Bit Manipulation - II

What is 000100 ?

$\rightarrow 4 \ (1<<2) \quad [2^2 = 1<<2]$

Power of left shift

OR operator

$N = 45 \rightarrow$

5	4	3	2	1	0
1	0	1	1	0	1

	1	0	1	1	0	1
OR (1<<2)	0	0	0	1	0	0
	1	0	1	1	0	1

$\rightarrow 45$

	1	0	1	1	0	1
OR (1<<4)	0	1	0	0	0	0
	1	1	1	1	0	1

$\rightarrow 61$

$45 + 2^4 = 45 + 16 = 61$

$N \mid (1<<i)$ = set i^{th} bit in N if it is unset (means 0)
else NO CHANGE

$= N + (1<<i)$ if i^{th} bit is unset (means i^{th} bit is 0)

$= N$ else

$1<<i = 2^i \Rightarrow 0.0.0\,1\,0\,0\,0\ldots0$

i^{th} bit

XOR operator

$$\begin{array}{rcccccc l}
 & 1 & 0 & 1 & 1 & 0 & 1 & = 45 \\
\text{XOR } (1<<2) & 0 & 0 & 0 & 1 & 0 & 0 & \\
\hline
 & 1 & 0 & 1 & 0 & 0 & 1 & = 41
\end{array}
\qquad
\begin{array}{rcccccc l}
 & 1 & 0 & 1 & 1 & 0 & 1 & \\
\text{XOR } (1<<4) & 0 & 1 & 0 & 0 & 0 & 0 & \\
\hline
 & 1 & 1 & 1 & 1 & 0 & 1 & = 61
\end{array}$$

$$\boxed{N \hat{} (1<<i) \longrightarrow \text{flips / toggles } i^{th} \text{ bit}}$$

$$\begin{aligned}
N \hat{} (1<<i) &= N + (1<<i) \quad \text{if } i^{th} \text{ bit is unset} \\
&= N - (1<<i) \quad \text{if } i^{th} \text{ bit is set}
\end{aligned}$$

AND operator

$$\begin{array}{rcccccc l}
 & 1 & 0 & 1 & 1 & 0 & 1 & \\
\text{AND } (1<<2) & 0 & 0 & 0 & 1 & 0 & 0 & \\
\hline
 & 0 & 0 & 0 & 1 & 0 & 0 & = 2^2 = 4
\end{array}
\qquad
\begin{array}{rcccccc l}
 & 1 & 0 & 1 & 1 & 0 & 1 & \\
\text{AND } (1<<4) & 0 & 1 & 0 & 0 & 0 & 0 & \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & = 0
\end{array}$$

$$N \& (1<<i) \begin{cases} \rightarrow (1<<i) & \text{if } i^{th} \text{ bit is set} \\ \rightarrow 0 & \text{else} \end{cases}$$

Question 1

Unset i^{th} bit of a number if it is set, else DO NOTHING.

$N = 45$ 1 0 1 1 0 1

if $i=2$ $\Rightarrow$ 1 0 1 0 0 1 $\Rightarrow$ 41

if $i=4$ $\Rightarrow$ 1 0 1 1 0 1 $\Rightarrow$ 45

Since we don't know that i^{th} bit is set/unset, we can't toggle. However, we can set and then toggle.

Code

$x = N \mid (1 \ll i)$ → this will set i^{th} bit

$ans = x \wedge (1 \ll i)$ → this will toggle i^{th} bit which means unset i^{th} bit

Alternatively

```
if (checkBit(N, i))
    N = N^(1<<i)
```

Question 2

Check if i^{th} bit is set ?

1. $N \mid (1<<i) == N$
2. $N \wedge (1<<i) < N$
3. $N \& (1<<i) == (1<<i)$
4. $N \& (1<<i) \; != 0$

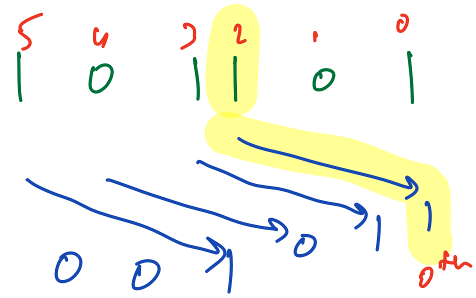
$N\&1 \rightarrow 1$ if 0^{th} bit is set (odd no.)

$\quad\;\; \rightarrow 0$ if 0^{th} bit is unset (even no.)

$N = 45$

$i = 2$

$(N >> i)$



$N = 10011$

$N >> 3 = 00010 = 2$

$(N>>3)\&1 = 0$

5. $(N >> i) \& 1 == 1$

Question 3

Count the number of set bits in N.

int → 32 bits   long → 64 bits

```
count=0
for (i=0; i<32; ++i) {
    if ( (N>>i)&1 )
        ++count
}
```

T.C: O(1)

	3	2	1	0		
N=10	1	0	1	0	N&1 = 0	(N>>0 = N)
N>>1	0	1	0	1	(N>>1)&1 = 1	
N>>2	0	0	1	0	(N>>2)&1 = 0	$\boxed{\text{count=2}}$
N>>3	0	0	0	1	(N>>3)&1 = 1	
N>>4	0	0	0	0	0	
N>>5	0	0	0	0	0	
⋮						
N>>31					0	

count=0

while(N > 0) { $N = N>>1$

if (N&1) (N%2 ==1)

++count

N = N>>1 (N = N/2)

}

N=10 10&1 = 0

N=5 5&1 = 1

N=2 2&1 = 0

N=1 1&1 = 1

N=0 stop

TC: O(1) $\Rightarrow O(\log_2 N)$

Negative numbers

$(-45)_{10} = (\ ?\ )_2$

int → 32 bits

MSB (most significant bit)

31	30	29	.	.	.	-	-	.	2	1	0

$$2^{31} > \underbrace{2^{30} + 2^{29} + 2^{28} + \ldots + 2^2 + 2^1 + 2^0}_{\text{31 terms}}$$

GP $a = 2^0$ $r = 2$ $n = 31$

$$\text{sum} = a\left(\frac{r^n - 1}{r - 1}\right) = 1\left(\frac{2^{31} - 1}{2 - 1}\right) = 2^{31} - 1$$

For explaining negative numbers, we'll use 8 bit numbers.

	7	6	5	4	3	2	1	0
45 →	0	0	1	0	1	1	0	1

↓ flip all bits

1's complement of 45 →	1	1	0	1	0	0	1	0
+1 →	1	1	0	1	0	0	1	1

$2^7 + 2^6 + 2^4 + 2^1 + 2^0 = 128 + 64 + 16 + 2 + 1$

$= 211$ ✗ -45

↓

$-2^7 + 2^6 + 2^4 + 2^1 + 2^0$

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

$= -45$

1 1 1 0 0 1 1 1

$-2^7 \quad 2^6 \quad 2^5 \qquad 2^2 \quad 2^1 \quad 2^0$

$= -128 + 64 + 32 + 4 + 2 + 1$

$= -25$

Two types of datatypes $\nearrow$ unsigned
$\searrow$ signed

Before negative numbers (Unsigned)

Min val. : 0 0 0 0 0 0 0 0 = 0
Max. val : 1 1 1 1 1 1 1 1 = 255

8 bit unsigned int

After negative numbers (signed)

Min val. : 1 0 0 0 0 0 0 0 $= -2^7 = -128$
Max val. : 0 1 1 1 1 1 1 1 $= 2^7 - 1 = 127$

normal 8 bit signed int

$(2)_{10}$ = 0 0 0 0 0 0 1 0

1's compl = 1 1 1 1 1 1 0 1
+1

+1 = 1 1 1 1 1 1 1 0 $= (-2)_{10}$

$-2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1$

$= -128 + 64 + 32 + 16 + 8 + 4 + 2$

$= -2$

1 0 0 0 0 0 0 1

$-2^7 + 1 = -127$

For 32 bit integers (signed)

Min val : -2^{31} $= -2147483648 \approx -2\times10^9$

Max val : $2^{31}-1$ $\approx 2\times10^9$

For 64 bit integers (long)

Min val : -2^{63} $\approx -9\times10^{18}$

Max val : $2^{63}-1$ $\approx 9\times10^{18}$

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

$2^{60} \approx 10^{18}$

$2^3 \times 2^{60} \approx 8\times10^{18}$

$2^{63} \approx 8\times10^{18}$

Question 4

Calculate sum of all elements in an array.

constraints: $1 <= N <= 10^5$

$1 <= A[i] <= 10^6$

~~int~~ → long sum = 0

for i = 0 to N-1

sum += A[i]

return sum

$A = [10^6\ 10^6\ \ldots\ 10^6]$

$N = 10^5$

$sum = 10^6 \times 10^5 = 10^{11} > 2\times10^9$

constraints → TLE

↳ overflow

Multiply 2 numbers

int a, b   a, b <= 2×10^9

Find $a \times b$?

$a \times b <= 4 \times 10^{18}$

we need long

```
int ans = a x b;   X
long ans = (a x b);   X
```

overflow will happen at multiply step

CPU

```
STORE a
STORE b
MUL a, b to temp
COPY temp to ans
```

overflow step → MUL a, b to temp

```
long ans = (long)(a x b);   X
```

$$\text{long ans} = \underbrace{((\text{long})a)}_{\text{long}} \times \underbrace{b}_{\text{int}}; \checkmark$$

long × int = long

```
long ans = (long)a x (long)b   ✓
```

```
long ans = a;
ans x= b;   ✓
```

Subtract 2 binary numbers [8 bits]

$45 - 12$

$12 = 0000\ 11\ 00$

1's complement of 12 $= 1111\ 0011$

2's (+1) $= 11110100 \Rightarrow (-12)$

$45 + (-12)$

↳ 2's complement of 12

$$\begin{array}{rcccccccccc}
 & & & 1 & 1 & 1 & 1 & 1 & & & \\
45 & = & & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
-12 & = & & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
\hline
 & & \cancel{1} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}$$

discard

$2^5 \qquad 2^0$

$2^5 + 1 = 33$

$$\boxed{\text{1's complement } + 1 = \text{2's complement}}$$