Basic Operators & , | , ^ , ~

<< , >>

Left Shift (<<)

$a << n = a \times 2^n$

$a = 10$: 0 0 0 0 1 0 1 0 : 10

discard

$a << 1$: 0 0 0 1 0 1 0 0 : $20 = 10 \times 2^1$

discard

$a << 2$: 0 0 1 0 1 0 0 0 : $40 = 10 \times 2^2$

$a << 3$: 0 1 0 1 0 0 0 0 : $80 = 10 \times 2^3$

$a << 4$: 1 0 1 0 0 0 0 0

$-(2^7)$

$-(2^7) + 2^5$

$-128 + 32$

$= -96$

160 ⇒ 8 Bits] Overflow

Quiz 1

$a =$ 0 0 0 0 1 1 1 1

$a << 2 =$ 0 0 1 1 1 1 0 0 $15 \times 2^2 = 60$

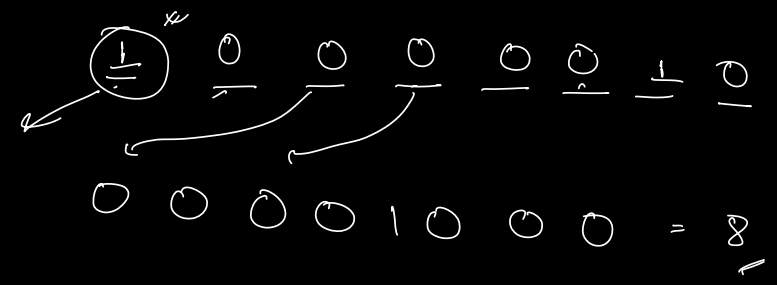
Quiz 2 $\quad a << n = a \times 2^n$

Quiz 3

$-a = \sim a + 1$

$a = -126$

$a << 2$



Limit of 8 bit $\quad$ 504 ✗

$[-128, 127] \Rightarrow$ Overflow

GOT $\Rightarrow$ 8 $\quad$ 8th's ✗

Right Shift (>>)

$a >> n = a / 2^n$ $\quad$ +ve

1 Byte data-type

$a = 40$ $\quad$ 0 0 1 0 1 0 0 0 discard

$a >> 1$; 0 0 0 1 0 1 0 0

$= 20$

$= 40/2^1$

$a >> 2$

$= 40/2^2$

00001010

Quiz 4

$a = 35$ 0 0 1 0 0 0 1 1

$a >> 2$ 0 0 0 0 1 0 0 0

$35/4 = 8$

Quiz 5

$a >> n = a/2^n$

Quiz 6

$25 >> 6 = 25/2^6 = 25/64 = 0$

Quiz 7

$a = -104$

$a >> 1$

1 1 0 0 1 1 0 0 0

a = -1

1 1 1 1 1 1 1 1

a >> 1

1 1 1 1 1 1 1 1

= -1

a >> 2

a >> 1024

```
byte a = -1;
while (a != 0) {
    a = a >> 1;
}
```

Hw

1. Take some +ve no. & some -ve no.
2. Try a >> x , a << x
3. Write code & verify.

Q — Amazon

Given an integer (+ve) N.
Check if i^{th} bit is set.

$N = 21$, $i = 2$

1 0 (1) 0 1
4 3 2 1 0

⇒ True

$N = 25$, $i = 2$

1 1 (0) 0 1

⇒ False

$N = 24$ 1 1 0 0 0

N = 1 0 1 1 1 0 0 ?
& 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 1/0

? → 1 odd
? → 0 even

N & 1
- odd → 1
- even → 0

$i = 1$

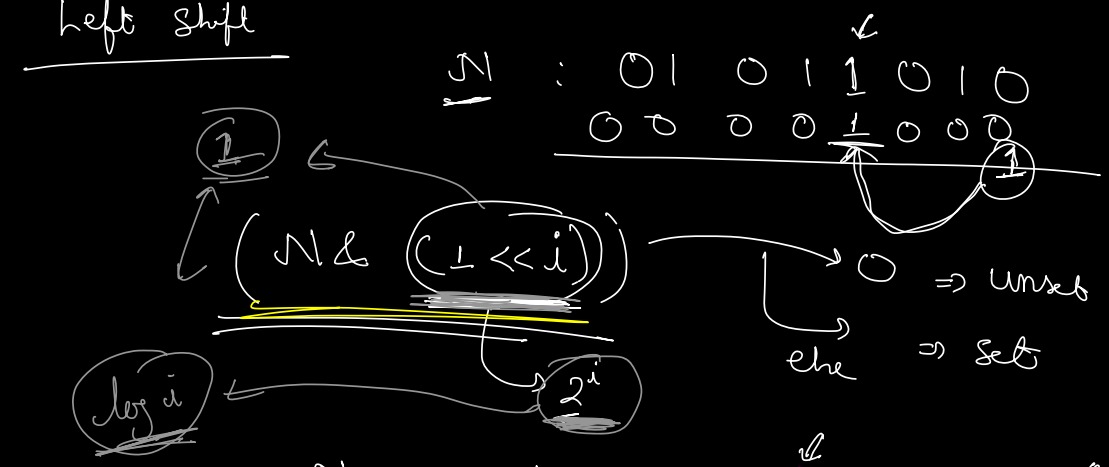
$(N >> 1)$ & 1 → a[1]

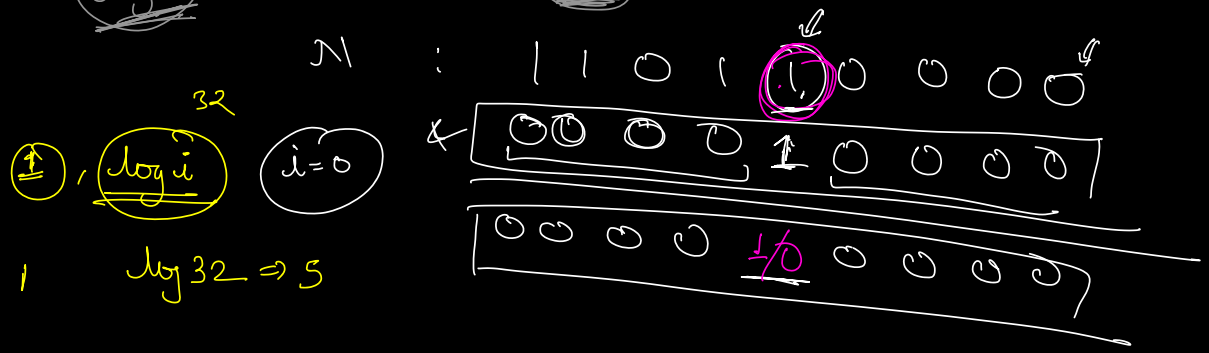
0 1 0 1 1 0 1 0
0 0 1 0 1 1 0 1

i^{th} pos

$(N >> i)$ & 1
- == 1 ⇒ Set
- == 0 ⇒ Unset.

Left Shift



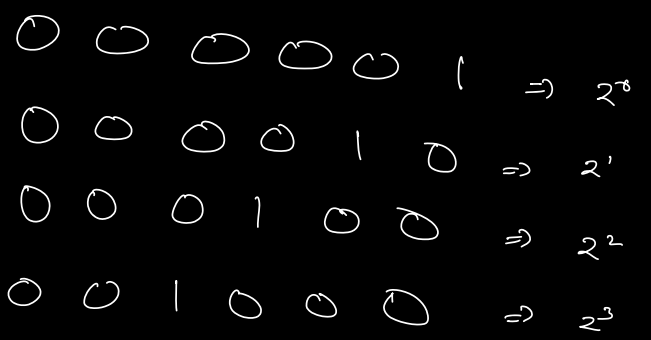


32

1, log i i=0

1 log 32 ⇒ 5

1 5



Q Given a number N. Set the ith bit.

N = 4 0 0 0 0 1 0 0

i = 1

0 0 0 0 1 1 0 = 6

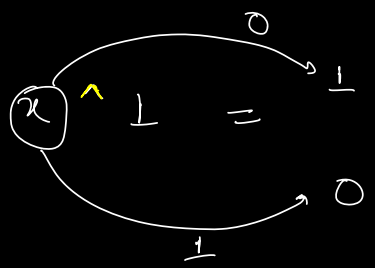
O(log N) ⇒ 0 → 1

N = 0 1 0 1 1 1 0 1 (bits 7 6 5 4 3 2 1 0)

$1 << 2$ = 0 0 0 0 0 1 0 0

$i = 2$

$N \mid (1 << i)$ = 1



Toggle Bit

$1 \to 0$

$0 \to 1$

```
int SetBit (int N, int i) {
    ret N | (1 << i);      O(1)
}
```

Toggle

HW Unset (N, i) 15 min

N = 0 1 1 0 0 1

$i = 3$

$\Rightarrow$ 0 1 0 0 0 1

Break 10:43

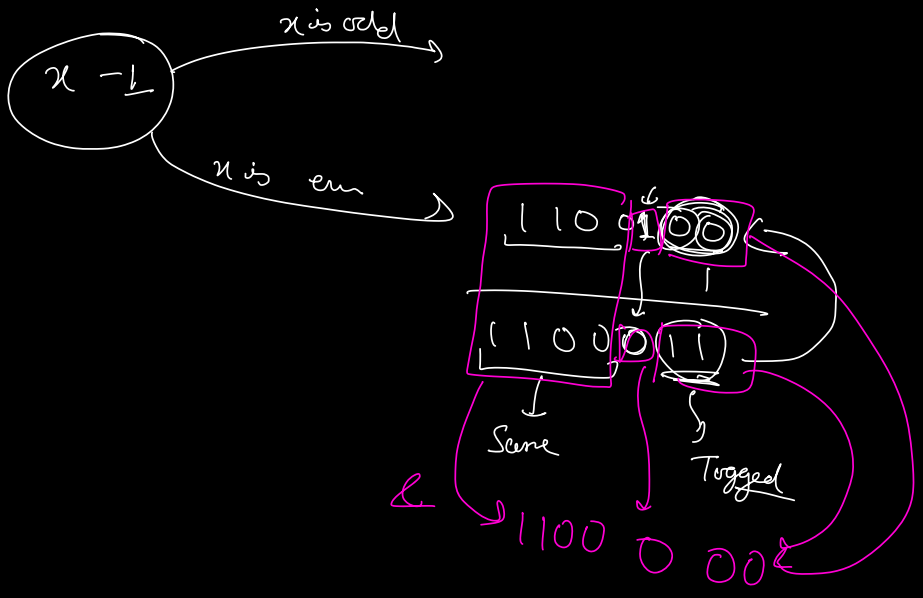
Bit Mask DP

Q Count set bits in a no. (true)

N & (N-1) ×

Unset the first set bit from the right side.

N	N-1	N & (N-1)
4 : 100	011	000
8 : 1000	0111	0000
16 : 10000	01111	00000
20 : 10100	10011	10000
24 : 11000	10111	10000
44 : 101100	101011	101000



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

$N \xrightarrow{/2} N/2 \xrightarrow{/2} N/4 \xrightarrow{/2} \cdots \xrightarrow{/2} 0$

$\log N$

$N = N/2;$

```
int CountSetBits(N){
    int C = 0;
    while (N > 0){
        N = N & (N-1);
        C++;
    }
    ret C;
}
```

$N = 0\,1\,1\,0\,1\,0\,1$

$0\,1\,1\,0\,1\,0\,0$

$0\,1\,1\,0\,0\,0\,0$

$0\,1\,0\,0\,0\,0\,0$

$0\,0\,0\,0\,0\,0\,0$

Q

Code Natives *
Interview Bit
Direct i *

Given an array of +ve integers.
All the no. appear thrice except two no.
Return the two single no.

A : [5, 3, 2, 5, 6, 2, 9, 3, 8, 8]

[6, 4]

Without any extra space (Count freq / HashMap)

X

Xor all elements

a ^ b ^ a ^ c ^ d ^ e ^ e ^ b

c ^ d

XOR = 101

001 010 000

Part I

```
XOR = 0;
for (i=0 ; i<n; i++){
    XOR = XOR ^ A[i];
}
// XOR = Sing1 No1 ^ Sig No 2
```

10^9 10^9 5, 3, 2, 5, 6, 2, 4, 3, 8, 8

101 011 010 101 110 010 100 011 1000 1000

⇒ O(N)

XOR = 6 ^ 4

100 ⇐

100 ⇐

XOR = 010

0 ^ 7

111

Unset

Set

3	011
2	010
6	110
2	010
3	011

Single I

5	101
5	101
4	100
8	1000
8	1000

Single II

Part II

```
// Find the pos of any set bit
i=0;
While(XOR >0){
    if( XOR & 1 == 1){
        pos = i;
    }
    i++;
    XOR = XOR >> 1;
}
```

⇒ O(logN)

⇒ pos → position of a set bit in a ^ b

```
ans1 = 0;  ans2 = 0;
for(i=0; i<n; i++){
  // if A[i] has a set bit at "pos" the collect in ans1
     else in ans2
  if( (A[i] & (1<<pos)) != 0)   ans1 = ans1 ^ A[i];
  else  Ans2 = Ans2 ^ A[i];
}
```

⇒ O(N)

ans 1 ⟶ a

ans 2 ⟶ b

$O(N) + O(\log N) + O(N)$

↑ XOR ↑ Pos ↑ ans

$3 \wedge 2 \wedge 6 \wedge 2 \wedge 3$ $= O(N)$

⇒ 5, 3, 2, 5, 6, 2, 4, 3, 8, 8

101 011 010 101 110 010 100 011 1000 1000

$1 \wedge 0 = 1$, $0 \wedge 0 = 0$

XOR = $6 \wedge 4$ = 010

Pos = 1

$a \wedge a = 0$

$3 \wedge 3 \wedge 2 \wedge 2 \wedge 6$

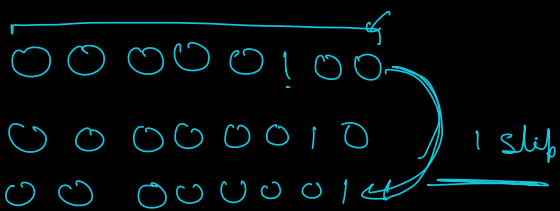
ans 1 = $0 \wedge 3 \wedge 2 \wedge 6 \wedge 2 \wedge 3$ = 6

ans 2 = $0 \wedge 5 \wedge 5 \wedge 4 \wedge 8 \wedge 8$ = 4

Doubts

4

4 << 2



2 × 2 ⇒ ⟶

4 % 2 ⟶

<< >> & | ^

Check Bit

0th bit

$2 \mid 9$
$4 \quad (1)$

9 — odd → 1001
8 — even → 1000
7 — odd → 111
6 — even → 110

1st

9 → 1 0 0 1

(9>>1)

L → R

0100

3rd

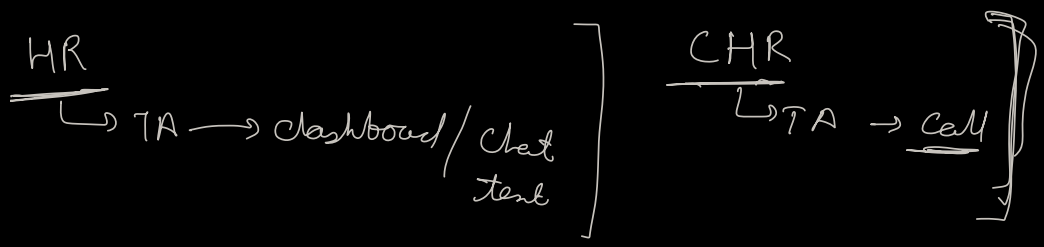
9 → 0010

9>>2

What if we have 72 Single No.

AP	Prime No.	$2^{10} \approx 1000$
GP	%	
HP	Pow	log
	$(a^m)^n$	

Linear bin


HR
TA → Dashboard / Chat text
CHR
TA → Call