Bit Manipulation – 2

Agenda

1. Bit Shifting Operators <<, >>

2. Bit Masking Concept + Problems

3. Problems

1 Bit Shifting Operators

a. — Left Shift → multiply 2

<<

$x_1 = 1011$

$x_2 = 0111$

$x_1 \wedge x_2 = 1100$

$2^3\ 2^2\ 2^1\ \boxed{2^0}$

3 2 1 0

		Decimal
	$0000\boxed{1}$	1
0001 << 1	$000\boxed{1}\boxed{0}$	2
0010 << 1	$00\boxed{1}0\boxed{0}$	4
0100 << 1	1000	8

LSB set to zero.

$0001 << \boxed{3} = 1000$

---

8 bits = 1 byte

Decimal

128 | 64 | 32 | 16 | 8 | 4 | 2 | 1

0000 1001 — 9 — 1 + 8

LS

1 — $\boxed{0001\ 0010}$ — 18 — 2 + 16 = 9 * 2

2 — 0010 0100 — 36 — = 9 * 2 * 2 = $9 * 2^2$

3 — 72 — = 9 * 2 * 2 * 2 = $9 * 2^3$

$\boxed{4}$ — 144 — = = $9 * 2^4$

decimal $\qquad$ decimal value

$$a << b \quad = \quad a * 2^b$$

TC : O(1)

---

int $\qquad$ 32 bits $\qquad 2^{32} - 1 \simeq 2 * 10^9$

Processor → instructions part of Hardware.

Quiz - 1

$$1 << 4$$
$$= 1 * 2^4$$
$$= 16.$$

Quiz - 2

$$30 << 2$$
$$= 30 * 2^2$$
$$= 120.$$

Quiz - 3 $\qquad$ 1011 & (~0 << 2) $\qquad$ ⇒ a & 0 = 0

00 0000 0000 → 0

~0 $\quad$ 00 1111 1111

<< 2

(~0)<<2 $\quad$ 11 1111 1100

& 1011 $\quad$ 00 0000 1011

1000 ⇒ 8

b. Right Shift >>

MSB LSB vanish

Decimal

0101 >> 1 5

0101 >> 1   0010 → 2 $5//2 = 2$

0101 >> 2   0001 → 1 $2//2 = 1$

128 64 32 16 8 4 2 1

0010 0010

0010 0010 >> 1   0001 0001

0010 0010 >> 2   0000 1000

$\frac{32}{2} + \frac{2}{2} = \frac{34}{2}$

$\frac{16}{2} + \frac{1}{2} = \frac{17}{2}$ $\frac{34}{2}$

$8 + 0 = 8.$ $\frac{\frac{34}{2}}{2} = \frac{34}{2^2}$

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

Quiz - 4

$16 >> 5 = \frac{16}{2^5} = \frac{16}{32} = \boxed{0}$

2. Bit Masking

5 4 3 2 1 0

1010 0011

LSB

a. Given a number N, $\boxed{\text{check}}$ if ith bit is set in the number or not.

0 & 1 = 0
1 & 1 = 1

Idea

N : 0101 1101

$i = 6$

& 0100 0000 ⟹ Bitmask

res : 0100 0000

a & 0 = 0

· filter relevant data
· get rid of junk.

0000 0001 << 6 = 0100 0000

Last $i = 0$

  1011
& 0001
  0001

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

```
def test_bit (number, index) :
    # Todo. Ret. True/ False.

    bitmask = (1 << index)
    res = number & bitmask
    return res > 0
```

6. Set Given Bit in number N (Update N)

$$\begin{array}{lcc} & 7\,6\,5\,4 & 3\,2\,1\,0 \\ N = & 0\,1\,0\,1 & 1\,1\,0\,1 \\ \Rightarrow & 0\,0\,1\,0 & 0\,0\,0\,0 \end{array} \longrightarrow (1 << 5)$$

set $i = 5$th bit in N.

$1 \to 1$

$0 \to 1$

$0 | 1 = 1$

$1 | 1 = 1$ } set

$1 | 0 = 1$

$0 | 0 = 0$

retains.

In the above,
$N \,\&\, (1 << 5) = 0$

$$\Downarrow$$

$$\boxed{N \,|\, (1 << 5)}$$

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

```
def set-bit(num, idx):

    bitmask = (1<<idx)

    # num should have
      idx bit set keeping
      all other bits same

    num = num | mask

    return num
```

Midway Recap

$a << b = a * 2^b$

① Shifting

a. $0010 << 2$

$1000 = 8$

$2 << 2 = 2 * 2^2$

$= 2^3 = 8.$

b. $1011 >> 3$

$0001 = 1$

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

$11 // 2^3 = 11 // 8 = 1$

② Bit Masking

- Check • $N \& (1 << i)$
- Set • $N | (1 << i)$

0001	0	1
0010	1	10
0100	2	100
1000	3	1000

$N = 1101$

$i = 3$

$1 << i = 1000$

$1\ \underline{\quad\quad}$
i zeros.

$$\begin{array}{r} 1101 \\ \& \ 1000 \\ \hline 1000 \end{array} \Rightarrow 8 > 0$$

Unset.

C. Clear ith Bit

i

7 6 5 4 3 2 1 0

N = 0101 1101

&

1011 1111 ← ~(1<<6)

1<<6

0001 1101

a & 0 = 0

0 & 1 = 0

1 & 1 = 1

retains

1 → 0
set → unset

unset → unset
0 → 0

If 1 then should become zero.

```
def clear_bit(num, idx):
    bitmask = ~(1<<idx)
    return num & bitmask
```

d. Toggle i^{th} Bit

1 → 0

0 → 1

mask bit is 1.

$1 \wedge 1 = 0$

$0 \wedge 1 = 1$

$0 \wedge 1 = 1$ XOR

$1 \wedge 1 = 0$

$i = 5$

```
          0101 1101
bitmask   0010 0000     1<<i
          ---------
          0111 1101
```

$i = 6$

```
          0101 1101
bitmask   0100 0000     1<<6
          ---------
          0001 1101
```

rekin.

$1 \wedge 0 = 1$

$0 \wedge 0 = 0$

```
def toggle_bit(num, idx):
        bitmask = 1<<idx
        return num ^ bitmask.
```

3. Problems

a. Single Set Bit

0001 0000 => YES

0011 0001 => NO.

Check if the given number has 1 set bit!
exactly

Brute Force

Count the set bits

$N = 15$

$\boxed{1111}$

$2^0 + 2^1 + 2^2 + 2^3 = 1+2+4+8 = 15.$

Decimal → Binary.

$15 \% 2 = 1$

$15/2 = 7$

$7 \% 2 = 1$

$7/2 = 3$

$3 \% 2 = 1$

$3/2 = 1$

$1 \% 2 = 1$

		rem
2	15	
2	7	1
2	3	1
2	1	1
	0	1

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

TC: $O(\log_2 N)$

	N = 16	N%2 rem
2	8	0
2	4	0
2	2	0
2	1	0
	0	$\boxed{1}$

10000

Cnt = 1

15 → × 1111

16 → ✓ 10000 $= 2^4$

4 100 $= 2^2$

5 101

6 110

7 111

$\cancel{\sqrt{8} = 2.9}$

8 1000 $= 2^3$

9 1001

Powers of 2 have a Single Set Bit

6	5	4	3	2	1	0	
0	0	0	$\boxed{1}$	0	0	0	$= 2^3$
0	$\boxed{1}$	0	0	0	0	0	$= 2^5$
0	0	1	1	0	0	0	$= \cancel{2^3 + 2^4}$

$1 << i$

$\sim(1 << i)$

Bitmask

least significant set bit

eliminate

borrow.

```
a       0100 0000
1       0000 0001
a - 1 = 0011 1111
```

```
  0000 1000
- 0000 0001
  0000 0111
```

$1 - 1 = 0$

```
  0101 0111
- 0000 0001
  0101 0110
```

```
  1110 1010
- 0000 0001
  1110 1001
```

maintained.

```
5 | 0101
1 | 0001
4 | 0100
```

```
6 | 110
1 | 001
5 | 101
```

```
  -1 -1 -1
   1 9 7 8
-    9 9 9
   0 9 7 9
```

LS set bit

N	0101	1000
	− 0000	0001
N−1	0101	0111

Unchanged

Set to 1

rightmost set bit.

LS set Bit is cleared.

Clear LS set Bit of N without affecting other bits.

Same

N	0101	1000
N−1	0101	0111
&		

$\boxed{a \& a = a}$

$a \& 0 = 0$

$N \& (N-1) =$ 0101 0000

We cleared the rightmost set bit.

Original prob Single set bit

N = 0100 0000

N−1 = 0011 1111

N & (N−1) = 0000 0000 ⟹ 0

$N \,\&\, (N-1) \Rightarrow$ clear the rightmost set bit. (LS)

$N = 1010\ 0000$

$N-1 = 1001\ 1111$

$N \& (N-1) = 1000\ 0000$

If > 1 set bits in N

$\Rightarrow \underline{N \,\&\, (N-1)} > 0$

Conclude : If 1 set bit

$\Rightarrow \boxed{N \,\&\, (N-1) = 0}$ ✓ except $N = 0$

Can we say if $N \& (N-1) = 0 \Rightarrow$ N is power of 2

Edge Case $N = 0$

⇓

×

$0 \,\&\, (0-1)$

$0 \,\&\, a = 0$

```
def single_set_bit(num):
    if num == 0 :
        return False

    return num & (num-1) == 0
```

$O(\log N)$

to

$O(1)$

Q) Can we also optimize count set bits using this logic?

$N = N \,\&\, (N-1) \Rightarrow$ clear the rightmost set bit

```
def cnt_set_bits(N):
    cnt = 0
    while(N>0):
        cnt += 1
        N = N & (N-1)
```

TC: O(num of set bits)

$N =$ 1010 0000

$N-1 =$ 1001 1111

$N \,\&\, (N-1) =$ 1000 0000

N'

N'-1   01111111

&   00000000

0110 1110

↓

01101100   1

↓

01101000   2

↓

01100000   3

↓

01000000   4

↓

→ stop. ← 0000 0000   5

$\Rightarrow$ Doubts

$$\overset{23}{1}\,000\;\overset{19}{0}000\;\overset{15}{0}\,0\,0\,0\;\overset{11}{0}000\;\overset{7}{0}000\;\overset{3}{0}\overset{2}{0}\overset{1}{0}\overset{0}{0}$$

$= 2^{23}$

$O(\log N) \longrightarrow$ 23 steps.

O(count set bits) $\longrightarrow$ 1 step. ✓