

## 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

$$\begin{array}{r}
 1 = 00000\ldots \quad 1 \\
 44 = \overbrace{\begin{array}{cccccc} x & x & x & x & x & x \end{array}}^{\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow} \quad 0 \\
 \hline
 1 \wedge 44 = \overbrace{\begin{array}{cccccc} x & x & x & x & x & x \end{array}}^{\_ \_ \_ \_ \_ \_} \quad 1 = 45
 \end{array}$$

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

~~2 AND 3~~ - 0 & 1

✳️ ✗ 1 AND 46

$$\begin{array}{r}
 1 = 00000000\ldots \quad 1 \\
 46 = \overbrace{\begin{array}{cccccc} x & x & x & x & x & x \end{array}}^{\_ \_ \_ \_ \_ \_} \quad 0 \\
 \hline
 1 \wedge 46 = \overbrace{\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \end{array}}^{\_ \_ \_ \_ \_ \_} \quad 0 = 0
 \end{array}$$

✳️ ✗ 1 OR 43

$$\begin{array}{r}
 1 = 0000000\ldots \quad 1 \\
 43 = \overbrace{\begin{array}{cccccc} x & x & x & x & x & x \end{array}}^{\_ \_ \_ \_ \_ \_} \quad 1 \\
 \hline
 1 \vee 43 = \overbrace{\begin{array}{cccccc} x & x & x & x & x & x \end{array}}^{\_ \_ \_ \_ \_ \_} \quad 1
 \end{array}$$

The logo consists of a stylized 'X' or 'K' shape enclosed within a circle.

# 1 XOR 57

$$\begin{array}{r}
 0000000\ldots 1 \\
 \times x x x x x \ldots 1 \\
 \hline
 x x x x x x \ldots \underline{1} \\
 \downarrow \\
 0
 \end{array}
 \rightarrow \underline{\underline{56}}$$

Increase by 1

1

$28 + 1 = 29$

$$1 \text{ OR } 27 = 27$$

2

$$\begin{array}{r} \text{---. } 00001 \\ \underline{- - - - -} \\ \times \cancel{\times} \cancel{\times} \cancel{\times} \cancel{\times} \cancel{\times} \end{array} \quad \text{AND} \quad \begin{array}{r} 27 \\ \underline{\underline{=}} \\ 26/27 \end{array}$$

- ## \* Left shift operator.

A diagram illustrating a left shift operation on a binary number. On the left, the word "Binary" is written in a large, slanted font above a horizontal line. To the right of this line is the binary number "10010011". A vertical arrow points downwards from the middle of the binary number to a rectangular box below it. Inside the box, the binary number "10010011" is shown again, with the last digit, "1", enclosed in a small oval. To the right of the box, the symbol "=< 1" is written, indicating a left shift by one position.

00021010 << 1 | left shift.

A diagram showing the left shift operation (`<< 2`) on the binary number `1001`. The number is enclosed in a blue oval. A blue arrow points from the bottom of the oval down to the number `100100`, which is the result of shifting `1001` two places to the left. To the right of the result is an arrow pointing right and the text `→ 1001`, indicating the final value.

int32.

int32 (5)

$\leftarrow 0000000000\ 0000101$   
 $1010$

100000,- . . . , 101

\*\* 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  
↑

0 0 1 0     $\ll 2$   
↓

8 bit  
4 bit  
16 bit  
32 bit  
64 bit

10 0 0 0     $\otimes = 8 \checkmark$   
0 0 0 0 0 0 1 0     $\ll 2$   
"    ⑧

Why do we fill 0?

Because it is the definition

\*

$15 \ll 2$

$\leftarrow \frac{15}{2}$

0 0 0 0 0 0 0 1 1 1 1 0 = A

$$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

0 0 0 0 0 1 1 1 1 0 0  
↑  
 $3 \rightarrow 5 \quad 2 \rightarrow 4$

$$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2$$

$$\downarrow A \times 2^2$$

\* General relation

$$a \ll b = a \times 2 \times 2 \times 2 \times 2 \times \dots \times 2 \times 2^b$$

*b times*

$a = 100000111000$

No. of time we are shifting

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

Q.

$$1011 \text{ AND } ((\sim 0) \text{ left shift.}) \approx 0000$$

1st.      ↓

$$\begin{array}{r} 1111 \\ \underline{\times 1011} \\ + 1000 \end{array}$$

$\sim 0$

$\sim (00000\dots0)$   
 $11111111\dots111)$

$$\sim 0 \neq 1$$

$$\sim 0 = 11111\dots111 \sim \text{~}$$

$$\begin{array}{r} 1111\dots111 \\ \underline{\times 1100} \\ + 1100 \end{array} \quad \ll 2 \quad \text{~}$$

$$\sim (0000000\dots)$$

$$\begin{array}{r}
 \text{AND} \\
 \cancel{\underline{111111\ldots\ldots}} \quad \cancel{\underline{1100}} \\
 \underline{0000\ldots000} \quad \underline{1011} \\
 \hline
 \underline{1000} = 8
 \end{array}$$

32 bits.

0 is int  
00000000000000000000000000000000

(Quiz)

$N \& (1 << i)$

1101 & (1 << 3)

$$\begin{array}{r}
 0001 \\
 \downarrow \\
 1000 \\
 \underline{1101} \\
 \rightarrow \boxed{1000} = 8
 \end{array}$$

\* Right shift:

--- 1011  $\geq 2$   
 lost

00 ----- 10

$$\begin{array}{rcl}
 \underline{1011} & \gg 3 & = 0001 = 1
 \end{array}$$

$$29 \gg 2 =$$

$$\cancel{11101} \gg 2 \quad 00111 \leftarrow \textcircled{7}$$

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

$$\begin{array}{rcl} 25 & \gg & 1 = 12 \\ 24 & \gg & 1 = 12 \end{array}$$

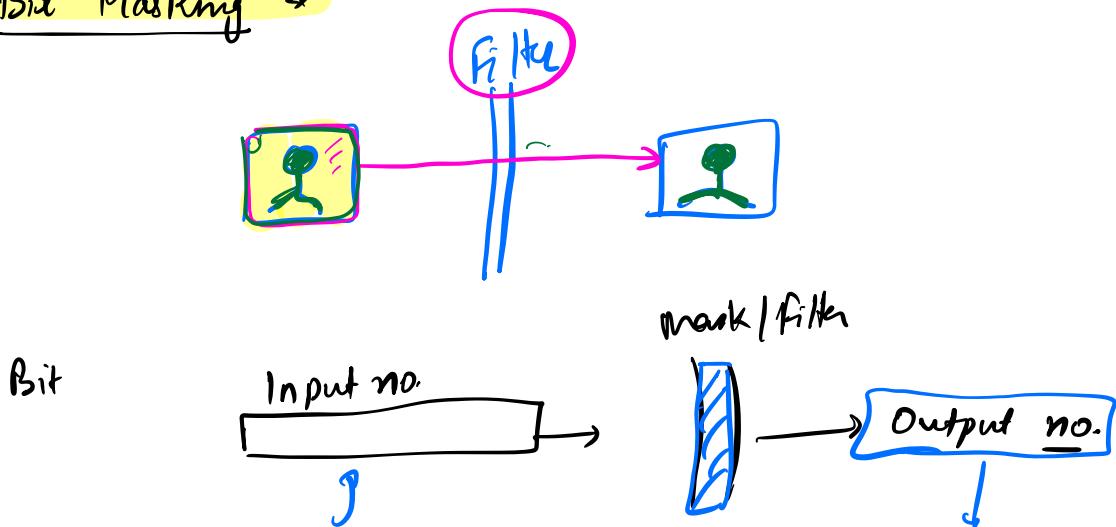
25      11001 <sup>get lost.</sup>  
24      11000

General relation

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

Break till 10:07

## \* Bit Masking \*



Check if a no. is even or odd.

100111000011r10 & 00000...1

Input

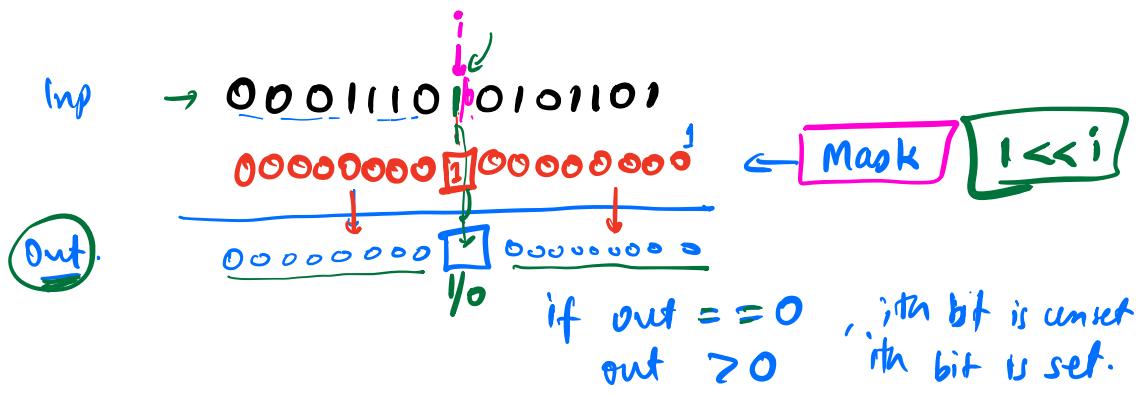
11  
0000000...1/0

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

Q. 1. Given a no.  $N$ , check if  $i^{th}$  bit is set in the no. or not.

from LSB  
(0-indexed)

Set bit  $\rightarrow 1$   
Unset bit  $\rightarrow 0$



\*\* number  $(1 \ll i)$   $> 0$        $= 0$   
 ↓  
 set      ↓  
 unset.

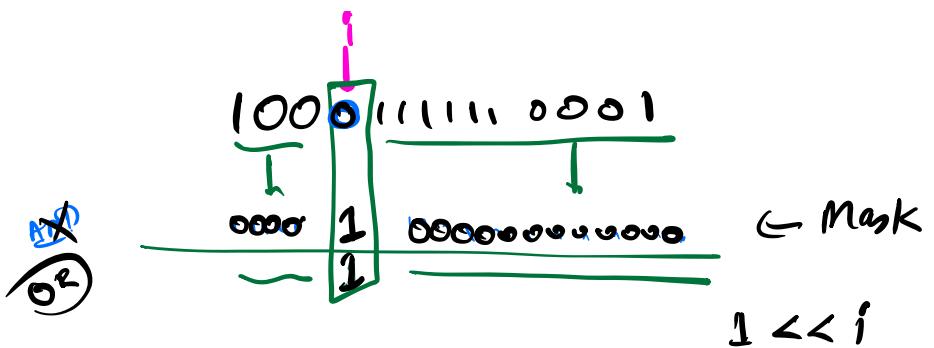
00.....000 1       $1 \ll i$

Q. Given a no.  $N$ , set the  $i^{th}$  bit

$=$

1001111	$\leftarrow$
1001111	
1101111	

$\leftarrow$  4th bit      set

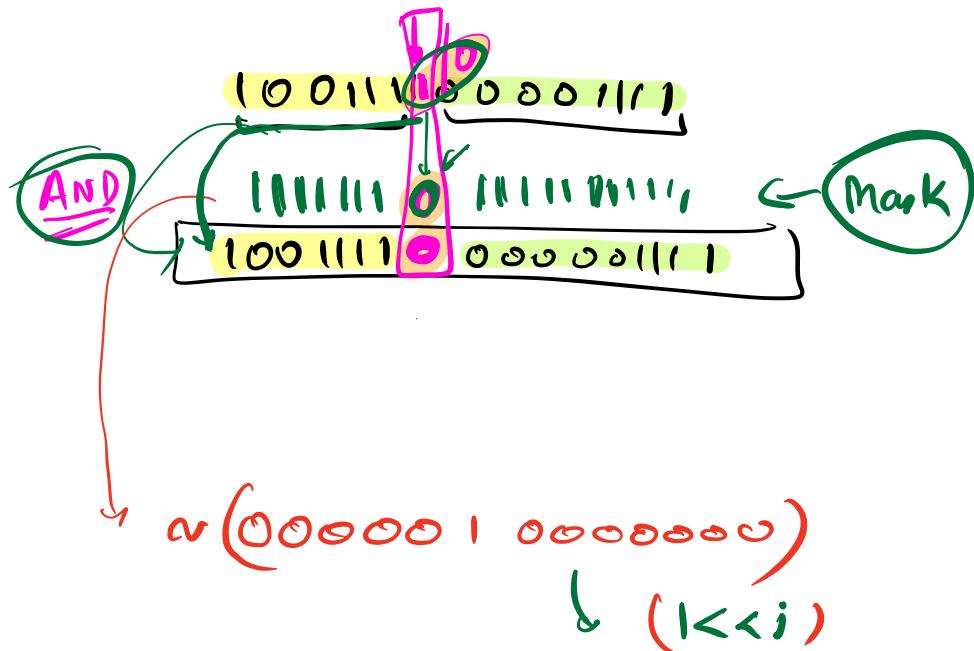


Ans :  $\text{num} | (1 << i)$

Q. Given a no. N, clear the  $i$ th bit.

~~XOR~~

$$\begin{array}{rcl} 1 & \xrightarrow{\text{XOR}} & 1 = 0 \\ 0 & \xrightarrow{\text{XOR}} & 1 = 1 \end{array} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Not valid.}$$



$$\text{Ans} = \text{num} \& \sim(1 \ll i)$$

Q.

Toggle.

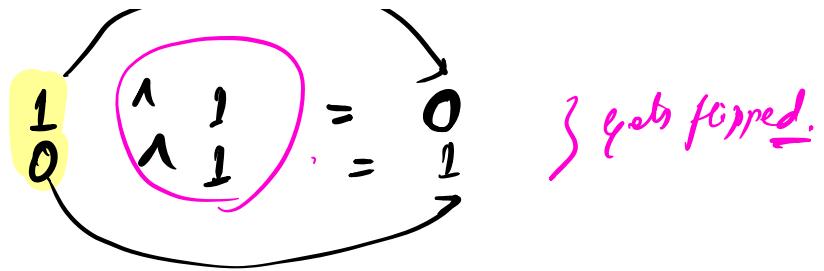
Given a no. N, toggle the  $i$ th bit.

$$\begin{array}{c}
 \begin{matrix}
 0 & \rightarrow & 1 \\
 1 & \rightarrow & 0
 \end{matrix} \\
 \text{XOR} \\
 \begin{array}{r}
 \text{Inp} \\
 \begin{array}{r}
 0 \rightarrow 1 \\
 0 \rightarrow 1
 \end{array}
 \end{array}
 \quad
 \begin{array}{c}
 \text{ith} \\
 \downarrow \\
 \begin{array}{r}
 10110011101 \\
 \text{---} \\
 00000010000 \\
 \text{---} \\
 101100011101
 \end{array}
 \end{array}
 \quad
 \leftarrow \text{Mask}
 \end{array}$$

0 AND ? ≠ 0

0 OR ? X

XOR



$$\text{Any} = \text{num} \wedge (1 \ll i)$$

Recap.

 num | ( $1 \ll i$ ) = Set the bit

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

 num ^ ( $1 \ll i$ ) = Toggle the bit

 num & ~( $1 \ll i$ ) = Clear the bit

## Problem

① Check if given number has exactly  
1 set bit.

⑨ → 1001 → 2 set bits.

⑯ → 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)

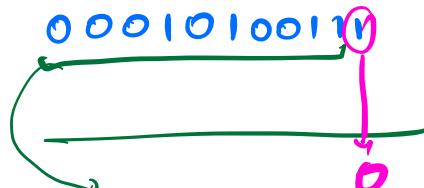
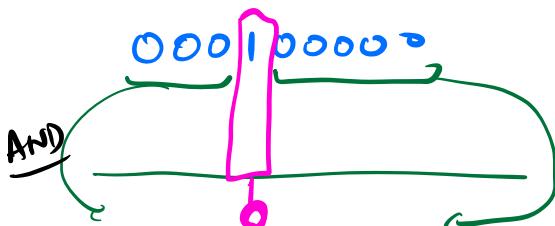
Idea: ~~\* \* Clear the rightmost set bit.~~  
 Hint and check if no. is 0 or not,  
 exactly 1 set bit.  
 $\geq 1$  set bit

$$\begin{array}{r} 10011\cancel{0}0000 \\ \underline{\quad 0} \\ 100110000 \end{array} > 0. \quad \hookrightarrow \text{More than } 1 \text{ set bit.}$$

$$0000\underline{1}0000 = 0.$$

$$00101\underline{0}0000$$

### Bitmask

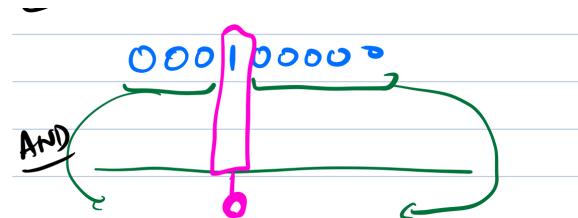
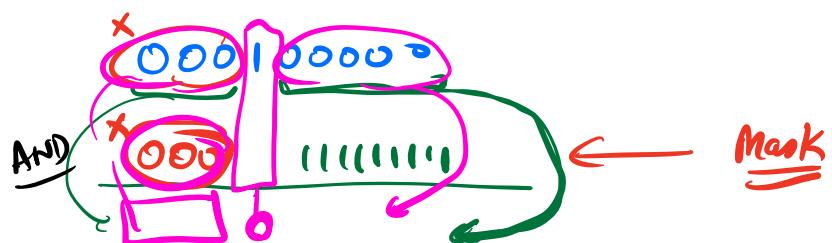


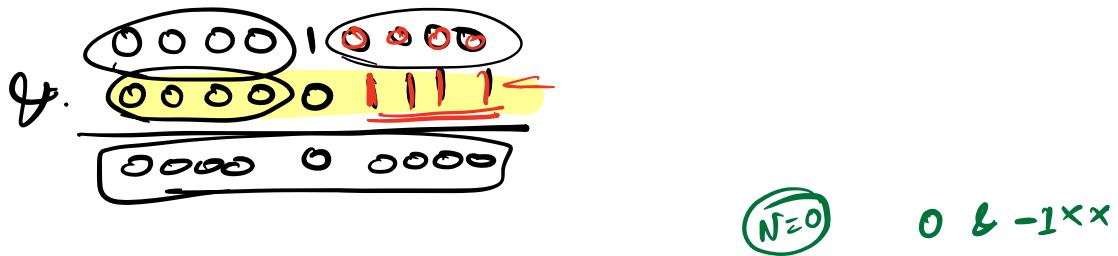
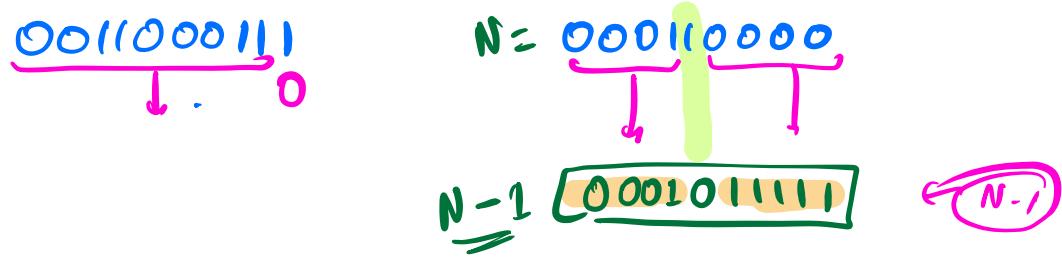
$$\begin{array}{r} + N-1 \\ \hline 010000001 \\ - 002000001 \\ \hline \dots 01111111 \end{array}$$

$$\oplus 0-1 = 1 \text{ with 1 borrow}$$

$$\oplus 0\underbrace{1111111}_{+1} = 10000000$$

$$\begin{array}{r}
 100111 \\
 | \\
 \hline
 100110 \\
 \end{array}
 \quad
 \begin{array}{r}
 1001110000 \\
 \hline
 | \\
 \hline
 1001101111
 \end{array}$$





Ans

~~if  $N \neq 0$~~  { if  $(N \& N-1) \geq 0$   
          // More than 1 bit is set  
        do  
          // only 1

edge case:  $\left\{ \begin{array}{l} \text{if } (\text{Num} == 0) \\ \quad // \text{No bit is set} \\ \text{else} \\ \quad N \end{array} \right.$

\*  
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.

```
int = 0
while(n > 0) {
    N = N & N - 1
    cnt += 1
}
```

return ans;

(TC)

:  $O(\text{No. of set bits in } N)$ .

Brute force  $\rightarrow O(\log N)$   
 $O(\text{no. of bits in } n)$

Initial score

0

Final score

A

24

+1 → \$1  
 or ×2 → free.

0 → 1 × 2 × 2 × 2 × 2 × 8\$

↓  
 1\$

9\$

HW

0 → 1 → 2 → 3 × 2 × 2 × 2  
 ↓      ↓  
 1\$      1\$

2\$

## Bit Manipulation

2 → ④

— x — x —

$$2^i \wedge N = 2^i$$

TC of bitwise operation  $\rightarrow$  O(1)



xof