Bit Manipulation - 2

Jun 8, 2022

AGENDA

- Some more bitwise operators
- Bit Masking concepts
- Some interesting problems

No class this Friday. (Jun 10)
We will continue with Bit Manipulation - 3 on Monday.

(*) 1 XOR 44

1 = 00000..... 1
44 = xxxxxxx..... 0
1^44 xxxxxxxx..... 1 = 45 ✓

Last digit of even no. in binary is 0
odd is 1

2|43 — 0/1

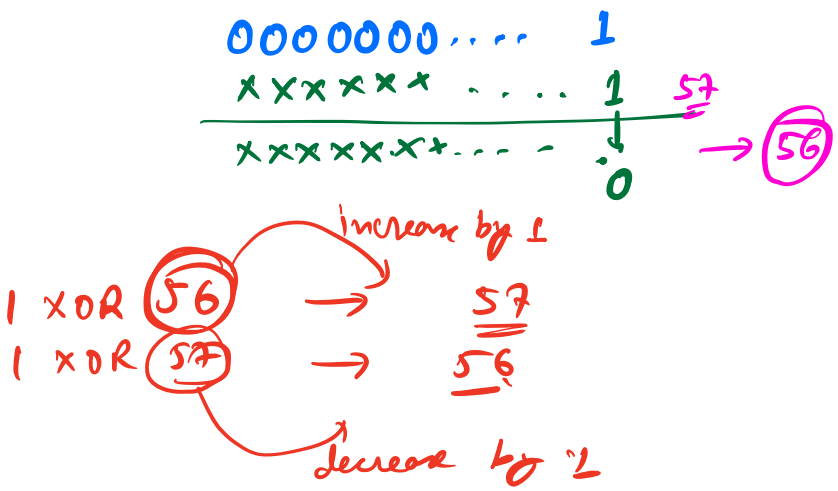
(**) 1 AND 46

1 = 0000000..... 1
46 = xxxxxxx..... 0
1&46 = 0000000...0 = 0

(**) 1 OR 43

1 = 0000000..... 1
43 = xxxxxx..... 1
OR (43) xxxxxx..... 1

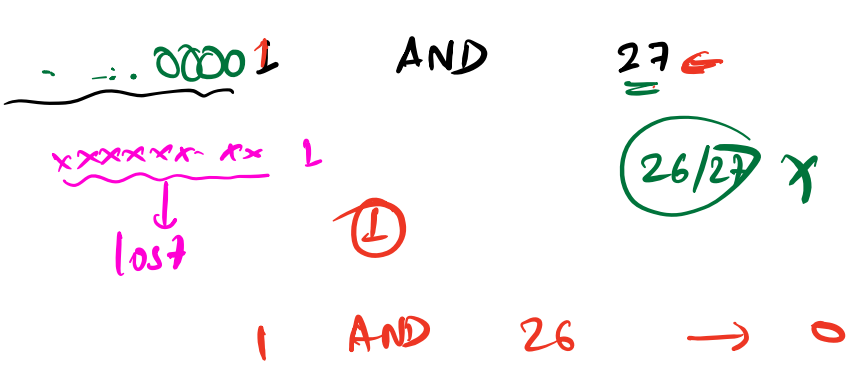
(xx) 1 XOR 57



(xx)

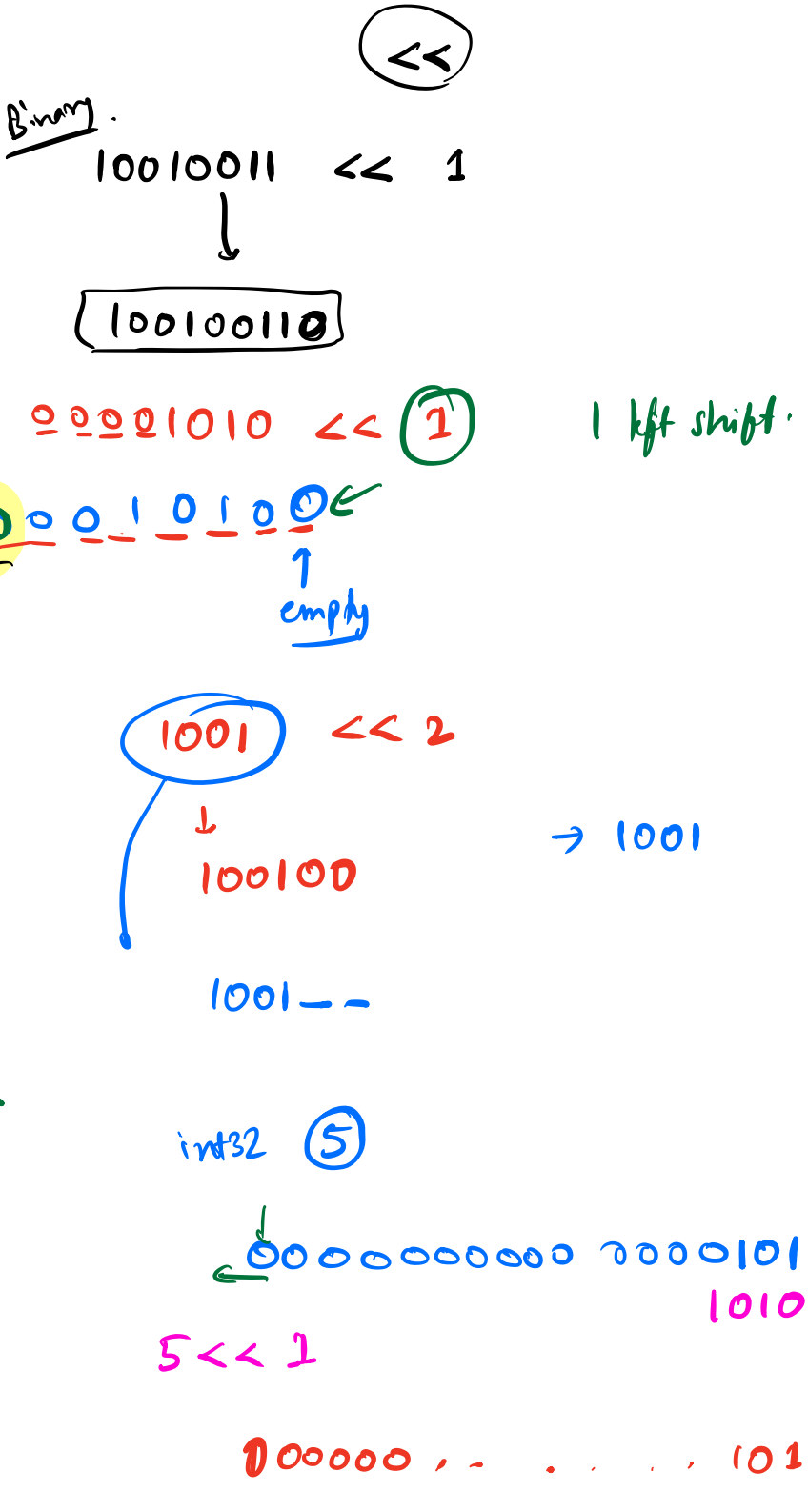
1 OR 27 = 27

(xx)



1 AND 26 → 0

* Left shift operator.



** If we Left shift. we multiply by 2.

8 bit variable

0 0 0 0 0 1 0 1

0 0 0 0 1 0 1 0

0010 << 2

8 bit
4 bit
16 bit
32 bit
64 bit
} = 8

1000 = 8. ✓

00000010 <<2

= (8)

Why do we fillo 0?

Because it is the definition

* 15 << 2

00000000 1111 = (A) → $1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

000000 1111 00 → $1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^2$

3 → 5 2 → 4

↓ $A \times 2^2$

* General relation

$a << b = a * 2 * 2 * 2 * 2 * \cdots * 2 * 2$ (b times)

$= a * 2^b$

No. of time we are shifting

** Every time we shift, no. gets multiplied by 2.

Q. $1011 \;\&\; ((\sim 0) << 2)$

AND, NOT, Left shift, lst., ±

~ 0000

$1111 << 2$

1100

1011

$+\ 1000$

~ 0

$\sim$ (00000 . . . . 0)

11111111111 . . . . . 111)

$\sim 0 \neq 1$

$\sim (0000000 \ldots)$

(1111111111111)

$\sim 0 =$ 11111 . . . . . 1111 ~

11111 . . . . 1111 $<< 2$

$=$ 11111 1100

```
    111111......1100
AND 0000.....000 1011
---------------------
                 1000  = 8
```

0 is int

32bits.

000000...

Quiz

N & (1<<i)

1101 & (1<<3)

```
0001
 ↓
1000
1101
----
1000  = 8
```

* **Right shift.**

```
_ _ _ _1011  >> 2     (11 → Lost)
00_ _ _ _ _ _ _ _10
```

1011 >> 3 = 0001 = 1 (011 Lost)

$29 >> 2 \quad =$

$11101 \quad >>2 \quad 00111 \leftarrow (7)$

(**) Every time you do right shift by 1, no. gets divided by 2.

$25 >> 1 = 12$

$24 >> 1 = 12$

25 11001 → get lost.

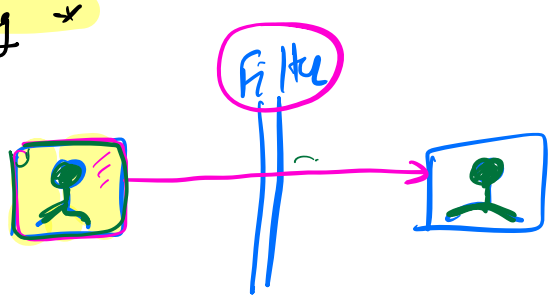
24 11000

General relation

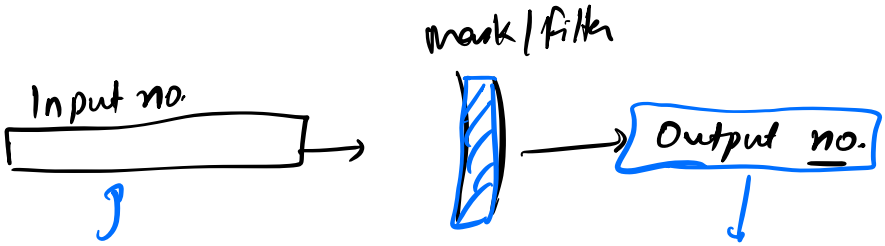
$$a >> b = \frac{a}{2^b}$$

Break till 10:07

* Bit Masking *



Bit



Check if a no. is even or odd.

10011100 0 0 11 10 & 00000.... 1

Input Mask

=

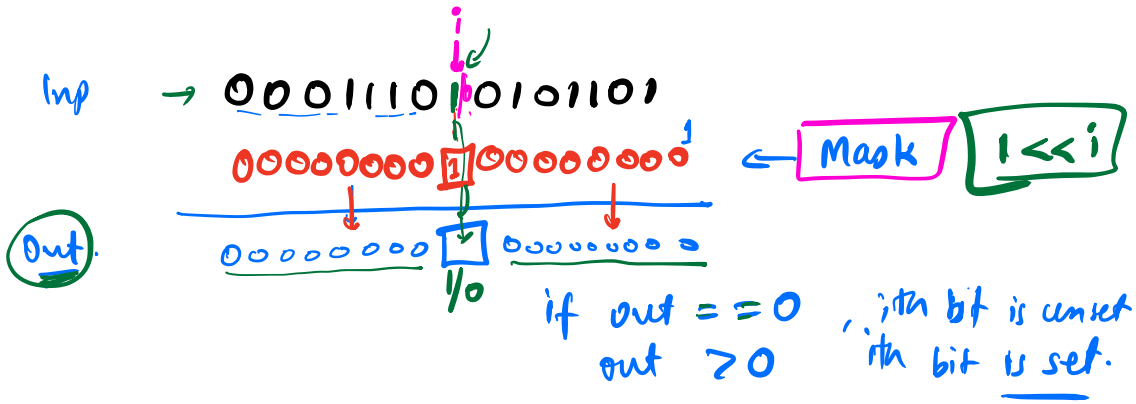
0000000.... 1/0

```
if (n & 1 == 1)
{
    //odd.
}
```

Q. 1. Given a no. N, check if ith bit is set in the no. or not.

ith → from LSB (0-indexed)

Set bit → 1
Unset bit → 0

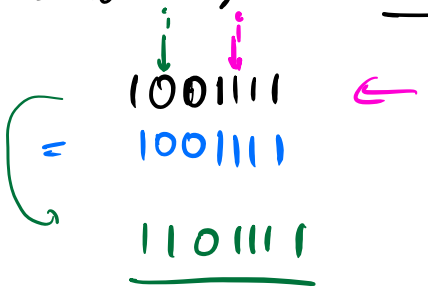


if out == 0, ith bit is unset
out > 0 ith bit is set.

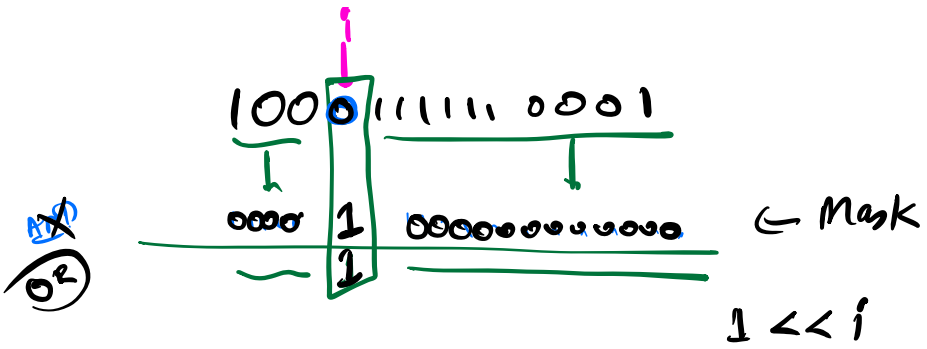
(**) num & (1<<i) > 0 → set
== 0 → unset.

00...000 1 ← 1<<i

Q. Given a no. N, set the ith bit.



(6) ← set 4th bit



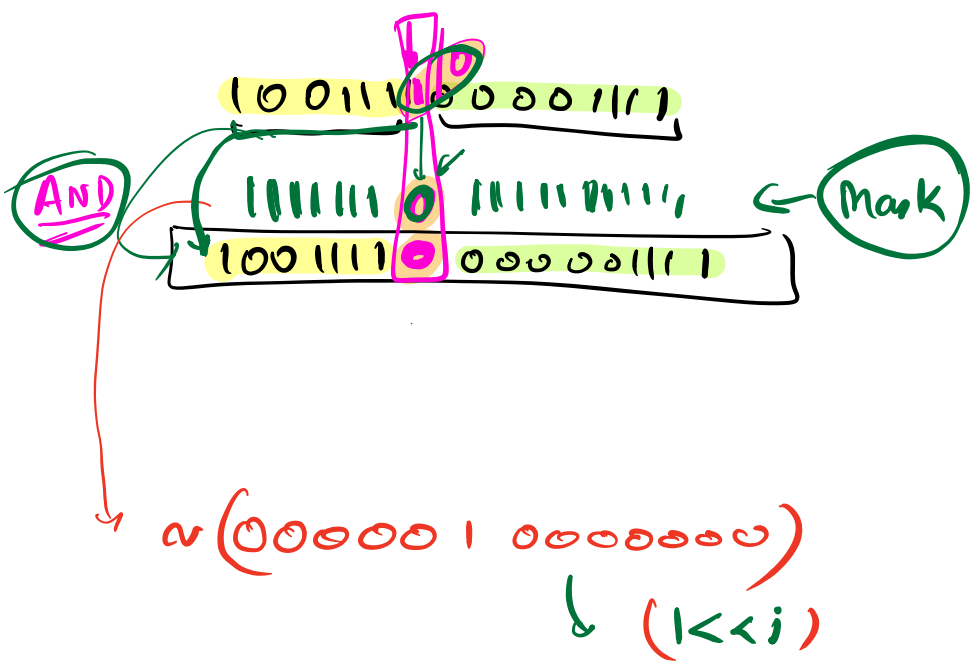
Ans : num | (1<<i)

Q. Given a no. N, clear the ith bit.

XOR

1 XOR 1 = 0
0 XOR 1 = 1

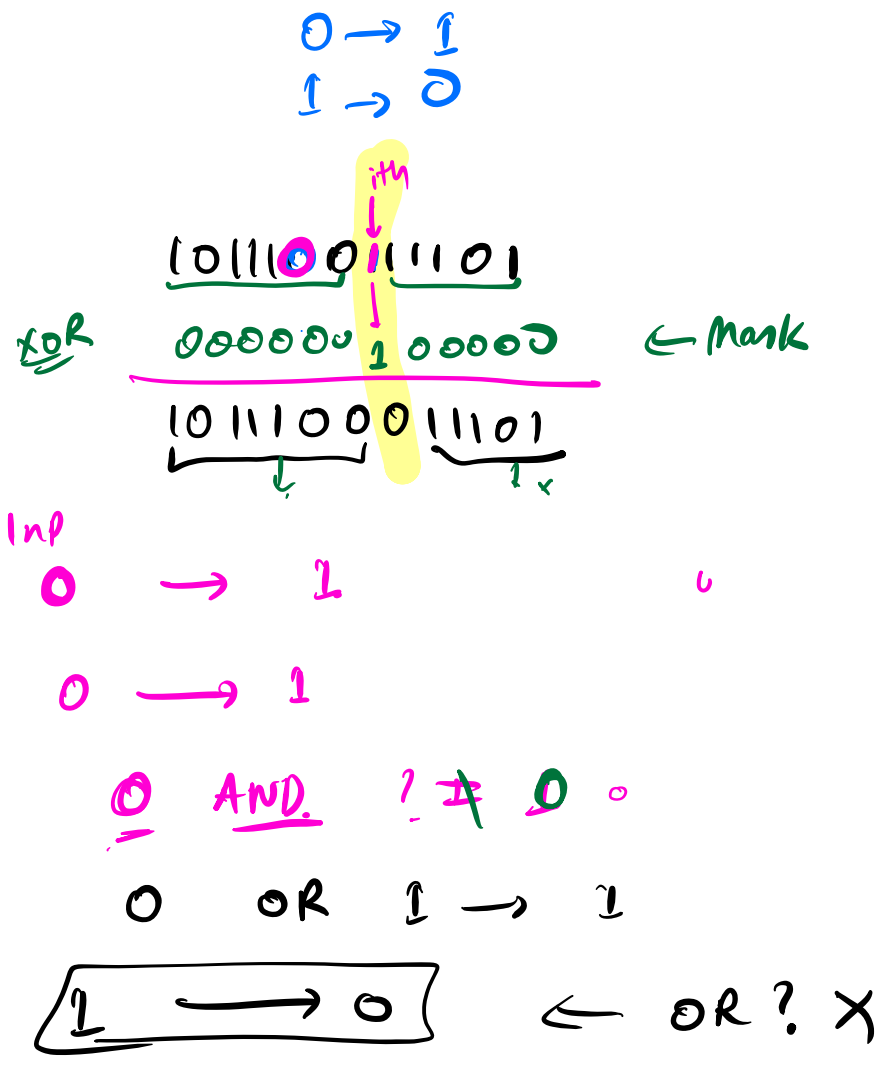
} X Not valid.



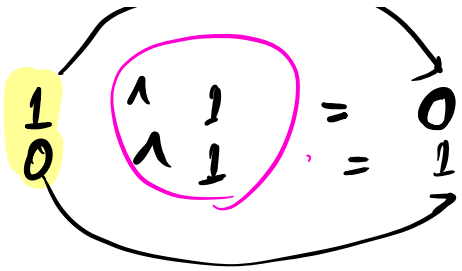
Ans = num & ~(1<<i)

Q. **Toggle**

Given a no. N, toggle the ith bit.



XOR



} gets toggled.

Any = num ^ (1<<i)

Recap.

num | (1<<i) = Set the bit

num & (1<<i) = Check whether the bit is set or not.

num ^ (1<<i) = Toggle the bit

num & ~(1<<i) = Clear the bit

Problem

① Check if given number has exactly 1 set bit.

(9) → 1001 → 2 set bits.

(15) → 1111 → (No)

Powers of 2 (2^0 is included) → have exactly 1 set bit.

Brute force.

```
cnt = 0
while (N > 0) {
    c = N % 2
    N = N/2
    if c == 1 :
        cnt += 1
}
if (cnt == 1)
    return True
else
    return False
```

→ TC: O(log N)

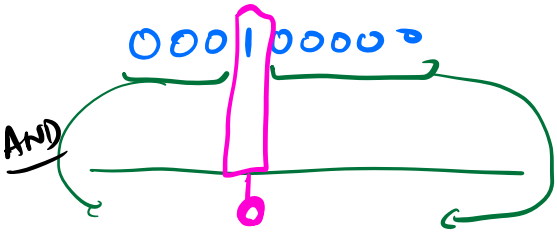
Idea: (**) Clean the rightmost set bit.
(hint)
and check if no. is 0 or not.
↓ exactly 1 set bit.
↓ >1 set bit

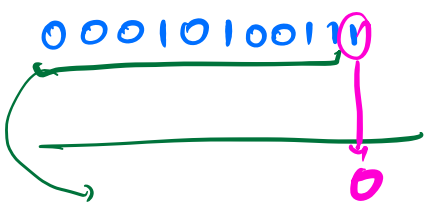
100110000
0
100110000 >0 ← More >1 than set bits.

000011000 0 ==0.

00 10100 00 0

Bitmask



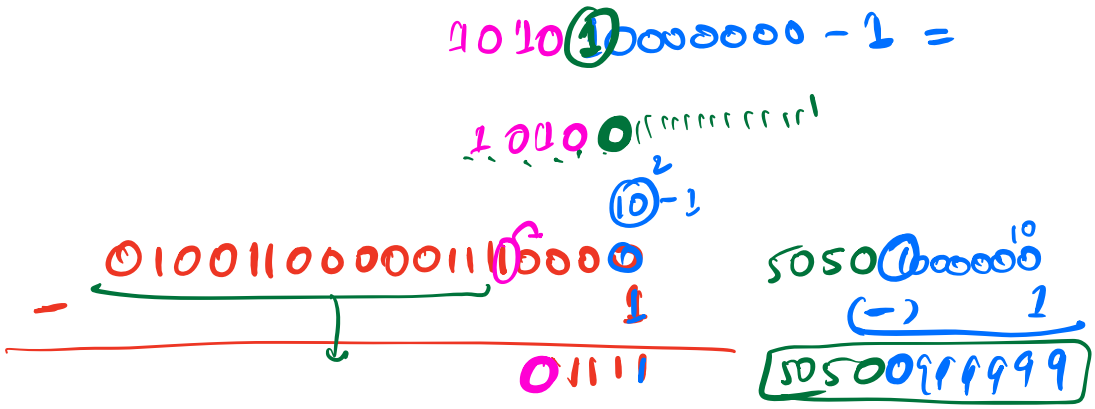


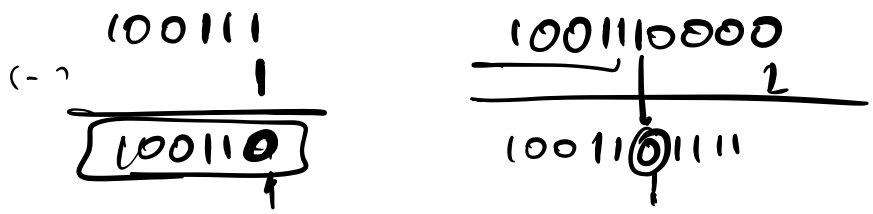
(*N-1*)

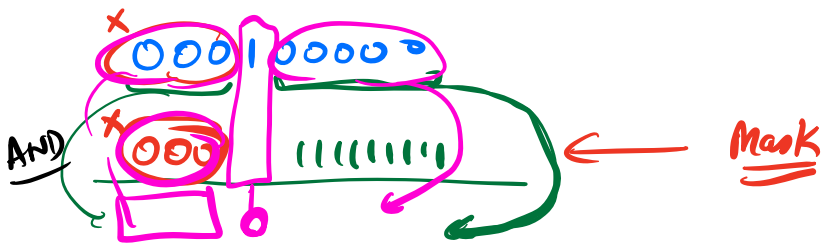
0100000 0 1
- 0 0 0000001
0 1111111

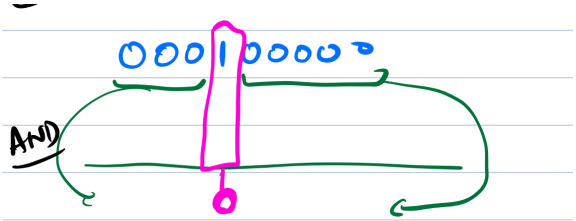
(*) 0-1 = 1 with 1 borrow

(*) 1 0 1111111 + 1 = 1000000





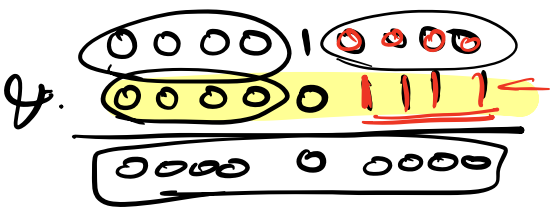

AND
Mask


AND

0011000111 → 0

N = 0001110000

N-1 = 0001011111 ← (N-1)



(N=0) 0 & -1 xx

Ans

if N>0 {
  if (N & N-1) > 0
    // More than 1 bit is set
  else
    // only 1
}

edge case. ← {
  if (Num == 0)
    // No bit is set
  else
    N
}

(xx) This is something you should remember.

* Count no. of set bits.

// Simply keep a count of how many times you are clearing the right most set bit.

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

TC : O(No. of set bit in N).

Brute force → O(log N)
O(no. of bit in N)

Initial score 0 ——————→ Final score A

24

+1 → $1
or x2 → free.

0 → 1 x2 x2 x2 x2 x 8 $

1$

9$

Hw

0 → 1 → 2 → 3 x2 x2 x2

1$ 1$

25

Bit Manipulation

$2 \rightarrow (4)_2$

— x — x —

$2^i \& N == 2^i$

```
  00001 0000
& 0
-----------
  00001 0000
```

TC of bitwise operation → O(1)

$L \wedge L = 0$

2, 3, 1, 0, 0, 1, 3, 4, 2, 2

XOR

$2 \wedge 3 \wedge 1 \wedge 0 \wedge 0 \wedge 1 \wedge 3 \wedge 4 \wedge 2$

XOR