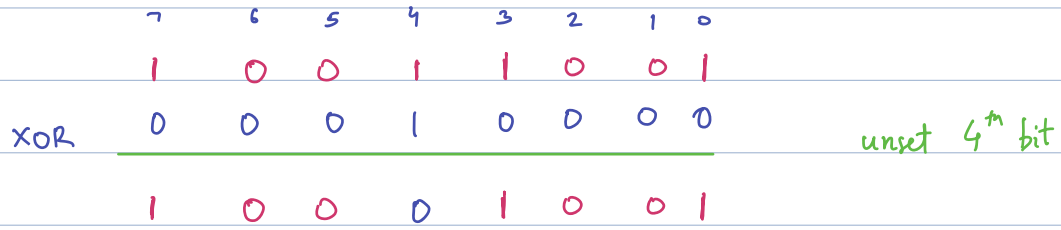




Ans $\boxed{N | (1<<i)}$

# Unset $i^{th}$ bit

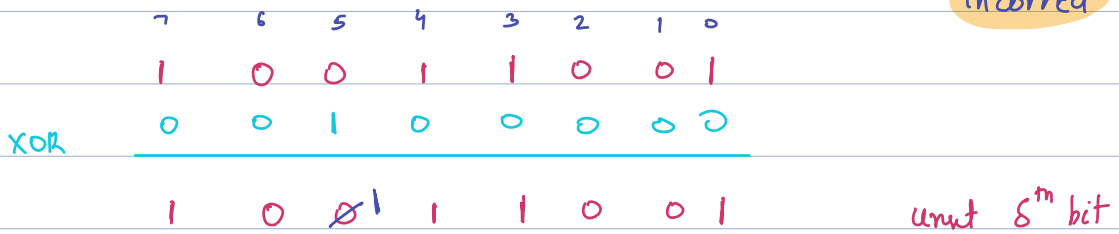
$1 \longrightarrow 0$

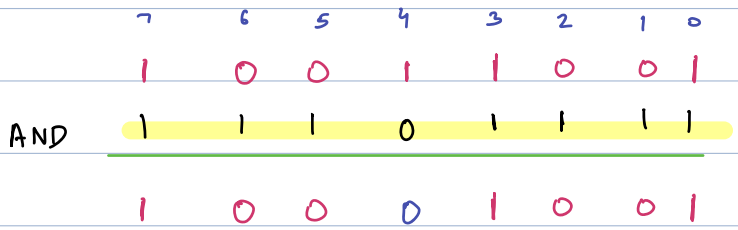
$0 \longrightarrow 0$




$A \wedge 0 = A$

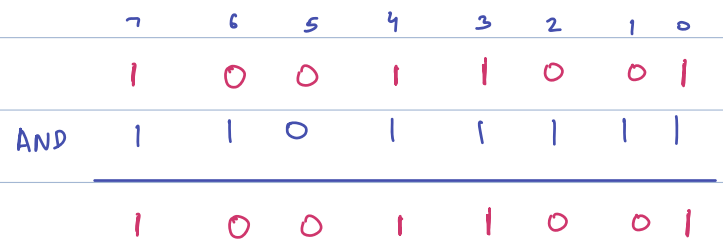
incorrect



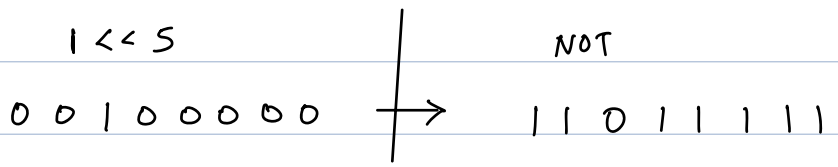




Ans

$\sim(1<<i)\ \&\ N$



1 1 1 0 1 1 1 1




HW : Toggle $i^{th}$ bit

Break (10:19 - 10:30)

unsigned

C, C++, C#, Go

signed

Java, python

C, C++, C#, Go

## Unsigned integer

>= 0

unsigned int x = 5

unsigned int x = -5

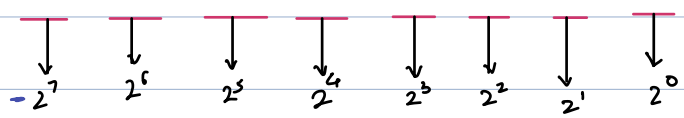
Not right

unsigned cannot handle negative nu


## Signed integer

int x = 5

int x = -5




Most significant bit has negative base value

$(-2)^{com}$ ∝

$-1 \times 2^{n}$

# Negative Numbers

signed 4 bits : $-2^3$ $2^2$ $2^1$ $2^0$

| | | | | | Decimal |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | ⟶ | 7 |
| 1 | 0 | 1 | 1 | ⟶ | -5 |
| 1 | 1 | 1 | 1 | ⟶ | -1 |
| 1 | 0 | 0 | 0 | ⟶ | -8 |

unsigned int

1 0 1 1 —Decimal→ 11

signed int

1 0 1 1 —Decimal→ -5

# How to find binary for -N

8 bits

N = 9 0 0 0 0 1 0 0 1

binary representation of -9

- Number = 2s complement

= 1s complement + 1

$$-N = \sim N + 1$$

N = 9 0 0 0 0 1 0 0 1

1s comp of N : 1 1 1 1 0 1 1 0

+1 : 0 0 0 0 0 0 0 1

7 6 5 4 3 2 1 0

N = -9 1 1 1 1 0 1 1 1

-128 + 64 + 32 + 16 + 4 + 2 + 1

-128 + 119 = -9

Hence proved

N = -3

N = 00000011

~N = 11111100

+1 = 00000001

-3 = 11111101

$-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 1$

$-128 + 125 = -3$

N: 11111101

what is m decimal?

signed

unsigned

N = 10 0 0 0 0 1 0 1 0

~N 1 1 1 1 0 1 0 1

\+ 0 0 0 0 0 0 0 1

1 1 1 1 0 1 1 0

↓

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

$= -128 + 118$

$= -10$

# Ranges

signed

[-8, 7]

4 bits

1 0 0 0 — -8 — Min

0 1 1 1 — 7

8 bits [-128, 127]

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

1 0 0 0 0 0 0 0 — Min — -128 $(-2^7)$

0 1 1 1 1 1 1 1 — 127 $(2^7-1)$

32 bits

1 ... 0 (31 ... 0) — $-2^{31}$

1 1 1 1 ... 1 (31 ... 0) — Max: $2^{31}-1$

$(1+2+4\cdots 2^{30})$ — GP — $a=1$, $r=2$, $n=31$

$a\left(\frac{r^n-1}{r-1}\right)$

int $[-2^{31}, 2^{31}-1]$

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

long : 64 bits

long range : $[-2^{63}, 2^{63}-1]$

logn

int


$10^9$

$2^{31} \longrightarrow$

$2^{10} \rightarrow 1024$

$2^{10} \approx 1000$

$2^{10} \approx 10^3$

$(2^{10}) \approx (10^3)^3$

$2^{30} \approx 10^9$

$2^{31}-1$

$\% (10^9+7)$

Extra HW qn

A = [1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1]

No. of subarrays having or = 1

single 1 in a subarray

Bitwise OR = 1

1) Sum of all subarray $\rightarrow O(N^2)$

subarray > 0

len $\leq 10^5$ → TLE

O(N)

A = [1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1]

No. of subarray having sum > 0 = Total subarray − subarray sum = 0

Total subarray = $\frac{N \times (N+1)}{2}$

Find no. of subarray having sum = 0

10 6 6 1

A = [1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1]

0

0

0

0

0 0

0 0

0 0

0 0 0

0 0 0

0 0 0 0

10 subarrays

10 + 6 + 6 + 1 = 23

[1 0 0 0 0 1]

$\frac{4 \times 5}{2} = 10$

```
long countO = 0 , ans = 0
for (i=0; i<N; i++) {
    if (A[i] == 0) {
        countO++
    }
    else {      // A[i] == 1
        ans = ans + countO * (countO+1)/2
        countO = 0
    }
}
```

`ans = ans + countO * (countO+1)/2`

↓ long

return $\frac{N \times (N+1)}{2}$ − ans

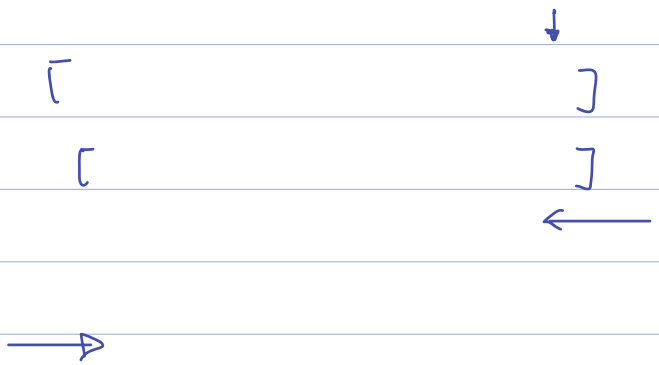
[1 0 0 0 0 0]

N => long

string → int

$2^{31}$ ←

carry : sum/2



long n = N*(N+1)/2;

~~int~~ N → int