Today's Content

1) Negative numbers Representation

2) Left Shift & Right Shift

3) 5 Interesting / Simple Bitwise problems

4) 1 Advanced Bitwise problem

8 bit

10 : 0 0 0 0 1 0 1 0

-10 : 1 0 0 0 1 0 1 0 ✗ ⇒ -128 + 8 + 2 ⇒ -118

2's complement = $-a = \sim a + 1$ } Proof in Today's Doubts

$\sim a$ is Nothing but its Complement

0 1

~10 : 1 1 1 1 0 1 0 1

+1 : 0 0 0 0 0 0 0 1

2%2

1 1 1 1 0 1 1 0 → Decimal Representation

-128 64 32 16 8 4 2 1

⇒ -128 + 64 + 32 + 16 + 4 + 2 = -128 + 118 = -10

23 : 0 0 0 1 0 1 1 1

~23 : 1 1 1 0 1 0 0 0

+1 : 0 0 0 0 0 0 0 1

1 1 1 0 1 0 0 1

$-128 + 64 + 32 + 8 + 1 = -128 + 105 = -23$

30 : 0 0 0 1 1 1 1 0

~30 : 1 1 1 0 0 0 0 1

+1 : 0 0 0 0 0 0 0 1

1 1 1 0 0 0 1 0

$-128 \quad 64 \quad 32 \quad 2 = -128 + 98 = -80$

left shift

8 bit Number

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
$a = 10$:	0	0	0	0	1	0	1	0	: 10
$a << 1$:	0	0	0	1	0	1	0	0	: $2^4 + 2^2 = 20$: $10 * 2$
$a << 2$:	0	0	1	0	1	0	0	0	: $2^5 + 2^3 = 40$: $10 * 2^2$
$a << 3$:	0	1	0	1	0	0	0	0	: $2^6 + 2^4 = 80$: $10 * 2^3$
$a << 4$:	1	0	1	0	0	0	0	0	: $-2^7 + 2^5 = -128 + 32 = -96$

$a << 4$:

→ $a * 2^4 \Rightarrow 10 * 2^4 \Rightarrow 160$

overflow 8 bit cannot store 160

→ $a = 10$

→ In 8 bit Number : [-128, 127]

$a = 127 =$	0	1	1	1	1	1	1	1
$(a+1)$ $+1 =$	0	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	0

$127 + 1 = 128$

-128

$a = 15$, $15 << 2$

0	0	0	0	1	1	1	1
0	0	1	1	1	1	0	0
		32	16	8	4		

$\Rightarrow 60$

$\Rightarrow 15 * 2^2$

$$a << n = a * 2^N$$

$a = -1$: 1 1 1 1 1 1 1 1   Decimal :

$a << 1$: 1 1 1 1 1 1 1 0 = -2

$-1 << 1 = -1 * 2^1 = -2$ ⇒ 8 bit Number ⇒ No Overflow

Right Shift

8 bit Number

	-128	64	32	16	8	4	2	1	
$a = 40$:	0	0	1	0	1	0	0	0	LSB discarded → Disc → 40 :
$a >> 1$:	0	0	0	1	0	1	0	0	: $2^4 + 2^2 \Rightarrow 20 : 40/2^1$
$a >> 2$:	0	0	0	0	1	0	1	0	: $2^3 + 2^1 \Rightarrow 10 : 40/2^2$
$a >> 3$:	0	0	0	0	0	1	0	1	: $40/2^3 \Rightarrow 5$
$a >> 4$:	0	0	0	0	0	0	1	0	: $40/16 \Rightarrow 2$

↳ Integer division

$a = 50$: $a >> 2$: $50/2^2 \Rightarrow 50/4 \Rightarrow 12$

$a = 10$: $a >> 4$: $10/2^4 \Rightarrow 10/16 \Rightarrow 0$

+ve positive Numbers

$a >> N : a/2^N$

No overflow

TODO : Right Shift for Negative Numbers

Problems → Check if i^{th} bit of N is Set or Unset

Set → 1, Unset → 0

$\{1 <= N \leq 10^9$, $[0 \leq i \leq 30]\}$ → i is bit position

```
bool CheckBit(int N, i){

}
```

N = 21, i = 2 → **Set**

4	3	2	1	0
1	0	1	0	1

N = 25, i = 2 → **unset**

4	3	2	1	0
1	1	0	0	1

N = 24, i = 2 → unset → false

4	3	2	1	0
1	1	0	0	0

N = 14, i = 3

3	2	1	0
1	1	1	0

→ set → True

Example

	7	6	5	4	3	2	1	0
N = 106 :	0	1	1	0	1	0	1	0

i = 0 : (N & 1) == 1, set else unset

	7	6	5	4	3	2	1	0
N = 106 :	0	1	1	0	1	0	1	0

i = 1 : (N >> i) & 1 == 1

N = 106

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

i = 0 : (N & 1) $\neq$ 0

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

N & (1 << 2) == 0

2nd bit is unset

7 6 5 4 3 2 1 0

N = 106 : 0 1 1 0 1 0 1 0

i = 3 : (N >> 3) & 1 == 1

```
bool checkBit (N, i){
    if((N>>i) & 1 == 1){
    { return True }
    else { return False }

    return (N>>i) & 1 == 1
}
```

```
bool checkBit (N, i){
    if(N & (1<<i) != 0){
        return true
    }
    else { return False }
}
```

10:30 PM breaks

7 6 5 4 3 2 1 0

	0	1	1	0	1	0	1	0
	0	0	0	0	1	0	0	0
8 =	0	0	0	0	1	0	0	0

i = 3

(1<<3)

N & (1<<3) != 0

3rd bit is set

Doubts

a = 10

a + 10 → 20

print(a) → 10

a = 10

a >> 1 → 5

print(a) → 10

Note: All bitwise operators only on Integer Datatypes

Given Integer N calculate No: of set bits are present

CountSetBits(N) : $0 \le N \le 10^9$ → int : 32 bits

int

[31, 30 . . . 0]

N = 25 : 1 1 0 0 1 : 3 bits

N = 40 : 1 0 1 0 0 0 : 2 bits

N = 35 : 1 0 0 0 1 1 : 3 bits

```
countSetBits(N){
    cnt = 0
    i = 0; i <= 30; i++){
        if( checkBit(N, i)){
            cnt++
        }
    }
    return cnt;
}
```

N = 109

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

	5	4	3	2	1	0
N>>1 :	1	1	0	1	1	0

```
while( N > 0 ){
    (N & 1) == 1
        // cnt++
    N = N >> 1
}
return cnt;
```

N = N/2

$\approx (\log_2 N)$

If N = 27 :

4	3	2	1	0
1	1	0	1	1

N = 27, 1st code : 31 Iterations

N = 27, 2nd code : 5 iterations

→ Prerequisites

$$\begin{array}{lcccccc} & & 4 & 3 & 2 & 1 & 0 \\ 11 & : & 0 & 1 & 0 & 1 & 1 \\ 25 & : & 1 & 1 & 0 & 0 & 1 \\ \hline 11 \wedge 25 & : & 1 & 0 & 0 & 1 & 0 \end{array}$$

At 4^{th} bit both are diff

At 1^{st} bit both are diff

$a \wedge b = 0$ $\quad a == b$

Given +ve Integer N, find least significant set bit position.

$$\begin{array}{lccccccc} & & 5 & 4 & 3 & 2 & 1 & 0 \\ N = 44 & : & 1 & 0 & 1 & 1 & 0 & 0 \end{array}$$

```
i = 0; i <= 30; i++){
    if( checkBit(N, i)){
        ans = i; break
    }
}
```

Given N array elements, every element repeats twice except 2 [unique elements], find the 2 unique elements

output

Ex1: 3 4 5 4 7 5 : 3 7

Ex2: 4 9 9 8 : 4 8

Eg 3:

(1010) (1000) (1000) (10101) (1001) (1100) (1001) (0110) (1011) (1010) (0110) (1100) (10001) (10101)

10, 8, 8, 21, 9, 12, 9, 6, 11, 10, 6, 12, 17, 21

Step 1: xor of all elements

→ xor of 2 unique elements

	4	3	2	1	0
17 :	1	0	0	0	1
11 :	0	1	0	1	1
17 ^ 11 :	1	1	0	1	0

At bit pos = 1 both 17 & 11 are different

Can we split on 0^{th} bit : No

1^{st} bit : YES

2^{nd} bit : NO

3^{rd} bit : YES

4^{th} bit : YES

let's split array

1^{st} bit set → set: { 10, 6, 11, 10, 6 } → xor of all elements: 11

1^{st} set unset → unset: { 8, 8, 21, 9, 12, 9, 12, 17, 21 } → xor of all elements: 17

let's split the array

3^{rd} bit set: { 10, 8, 8, 9, 12, 9, 11, 10, 12 } → 11

3^{rd} bit unset: { 21, 6, 6, 17, 21 } → 17

Pseudo Code

1) val = xor of all Elements

2) pos = Get any set pos in of val

3) Split based wether a number is set or not at pos^{th}

---

xr = 17^9

	4	3	2	1	0
17 :	1	0	0	0	1
9 :	0	1	0	0	1
	1	1	0	0	0

17   9

```
val = 0
i = 0; i < N; i++) {
    val = val ^ ar[i]
}
pos = 0
i = 0; i <= 30; i++) {
    if( checkBit (val, i)) {
        pos = i; break;
    }
}
set = 0, unset = 0
i = 0; i < N; i++) {
    if (checkBit (ar[i], pos)) {
        set = set ^ ar[i]
    }
    else {
        unset = unset ^ ar[i]
    }
}
print (set, unset)
```

2nd Complement

$x \geq 0$ N bit {Doubts}

0 | 1 0 0 ... 0 1 0 1 0 0 1 0

2^{N-1} | 2^{N-2} 2^{N-3} ... 2^0

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

$\sim x$

1 | $\underbrace{0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0}\ 1\ 1$

-2^{N-1} $\quad 2^{N-1} - 1 - x$

$\Rightarrow \cancel{-2^{N-1}} + \cancel{2^{N-1}} - 1 - x$

$\sim x = -1 - x$

$\boxed{-x = \sim x + 1}$

$2^s\ x = \sim x + 1$

Ex:

[2 ^ 3 ^ ~~7~~ ^ 3 ^ ~~1~~ ^ 2] = 6

6 ^ 2

110
010
100

```
i = 0; i < N; i++) {
    j = i; j < N; j++) {
        if (ar[i] ^ ar[j] == 0) {
        }
    }
}
```

~~3~~ 4 6 7 7 4

-1 1 -1 1