New Chapter

Bit Manipulation

#Binary Number System

* Decimal to Binary
  divide like LCM
  $(4)_{10} = (100)_2$

2	4	
2	2	0
	1	0

* Binary to Decimal
  Multiply with $2^{0,1,2\cdots}$
  $(100)_2 = (4)_{10}$
  $1\times 2^2 + 0\times 2^1 + 0\times 2^0 = 4$

Bitwise Operators

* Binary AND &

Rules-

0 & 0 0
0 & 1 0
1 & 0 0
1 & 1 1

eg: 5 & 6

A = 0101 B = 0110

```
  101
& 110
-----
  100  = (4)10
```

⇒ 5 & 6 = 4

* Binary OR |

Rules-

0 | 0 0

0 | 1 1

1 | 0 1

1 | 1 1

eg: 5 | 6

A = 0101 B = 0110

```
  1 0 1
| 1 1 0
-------
  1 1 1
```

5 | 6 = $(7)_{10}$

* Binary XOR ^

Rules- When same gives zero

0 ^ 0 0

0 ^ 1 1

1 ^ 0 1

1 ^ 1 0

eg: 5 ^ 6

```
  1 0 1
^ 1 1 0
-------
  0 1 1
```

5 ^ 6 = $(3)_{10}$

* Binary One's Complement ~ (NOT)

works only on one bit

Rules-

~0 1

~1 0

eg: -5

↓

~0101

0 1 0 → $(2)_{10}$

This is wrong

PTO

This is what actually happens —

In computers, numbers are ~~not~~ stored in many bits, eg: 5 is not stored as 101 it is stored as 00 ... 00101

Process —

$\sim 5 \rightarrow 00000101$

$\sim$ ↓

11111010

MSB

MSB is one
So we take 2's comp.

↓ 1st comp

00000101

↓ +1

$(00000110) \Rightarrow (6)_{10}$

This is minus 6 due to ~~MSB~~ 2's comp method

$\therefore \quad \sim 5 = -6$

Special case

$\boxed{\sim 0}$

$\sim 0 = -1$

0000

↓

1111

↓

$0000 \rightarrow 0000 = -1$

* Binary left shift <<

Rules

a << b means all digits of a will shift by 'b' spaces and the spaces will be filled zero.

formula-

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

eg: 5 << 2

A = 000101

5 << 2 = 010100 = $(20)_{10}$

* Binary right shift >>

same as left just right in place of left.

000101

0000 1 0 1

eg: 6 >> 1 A = 000110

6 >> 1 = 000011 = $(3)_{10}$

formula-

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

Property-

In odd nos → LSB = 1
even nos → LSB = 0
} in binary

Q) Check if a no is even or odd

Steps

1) for this we will & the number with 1, coz it will gives us output 1 or zero base upon the LSB.

eg: 3 & 1

```
  011
& 001
-----
  001 → odd
```

4 & 1

```
  100
& 001
-----
  000 → even.
```

1<<0

Operations

Q) Get ith bit

In odd Even we needed 0^{th} bit's value that's why we were using & 1 or let's say & (1<<0)

But here we need ith bit so we will use $\underline{\&(1 << i)}$.

eg: $(15)_{10}$ get 2^{th} bit Bitmask = i<<i

$\Rightarrow i = 2$

15 = 0000 1111

&1<<2 = 00000100

00000100 $\Rightarrow$ a non zero number

i.e. i^{th} bit == 1

Q) Set ith bit

means change ith bit to 1

∴ 1) bitmask = 1<<i

2) n | bitmask. as OR will set ith bit as 1, coz any | 1 is 1.

Q) Clear ith bit

means change ith bit to zero

∴ 1) for zero, bitmask = ~(1<<i)

2) n & bitmask.

eg: ~~100~~ $(10)_{10}$ clear 1th bit

i = 1 n = 1010

bitmask = ~(1<<1) = ~(0010)

= 1101

```
  1010
& 1101
------
  1000  = (8)10
```

Q) Update ith bit

method 1

if / else

method 2

1) Store the no in new variable by clearing the ith bit

2) Bitmask = newBit << i ; n | Bitmask

why ? ↓

clear case

nB = 0

nB << i

0 << i = 0

So when

0 | n = n

∴ We will have n with clear bit from step 1

set case, nB = 1

nB = 1

(nB << i) | n

↓

same as set bit

∴ We will have n with ith bit set.

Q) Clear last i bits

Smjho →

when are bits cleared → & with 0

n = 15 = 1111

i = 2

So here we will need 1100
which is opposite of 1<<2
This is value of -1<<2
and -1 is called ~0

∴ 1) Bitmask = ~0 << i

2) return n & Bitmask.

eg: 1111 -1<<2 = 1100
& 1100
1100 = 12

9) Clear range of bits

eg: n = 100 1 1 1 0 1 0 0 1 1 1 , i = 2, j = 7
(j=7, i=2)
range

for this we will need bitmask

↓

[1 1 1 1 0 0 0 0 0 0 1 1]

But how to get this?

↳ this is simply OR | of

1 1 1 1 0 0 0 0 0 0 0 0 = a
0 0 0 0 0 0 0 0 0 0 1 1 = b

∴
1) Bitmask = a | b
where a = (~0) << (j+1)
b = 1 << (i - 1)

2) return n & Bitmask;

explanation

a = simple as before

$b = 2^i - 1$

~~$b = 2^i$~~ $2^i = 1 << i$

$b = 2^i - 1$

$b = (1 << i) - 1$

in binary,

01 = 1 = $2^1 - 1$
011 = 3 = $2^2 - 1$
0111 = 7 = $2^3 - 1$
01111 = 15 = $2^4 - 1$

Q) Check if a no is Power of 2 or not

$4 \rightarrow 2^2$
$8 \rightarrow 2^3$
$7 \rightarrow X$

→ first lets see the pattern

$4 \rightarrow 100$
$3 \rightarrow 011$
$\& \rightarrow 0$

$8 \rightarrow 1000$
$7 \rightarrow 0111$
$\& \rightarrow 0$

if n is 2^x then **n & (n-1) == 0**

Q) Count Set bits

1) loop till n == 0
2) right shift & if LSB == 1 count ++
3) return count

Complexity = O(log n)

i.e. = no of bits required to represt no which is (log n + 1)

So T. comp. = O(log n)

Q) Fast Exponentiation

for calculating a^N it takes only

$O(\log n)$ time

eg:

$a^5 = a^{101} = (1 \times a^4) * (1) \times (1 \times a)$

~~assign a~~ ~~binary 2's~~

$= a^5$

$3^5 = 3^{101} = (1 \times 3^4) * (1) \times (1 \times 3^1)$

$= 3^5$

Steps -

1) variables, a = 3 (value)
   ans = 1 , n = power

2) loop till n > 0
   if LSB == 1
   ans = ans * a
   a = a * a
   n = n >> 1

3) return ans;