Bit Manipulation

Part - 2

April 13, 2022

AGENDA:

- Some more Bitwise operators
  (Left shift and right shift)
- Bit masking concept
  & some problems.

XOR. $1 \wedge 0 = 1$, $1 \wedge 1 = 0$

OR $1 | 0 = 1$

AND. $1 \& 0 = 0$

NOT $!0 = 1$

* Left Shift.

8 bits. 1

00000101 ←

00001010 ←

Left shift operator. "<<"

59

0059

0590

$(abcd)_k \rightarrow x$

$(abcd0)_x \rightarrow (kx)_k$

$(0110)_2 << 1$ → 6

$= 1100$ ← 12.

4 bits. 3 2 1 0

0 1 1 0

1 1 0 0 ← 1

$= 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

$= 1\times2^3 + 1\times2^2 + 0\times2^0 + \_\_$ (0)

Multiply by 2.

* Assume. that MSB is always 0.

[In Python, don't worry about overflow).

int → 32 bits.

(1) num * 2.

1000 ← 8

(8*2) ←

001011 << 1

↓

010110 Gets multiplied by 2.

000101 << 2

↓

010100 Multiplied by 4!

Qu'2.

(2)

0010 << 2.

↓

1000

(8)

2×2^2.

$= 8$.

Quiz.

a = 15

print(a << 2)

(15)10

= 00011111

<< 2.

= 111100 → 60.

8+ 1111

(7) = (9) + 3

→ 2+1

15 x 2 x 2.

= 60. ✓

(10)

8 + 4 + 2 +

1010

10.

8 + 4 + 2 + 1

→ 1 0 1 0

13

8 + 4 + 2 + 1

→ 1 1 0 1

General relation

$a << b = a * \underbrace{2 \times 2 \times 2 \times \ldots \times 2}_{b \text{ times.}} = a \times 2^b$

15 << 2 = 15 x 2 x 2

13 << 4 = 13 x 2 x 2 x 2 x 2

Quiz. 1101 & (1 << 3) → 0000001

= 1101 & 1000 →

1101
& 1000
1000

= 1000

= (8). ✓

Right shift operator

`>>`

gets lost! (LSB is lost)

0 1 0 1 >> 1

= 0 0 1 0

2

5

5//2 = 2.

4

0 1 0 0 >> 1

0 1 0 1 >> 1

5

5//2 = 2
4//2 = 2.

Quiz.

11

1 0 1 1 >> 3

$11 // 2^3$

= 11//8

= 1

0 . 1 0 1

0 0 1 0

0 0 0 1 → 1

0. $(91001)_{10}$ // 2. //2 //2 . . . .

1 // 2 = 0. ✓

General relation

$$a >> b = a // 2^b$$

$$\times \left[\frac{a}{2^b} = a * 2^{1/b} \right]$$

Quiz.

$29 >> 2$

$= 29/(2 \times 2)$

$= 29/4 = 7$

Quiz.

which of these is a power of 2?

$2 << n$

$1 << n+1$

$2 << n+1$

$1 << n.$

10
100 ← 4
1000 ← 16

10

10 000000

2^n.

0000001 ← 2^0.
10
100
1000
100000.

Quiz. Which is 5 power n ?

$5 * (1 << n)$ ✓ $5 * 2$, $5*4$, $5*8$, $5*16$

$5 << n$ → ✗ $5 * 2^n$

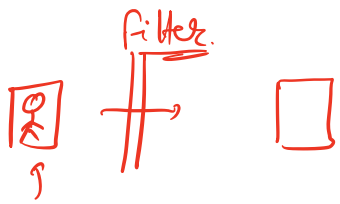
$5 << (n-1)$ ✗ $5 * 2^{n-1}$

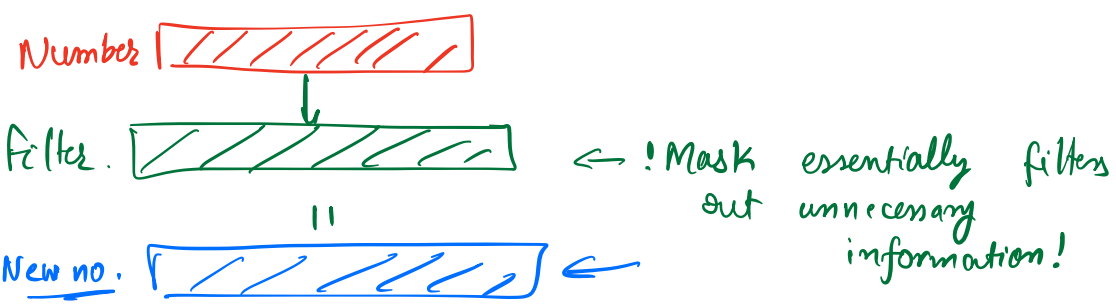
None.

5^n ✗

$2^{n-1} / 2^{n-2}$

Bit masking





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

(from LSB side, 0-indexed)

16 → 10000, 11111 → 15 → 1 1 1 1 → 4 bits (set)

9 → 1001 → 2 set bits.

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

(0-indexed)
4th bit is set or not ?
Set!

Brute force → Convert into binary, Check ith bit! → O(log N)
N

Bit masking

1 0 1 1 0 0 1 1
AND 0 0 0 1 0 0 0 0 ⟸ Mask.
0 0 0 1 0 0 0 0

will be either 0 or 1, depending on whatever it was in the original no.

> 0 : ith bit is set.
= 0 : ith bit is unset.

Num: 1 1 0 0 1 1 ⟵ (N) ith bit
AND
Mask 0 1 0 0 0 0 ⟵ 1<<i
ans = 0 1 0 0 0 0
i ≥ 0.

1 0 0 0 1 1
AND
0 1 0 0 0 0 ⟵ 1<<i
ans. 0 0 0 0 0 0

if ans > 0 : ith bit is set.
if ans = 0 : ith bit is unset.

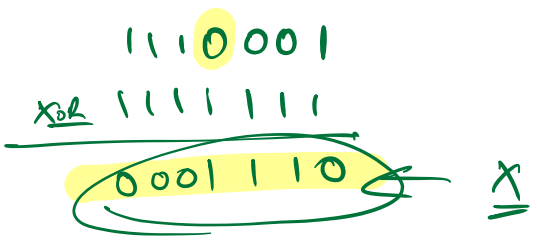
Pseudo-code.

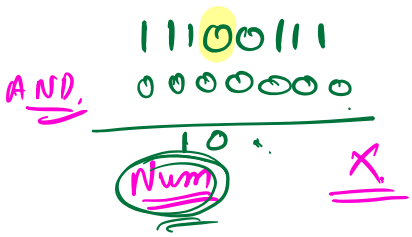
```
// Return true if ith bit is set.
def check_ith_bit (num, index):
    bitmask = 1<<index
    val = num & bitmask
    return val>0.
```

num → Decimal no

TC: O(1)

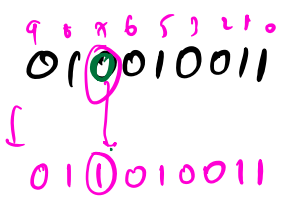
// ← if val > 0 return True else return False





Q.2.

Given a no. N, set the ith bit.



i = 7.

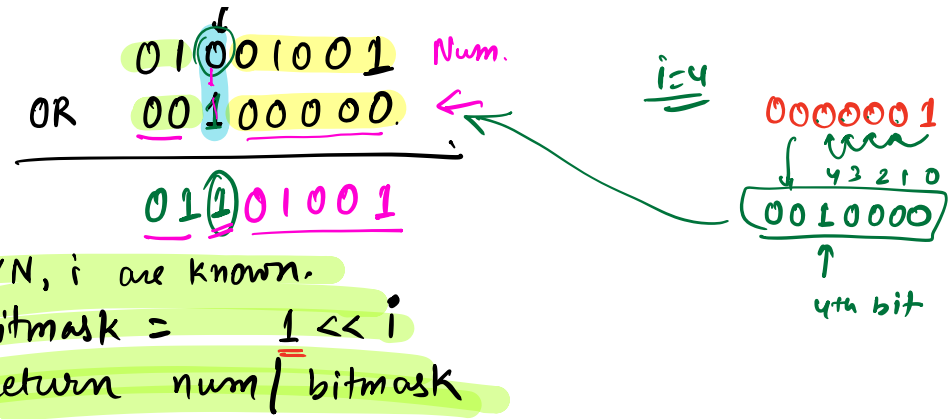
(Don't do anything if it is already set.)

Num = 5 , i = 1

101
111

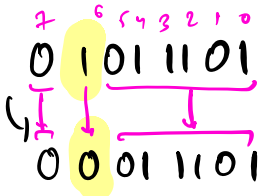
Ans = 7.

ith bit



Break till 10:30!

Q.3. Given a no. N, clear the ith bit.



(Ignore if ith bit is already 0).

Original

1 & 0 = 0 ✗ Not conserved

1 & 1 = 1 ✓

0 & 1 = 1 ✓ Con

1 | 0 = 1

0 | 1 = 1

✗

AND & conserve =

Num (1|0). AND 1 = 1|0

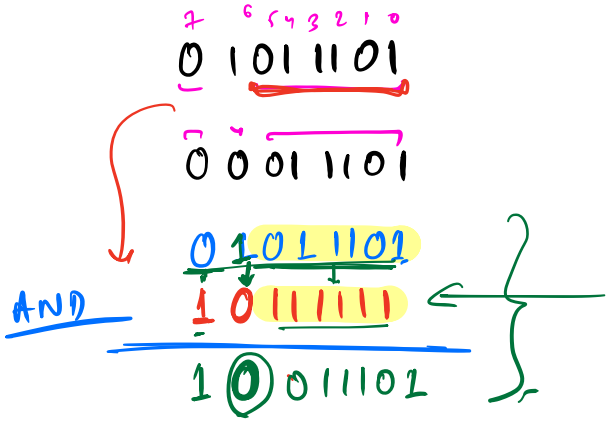
1 & 1 = 1

1 & 0 = 0

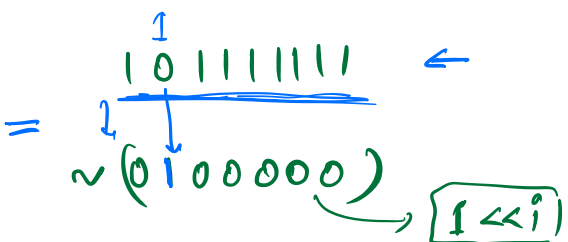
(i & 1 = 1)

Num (1|0) OR 0 = 1|0

1 | 0 = 1
0 | 0 = 0

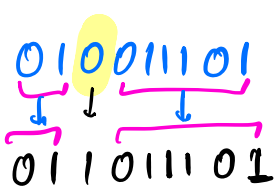


How do you come up with this mask ?



bitmask = ~(1<<i)
return num & bitmask.

Q.4. Given a no. N, toggle the ith bit.



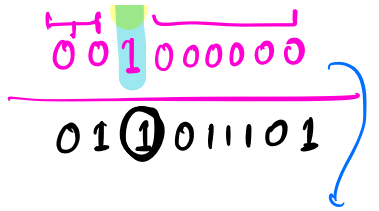
4

0 1 0 0 1 1 1 0 1

0/1 → 0. AND.

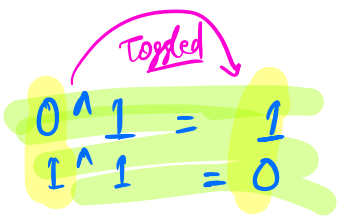
0/1 → 1 OR

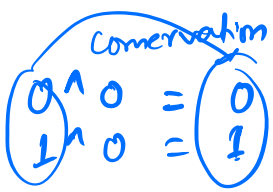
{ 0 → 1, 1 → 0. } NOT. XOR



(0/1) ^ 1 = (1/0)

toggled.





x Toggle.

Pseudo-code.

```
// num, i are known
bitmask = 1<<i
return num ^ bitmask.
```

Recap.

1. Check if ith bit is set. → AND (1<<i)
2. Set ith bit → OR (1<<i)
3. Clear ith bit → AND (~1<<i)
4. Toggle ith bit. → XOR (1<<i)

Problems.

1. Check if given no. N has exactly 1 set bit.

Inp = 15
Out = No

Inp = 16
Out = True.

Inp = 8
Out = True

Inp = 7
Out = No.

Inp = (10)
Out = False.

10
100
1000
100000

→ powers of 2!

Brute force :

Convert to binary
count no. of set bits.
Check if count > 1 → False
= 1 → True.

↓
O(log N)

Bitmasking :

00100000

00101000010

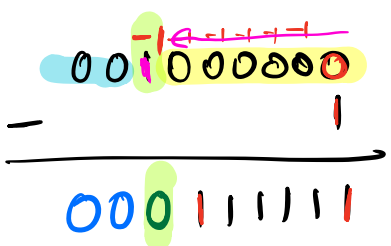
* If I clear right most set bit, and check if resulting no. is = 0 ⇒ It has exactly 1 set bit.
Otherwise ⇒ It has more than set bits.

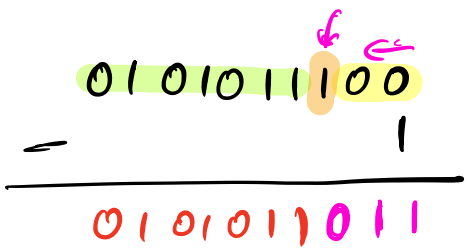
* Question boils down to:

Clear the right most set bit.

```
X   = 00100        X   = 10100
X-1 = 00011        X-1 = 10011
```

```
X   = 00101
X-1 = 00100
```



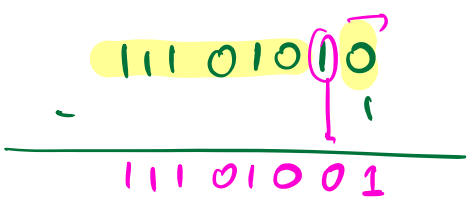


Observations:

1. All bits to the right of Least significant set bit, are set to 1.
   (right most set bit)

2. Least significant set bit is cleared.
   (right most set bit)

3. All bits to the left of Least significant set bit is conserved.



N = 0 1 0 1 0 0 1 0 0 0 0

N-1 = 0 1 0 1 0 0 1 0 1 1 1 ← Mask to clear right most set bit.

N & N-1 = 0 1 0 1 0 0 1 0 0 0 0 } clearing right most set bit.

if (N & N-1) > 0 :
    return False
else
    return True.

Q. 0 0 0 1 0 0 0 0 0 — N
− 1
0 0 0 0 1 1 1 1 1 → N-1
AND. (
||
0

N & N-1 → right most set bit is cleared.
N & N-1 = 0 if there was only one set bit.

0 0 1 0 1 0 0 0 N
− 1
0 0 1 0 0 1 1 1 N-1

N & N-1 → right most set bit is cleared.

N & N-1 > 0 → More than 1 set bits.!

Q.2. Given a no. N, count no. of set bits.

Idea:

Clear the rightmost set bit as many times as possible.

N = 10100111000

↓ N & N-1

N & N-1 = 10100110000

10100100000

↓

Pseudo-code.

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

N & N-1 → N

N = 2986

No. of iterations =

1 to 2986

↓

✗ O(N)

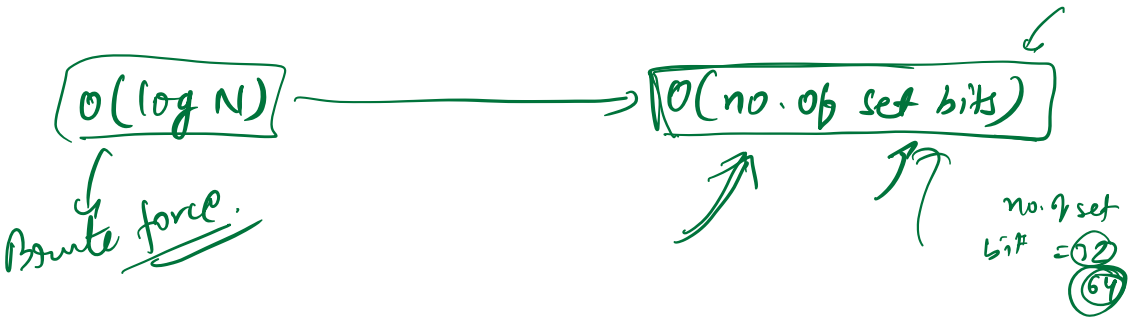
10 bits. 1 0 1 0 0 1 1 0 0 0

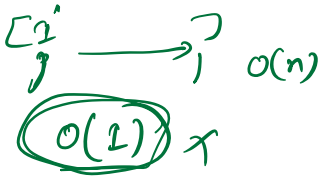
→ 1 0 1 0 0 1 0 0 0 0

→ 1 0 1 0 0 0 0 0 0 0

→ 1 0 0 0 0 0 0 0 0 0

O(no. of set bits)





Q. Game.

Initial score = 0

At every chance,

1) either you double the score → Free
2) you increase the score by 1. → 1$

what is the min. dollars you need to spend to reach from 0 to N?

↓

final score exactly be N.

0 → 15

0 → 1 x 2 x 2

1$

0 → +1 1$

*2 Free.

0 → 16 — 1$

0 → 1 x 2 x 2 x 2 x 2 ..

1$

1<<i → O(1) time.

64 → 7
32 → 6
16 → 5
8 → 4
4 → 3

25 → $\log_2 25$

No. of bits present in a no. = log N

32 (64)

32 bits.

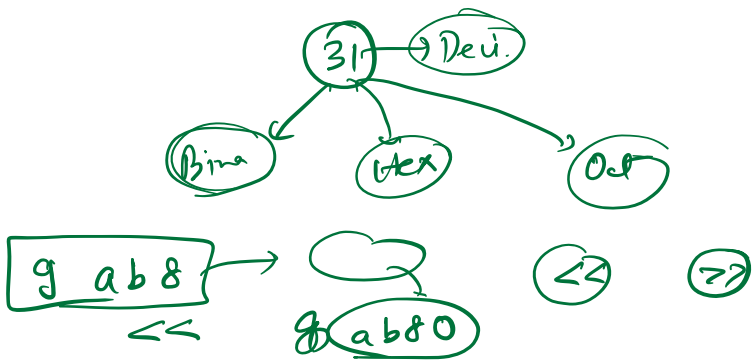
10011000

00000000
+1
00000001

Work on.

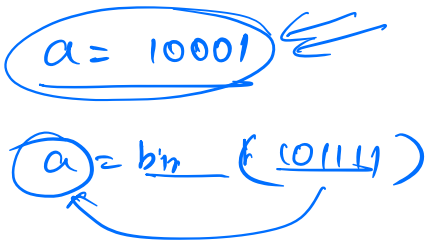
AND, OR, NOT, <<, >> → Decimal nos.

5<<1 5<<2



* Operators work on numbers (not number systems)

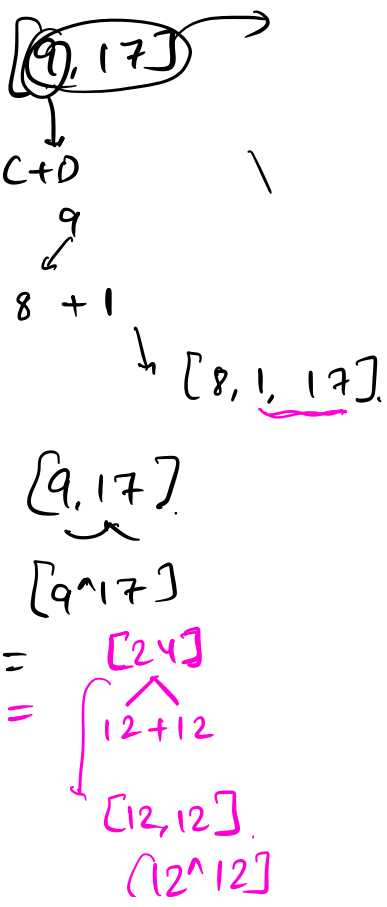
Bitwise operators work on the binary representation of that no.

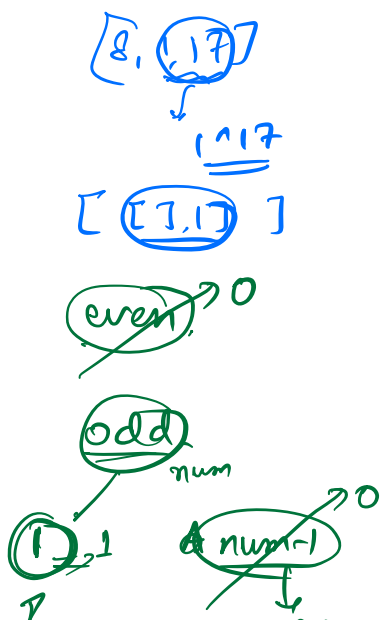


0 -

6

0 - 1





even