Bit Manipulation - 2

Jul 19, 2023

AGENDA

- Left shift and Right shift operator
- Bit masking and related problems
- Negative nos.
- Ranges and overflows

Revision

$11 \mid 1 = 11$

$a \& 1 = 1$ if a is odd

(for some no. a) 0 if a is even.

$a \mid a = a$

$a \wedge a = 0$

Left shift

<<

It left-shifts all the bits.

00101011 << 1 : Left shift all bits by 1.

01010110 [0 is put in LSB, all other bit are shifted to left by one].

00001010 << 3 : Left shift all bits by 3.

⇒ 01010000

Consider 8 bits

5 << 1 = 10

13 << 1 = 26

15 << 1 = 30

00000101 → 00001010

00001101 → 00011010

00001111 → 00011110

(MSB) This is always 0.

6 5 4 3 2 1 0
0 0 0 1 0 1 1

6 5 4 3 2 1 0
0 0 1 0 1 1 0

$= 0*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

$= 0*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$

* $a << 1 = 2 * a$

* $a << 2 = (a * 2) * 2$
  $a << 3 = ((a * 2) * 2) * 2$
  $a << 5 = a * 2^5$

* $a << m = a * 2^m$ → O(1) operation

Right shift

>>

00001011 >> 1

= 00000101

(All bits right shift by 1, LSB disappears).

5 >> 1
= 2

13 >> 1
= 6

12 >> 1
= 6

14 >> 1
= 7

101 >> 1
= 10

1101 >> 1
= 110

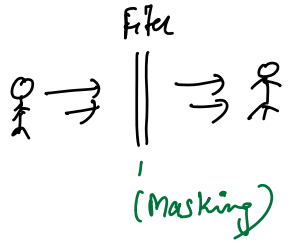
1100 >> 1
= 110

1110 >> 1
= 111

* $a >> 1 = a/2$ [Integer division]

* $a >> m = a/2^m$

Bit masking



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

↳ 0-indexed from the right.

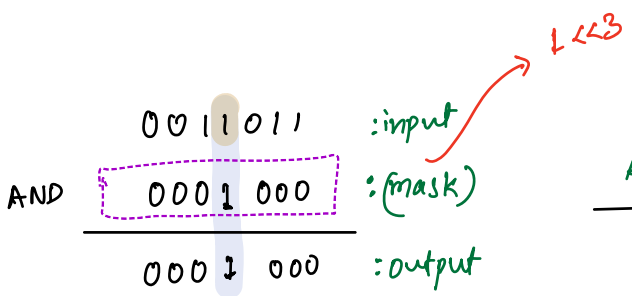
Set bit : 1

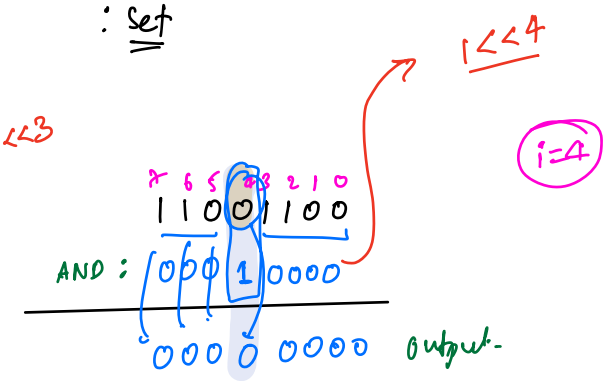
Unset bit : 0

e.g.. N = 27, i = 3

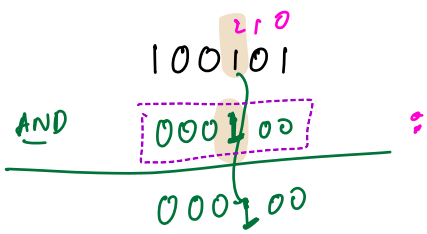
6 5 4 3 2 1 0

0 0 1 1 0 1 1 : set





N = 37, i = 2, check whether ith bit is set.



: $1 << i$ / 2^i

1

00000001<<3

0000 1000

Code

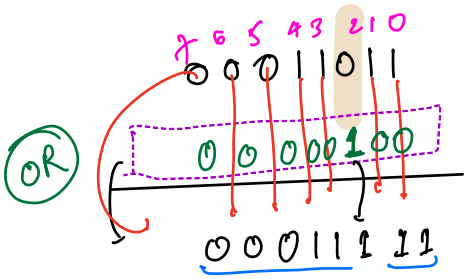
```
checkIthbit( int N, int i)
{
        mask = 1 << i
        res = N & mask
        If (res == 0)
            return false    // Bit was unset.
        else
            return True     // Bit was set.
}
```

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

: Make ith bit 1.

(no change if it is already one).

N = 27, i = 2

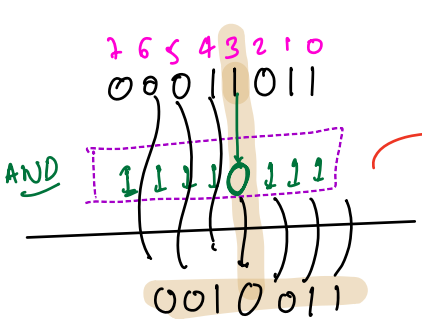



```
set Ith bit (int N, int i)
{
    mask = 1 << i ;
    return N | mask ;
}
```

T.C. → O(1)

Q. Given a no. N, clear the ith bit.
(Make the ith bit 0.)

N=27, i=3



NOT

$= \sim(00010000)$

$= \sim(1<<i)$

int a=5 = 32 bits

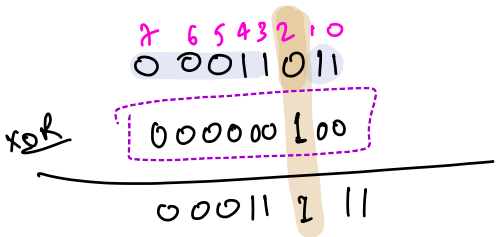
```
clearIthbit(int N, int i)
{
        mask = ~(1<<i);
        return N&mask;
}
```

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

→ 1 → 0

0 → 1

N = 27 , i = 2




```
toggleIthbit(int N, int i)
{
    mask = 1<<i
    return N ^ mask;
}
```

N = 11011

^ 00100

11111 ✓

Break till 8:24 AM

Q. Count no. of set bits in N.

N=27

00000000000000011011 ans = 4

if int → 32 bits
long → 64 bits.

```
cnt=0
for(int i=0; i<32 ;i++)
 {
        if ( checkIthbit(N, i))
              cnt++;

 }
return cnt.
```

N=27

00000000000000011011 ans = 4

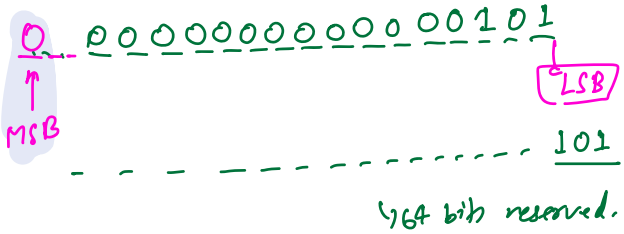
```
cnt=0
while(N>0)
 {   if ( N&1 == 1)
              cnt++
     N = N >>1

 }
return cnt.
```

Negative nos.

```
int a = 5 ;
```

32 bits.

```
long a=5;
```



(MSB is reserved as a sign bit.)

↳ MSB is 1 if it is a negative no.
MSB is 0 if it is a positive no.

8 bit nos.

00000101 = 5

(+) 1000 0101 ≠ -5

≠ 0 = 0

How to represent negative no.?

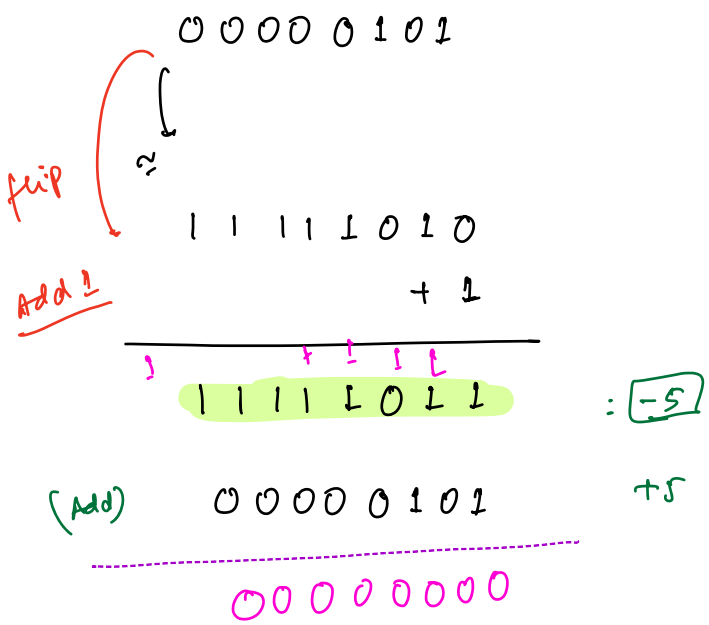
"2's complement"

To get binary repren of $-N$

1. Get Binary repren of N.
2. Invert the no. (flip all bits)
3. Add 1.

Binary rep. of -5

00000101 = 5



3: 00000011

11111100

+ 1

11111101 = -3

10: 00001010

11110101

+ 1

11110110 : -10

Binary to Decimal

7 6 5 4 3 2 1 0

00101010 : $2^5 + 2^3 + 2^1$

7 6 5 4 3 2 1 0

10101111 : $2^7 + 2^5 + 2^3 + 2^2 + 2^1$ (upto now).

7 6 5 4 3 2 1 0

10101111 = $-2^7 + 2^5 + 2^3 + 2^2 + 2^1$

place value = -2^7

sign bit

int a = 5

$-128 + 32 + 8 + 4 + 2$

$= -128 + 32 + 14$

$= -128 + 46$

$= -82$

4 bit nos.

If 4 bits, how many nos. can you represent ?

_ _ _ _

2^4 nos. = 16 nos.

0000	: 0
0001	: 1
0010	: 2
0011	: 3
0100	: 4
0101	: 5
0110	: 6
0111	: 7
1000	: -8
1001	: -7
1010	: -6
1011	: -5
1100	: -4
1101	: -3
1110	: -2
1111	: -1

[Min no. = -8
Max no. = 7]

if sign is not considered,

0000 → 0

1111 → 15

$-2^3+2^2+2^1+2^0$
$= -1$

(_) _ _ _ → 0 to 15 ?

sign

or -8 to 7 ?

_ _ _ _ : 0-15 if unsigned

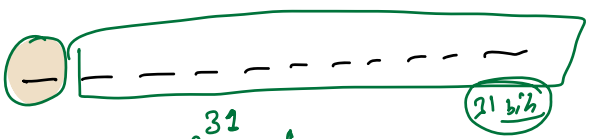
: -8 to 7 if signed.

int a=5;

a=-3 ✓

[int no. → 32 bits]

compiler assumes that it is signed.



max: $2^{31}-1$

Min: -2^{31}

unsigned int a=5;

a=-3; ? XX

error

[32 bits.
No sign bit!]

Max: $2^{32}-1$

Min: 0

1 0000000

3 2 1 0

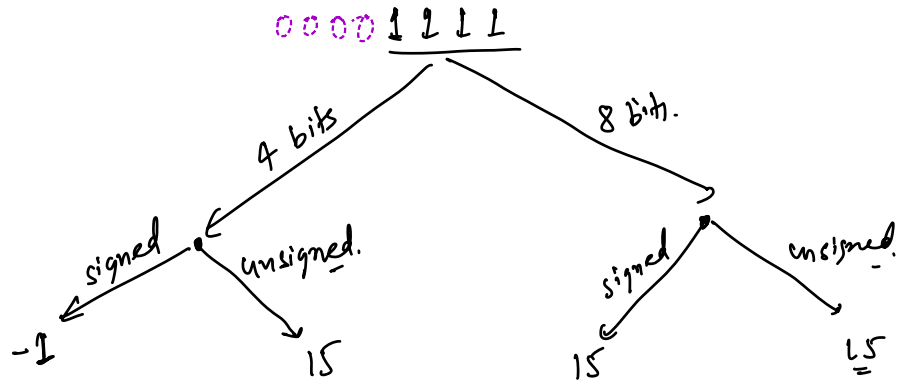
1 1 1 1

$= 2^4 - 1 = 15$

$2^3 + 2^2 + 2^1 + 2^0$

$= \frac{1*(2^n-1)}{2-1}$

$= 2^n - 1$



Range of Data types

		Min	Max
int (32 bits)	signed.	-2^{31}	$2^{31}-1$
	unsigned	0	$2^{32}-1$
long (64 bit)	Signed.	-2^{63}	$2^{63}-1$
	unsigned.	0	$2^{64}-1$

Constraints

```
int a = 10^5
int b = 10^6

int c = a*b
```

! Overflow


```
long c = a*b    XXX

long c = (long) a*b
```

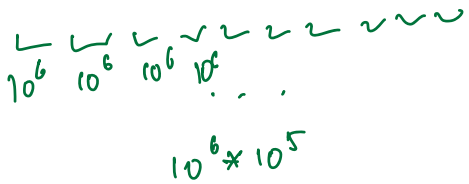
$1 <= N <= 10^5$

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

Sum of nos. in the array.

```
int sum=0
for(int i=0; i<n; i++)
{
        sum += arr[i]
}
return sum
```

long



Max possible sum value

$= 10^{11}$

10^{11} cannot be stored in int.

Range of Data types & Constraints.

Bit masking :

$10 = 2^{\log_2 10}$

$10^{11} = \left(2^{(\log_2 10)}\right)^{11}$

$= 2^{\log_2 10 * 11}$

$= 2^{3.1 * 11}$

$= 2^{33...}$

3.

32 bit

64 bit

5 & 2

→ Bitwise operator

int a=5;

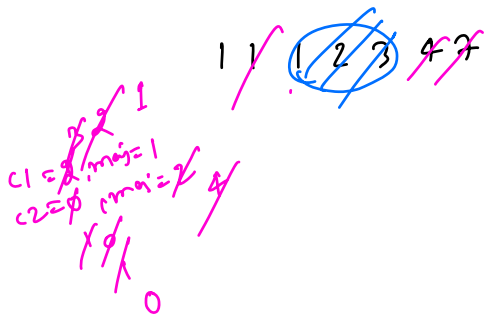
0 : 0 0 0 0 0 → 0th value

1 : 0 0 0 0 1 → 1st

2 : 0 0 0 1 0 → 2nd

3 : 0 0 0 1 1 → 3rd value

4 :


1 1 1 2 3 4 7
1
c1=ϕ max=1
c2=ϕ max=2
ϕ
0