Lecture ÷ Bit Manipulation-2

Agenda

- Power of left shift
    - Check ith set bit ✓
    - Set ith bit ✓
    - Toggle ith bit
    - Unset ith bit
- Count set bit in N.
- Binary representation of -ve numbers.

Prerequisite

$a << n = a * 2^n.$

$1 << 3 = 1 * 2^3 = 8.$

$2 << 4 = 2 * 2^4 = 2 * 16 = 32.$

Set bit = if bit value is 1.

unset bit = if bit value is 0.

Qu1 check kth set bit.

ip : n = 45 $\left[\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{array}\right]$

k = 2 , 2nd bit of 45 is 1 — true

k = 4 , false.

k = 0 , true

Hint :

Observation 1

n = 45

k = 2. (1<<2)

$1 * 2^2$

4

	5	4	3	2	1	0
	1	0	1	1	0	1
&	0	0	0	1	0	0
	0	0	0	1	0	0

= 4 = 1<<2.

k = 4

1<<4

2^4

		5	4	3	2	1	0
45 ⇒		1	0	1	1	0	1
16 ⇒	&	0	1	0	0	0	0
		0	0	0	0	0	0

= 0

k = 3

1<<3

2^3

	5	4	3	2	1	0
45 ⇒	1	0	1	1	0	1
8 ⇒	0	0	1	0	0	0
	0	0	1	0	0	0

⇒ 2^3 = 1<<3.

8

N & (1<<k) — non-zero (1<<k), kth bit of n=1.

— 0, kth bit of n=0

```
boolean checkKthSetBit(int n, int k) {
    int val = n & (1<<k);
                    2^k.
    if ( val == 0) {
        return false;
    }
    return true;
}
```

TC: O(1).

SC: O(1)

a^n :- TC :- $\log_2 n$.

1<<(n) :- 2^n.

(0 - 31)

n ∈ [0, 31]

int:- 4 bytes = 32 bits

31 30 29 . . . . . . 2 1 0

Qu2 Set kth bit of a number. [make kth bit ==1]

n = 45.

$$\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & \cancel{0} & 1 \\ & & & & \uparrow & \end{array}$$

k = 1.

$$\begin{array}{cccccc} 1 & 0 & 1 & 1 & 1 & 1 \end{array} \Rightarrow 47.$$

k = 4

$$\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & \cancel{0} & 1 & 1 & 0 & 1 \\ & 1 & & & & \end{array} \Rightarrow 61.$$

k = 2

$$\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & \cancel{1} & 0 & 1 \\ & & & 1 & & \end{array} \Rightarrow 45.$$

Observation

n = 45

$$\begin{array}{c|cccccc} & 5 & 4 & 3 & 2 & 1 & 0 \\ & 1 & 0 & 1 & 1 & 0 & 1 \\ k = 4.\ (1 << 4) & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 16 & 1 & 1 & 1 & 1 & 0 & 1 \end{array} \Rightarrow 61.$$

$$\begin{array}{c|cccccc} & 5 & 4 & 3 & 2 & 1 & 0 \\ k = 1\ (1 << 1) & 1 & 0 & 1 & 1 & 0 & 1 \\ 2^1 = 2. & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline & 1 & 0 & 1 & 1 & 1 & 1 \end{array} \Rightarrow 47.$$

n. | (1 << k)

- kth bit = 0   change kth bit to 1.
- kth bit = 1   no change, kth bit will 1 itself.

```
int   setkthbit( int n, int k){

      return  n | (1<<k);

}
```

TC: O(1)

SC: O(1)

Q Toggle ith bit. of n. [$1 \to 0$, $0 \to 1$]

$$\begin{array}{lcccccccl} & 5 & 4 & 3 & 2 & 1 & 0 & & \\ n = 45. & 1 & 0 & 1 & 1 & 0 & 1 & & \\ k = 4. & 1 & 1 & 1 & 1 & 0 & 1 & = & 61. \\ k = 2 & 1 & 0 & 1 & 0 & 0 & 1 & = & 41 \\ & 5 & 4 & 3 & 2 & 1 & 0 & & \\ k = 3. & 1 & 0 & 0 & 1 & 0 & 1 & = & 37. \end{array}$$

$2^5 + 2^2 + 2^0$

$$\begin{array}{lccccccc} & & 5 & 4 & 3 & 2 & 1 & 0 \\ n = 45. & & 1 & 0 & 1 & 1 & 0 & 1 \\ k = 4\ (1<<4) & \wedge & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 16 & & 1 & 1 & 1 & 1 & 0 & 1 \\ \hline \end{array}$$

$$\begin{array}{lccccccc} & & 5 & 4 & 3 & 2 & 1 & 0 \\ k = 3\ (1<<3) & & 1 & 0 & 1 & 1 & 0 & 1 \\ & \wedge & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline & & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array}$$

$$\begin{array}{lccccccc} & & 5 & 4 & 3 & 2 & 1 & 0 \\ k = 0\ (1<<0) & & 1 & 0 & 1 & 1 & 0 & 1 \\ 2^0 = 1 & \wedge & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline & & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline \end{array}$$

$$n \wedge (1<<k) \begin{cases} k\text{th bit} = 0 & k\text{th bit changes to } 1 \\ k\text{th bit} = 1 & k\text{th bit changes to } 0. \end{cases}$$

```
int toggle(int n, int k) {
    return n^(1<<k);
}
```

TC: O(1)
SC: O(1)

Qu Unset the ith bit of a no.? [changing bit value to 0].

$$n = 45 \left[\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{array} \right]$$

k = 3 → 1 0 0 1 0 1 = 37.

k = 2 → 1 0 1 0 0 1 ⇒ 41.

k = 1 → 1 0 1 1 0 1 ⇒ 45.

Approach

```
if ith bit is set — [ checking ith bit)
    change the ith bit of 0 ] toggling
else —
    do nothing
```

```
int unset( int n, int k) {
    if ( checkkthsetBit(n, k)) {
        return n^(1<<k);
    }
    return n;
```

Break: 8:25 AM

Qu Count set bits of n.

n = 45 [0 0 - 0 1 0 1 1 0 1] ⇒ 4 . set bits.

int : 4 bytes = 32 bits.

int n = 45. 0 0 0 0 0 0 - - - - - - - _ _ _ _ 0 0 1 0 _ 1 1 0 1

31 30 29 - - - - - - - - - - - - - - - - - - 4 3 2 1 0

Idea: Traverse from 0 to 31 —

check the bit value is set or not?

```
int countSetBit (int n) {
    int cnt = 0;
    for( i=0; i<32; i++) {
        if( checkkthBitSet(n, i) ) {
            cnt += 1;
        }
    }
    return cnt;
}
```

TC: $O(32) \simeq O(1)$

SC: $O(1)$

Approach 2.

n = 45.

5 4 3 2 1 0

1 0 1 1 0 1

1 & 0 0 0 0 0 1

0 0 0 0 0 1 → if n & 1<<0 (1) is 1, it means set bit.

n = 45

5 4 3 2 1 0

1 0 1 1 0 1

n>>1

0 1 0 1 1 0

& 1 0 0 0 0 0 1

0 0 0 0 0 0 ⇒ 0, bit is unset.

n>>2 0 1 0 1 1 0 (n>>1)

0 0 1 0 1 1

&1 0 0 0 0 0 1

0 0 0 0 0 1 ⇒ 1, bit is set.

Idea.

calculate n = n>>i [right-most bit will always be ith bit]

n & 1 → 1 (count++)

→ 0 (do nothing)

```
int countSetBits (int n) {

    int cnt = 0;

    while ( n > 0 ) {

        if (n & 1 == 1) {

            cnt += 1;

        }

        n = n>>1;

    }

  return cnt;

}
```

TC: $\log_2(n)$ — max value = 32.

SC: O(1)

Negative numbers

$(-45)_{10} = (x)_2$.

int = 4 bytes = 32 bits.

$$\begin{array}{cccccccccc} \_ & \_ & \_ & \_ & - - - - - - & \_ & \_ & \_ & \_ & \_ \\ 31 & 30 & 29 & 28 & - - - - - - & 4 & 3 & 2 & 1 & 0 \end{array}$$

↳ signed bit $\left[\begin{array}{l} 1 \text{ — negative no} \\ 0 \text{ — positive no} \end{array}\right]$.

$$\begin{array}{cccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 31 & 30 & - - - - & & & & 3 & 2 & 1 & 0 \end{array} . = \text{negative no.}$$

$$-2^{31} \quad + \quad 2^3 + 2^1$$

$$\begin{array}{cccccccc} 0 & \_ & \_ & \_ & \_ & 1 & 0 & 1 \\ 31 & 30 & - - - - & & & 2 & 1 & 0 \end{array} = \text{positive no}$$

→ 31st bit = signed bit.

if 31st bit == 1. (-ve numbers)

== 0 (+ve no.)

+ve no = $\begin{array}{c} 0 \\ 31 \end{array}$ _ _ _ _

-ve no = $\begin{array}{c} 1 \\ 31 \end{array}$ _ _ _ _

Range of int: $\underbrace{-2^{31}}_{\text{Integer.MIN\_VALUE}}$ to $\underbrace{2^{31}-1}_{\text{Integer.Max\_value}}$.

Proof:

max: $\underset{31}{\underline{0}}\ \ \underset{30}{\underline{1}}\ \ \underset{29}{\underline{1}}\ \ \underset{-}{\underline{1}}\ \ \underset{-}{\underline{1}}\ \ \underset{-}{\underline{1}}\ \ \underset{-}{\underline{1}}\ \ \underset{-}{\underline{1}}\ \ \underset{-}{\underline{1}}\ \ \underset{-}{\underline{1}}\ \ \underset{3}{\underline{1}}\ \ \underset{2}{\underline{1}}\ \ \underset{1}{\underline{1}}\ \ \underset{0}{\underline{1}}$

$2^{30} + 2^{29} + 2^{28} \cdots\cdots 2^{3} + 2^{2} + 2^{1} + 2^{0}$

$2^{0} + 2^{1} + 2^{2} - - - - - 2^{28} + 2^{29} + 2^{30}.$

$a = 1$

$r = 2.$

$n = 31.\ [0-30]$

$$sum = \frac{a(r^{n}-1)}{r-1} = \frac{1(2^{31}-1)}{2-1} = 2^{31}-1$$

min: $\underset{31}{\underline{1}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underset{-}{\underline{0}}\ \ \underline{0}\ \ \underset{3}{\underline{0}}\ \ \underset{2}{\underline{0}}\ \ \underset{1}{\underline{0}}\ \ \underset{0}{\underline{0}}$

$-2^{31} + [0] = -2^{31}$

Representation

$(-45)_{10} = (x)_2$

1. find bit representation of same +ve no.
2. flip all bits
3. add 1 to it.

Ex $(-45)_{10} = (x)_2$

$$\begin{array}{llcccccccccccccc}
 & & 31 & 30 & 29 & 28 & & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
1. & 45 = & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
2. & \text{flip :-} & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
 & \text{Add} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
 & & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
\hline
\end{array}$$

$$-2^{31} + \underbrace{2^{30} + 2^{29} + 2^{28} \cdots 2^{6}}_{GP} + \underbrace{2^{4} + 2^{1} + 2^{0}} = -45$$

$a = 2^6$

$r = 2$

$n = [6-30] = 30-6+1 = 25$

$$-2^{31} + \frac{2^6(2^{25}-1)}{2-1} + 16 + 2 + 1$$

$$\cancel{-2^{31}} + \cancel{2^{31}} - 2^6 + 19$$

$$-64 + 19 = \boxed{-45} \text{ Ans.}$$

Tips and tricks

Qu Given arr[n], return sum of all array els.

$1 <= n <= 10^5$

$1 <= arr[i] <= 10^6$

long ~~int~~ sum(int [] arr) {

  long ~~int~~ tot = 0;

    for(int el: arr) {

      tot += el;

    }

    return tot;

}

arr: $[\overset{10^5-1}{10^6} \quad 10^6 \quad 10^6 \quad 10^6 \quad ----- \quad \overset{1}{10^6} \quad \overset{0}{10^6}]$

sum = $10^6 * 10^5 = 10^{11}$.

Qu $1 <= a <= 10^9$

$1 <= b <= 10^9$

a * b;

int ans = a * b [wrong] $10^9 * 10^9 = 10^{18}$

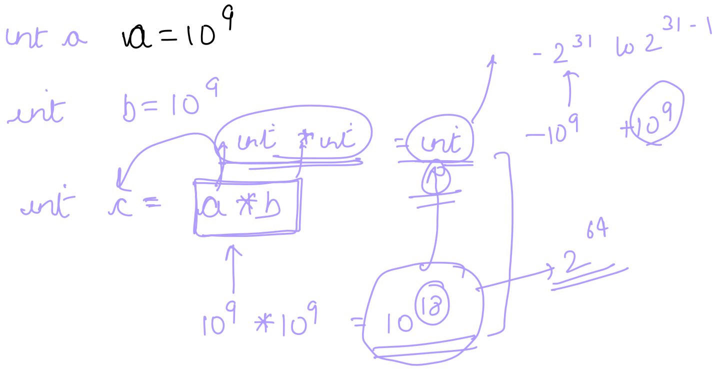
long ans = $\underbrace{a * b}$ [wrong]

int int = int (overflow)

long ans = (long) (a*b) [wrong]

long ans = $\underbrace{\text{long}(a)}_{\text{long}}$ * $\underbrace{b}_{\text{int}}$. [correct]

Thankyou ☺


int a a = 10^9
int b = 10^9
int c = a * b
int * int
= int
- 2^31 to 2^31 - 1
-10^9
+10^9
10^9 * 10^9 = 10^18
2^64