DSA SERIES

- Learn Coding

Topic to be Covered today

Bit Manipulation

LETS START TODAY'S LECTURE

Bit Manipulation

Lecture - 37

Bit Manipulation

① How to convert a decimal number into binary format.

⇒ Binary matlab 0 and 1.

$(\text{Decimal})_{10} \longrightarrow (\text{Binary})_2$

This indicates the format of number.

There are different format of numbers:-

- Octal Number (8)
- Hexadecimal (16) (0 to 9 & A-F)
- Decimal (10)
- Binary (2)

$(15)_{10}$ ⇒

2	15	1
2	7	1
2	3	1
	1	

$(15)_{10} = (1111)_2$

$(16)_{10} =$

2	16	0
2	8	0
2	4	0
2	2	0
	1	

$(16)_{10} = 10000$

$\Rightarrow$

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

* How to Revert Back a Binary number to Decimal?

$$(10000)_2 = 2^4 \times 1 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 0$$
$$= 16 + 0 + 0 + 0 + 0$$
$$= 16$$

Now Lets form number 1 to 20?

1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001
10 = 1010
11 = 1011
12 = 1100
13 = 1101
14 = 1110
15 = 1111
16 = 10000
17 = 10001
18 = 10010
19 = 10011
20 = 10100

Code for converting Decimal to Binary

* string func(int n){

```
string func(int n){
    res = " ";
    while (n > 0) {
    if (n%2 == 1)  res+='1';
    else
        res+='0';
    n=n/2;
    }
    reverse (res);
    return res;
}
```

*Code for converting Binary to decimal

```
int func(string x) {
int len = x.length();
      =0;
for ( i = len-1 → 0) {
    if (x[i] == '1')
        num = num *p

int n = x.length();
 int res =0;
 for (int i=0 ; i<n; i++){
     if (x [i] = = '1'){
     res += pow (2, n-1-i);
     }
 return res;
}
```

0 1 2 3

1 1 1 1

3 2 1 0

n = (4-1-i)

$\boxed{4-1-1}$

1's Complement.

Just flip the bits.

5 = 0101

= 1111

Most significant bit = 0

signed → positive
→ Negative

MSB = 1.

unsigned → only non-negative number.

2's complement.

↳ It is obtained by
- taking 1's complement
- Adding 1 to the result.

Suppose :- We need to store −10 int 8 bit-binary.

```
    00001010
 ↳ 11110101
          +1
 ___________
    11110110
```

So. −10 in 2's complement is 11110110

Computer do not directly store negative number.
They store them in 2's complement form.

Operators in Bit manipulation.

i) AND (&)
ii) OR (|)
iii) XOR (^)
iv) NOT (~)
v) Left shift (<<)
vi) Right shift (>>)

Addition Rule.

$$\begin{cases} 0+0 = 0 \\ 0+1 = 1 \\ 1+0 = 1 \\ 1+1 = 0 \text{ (carry move to next higher bit)} \end{cases}$$

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

```
  1 1 1 1
      + 1
---------
1 0 0 0 0
```

i) And. (&)

Results 1 if both bits are 1, else 0.

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

ii) OR (|)

- Results 1 if at least one bit is 1.

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

(iii) XOR (^)

Exclusive OR. : Result 1 if bits are different, else 0.
Pairing मे नहो ~~कि~~ होने चाहिए।

(iv) NOT (~) $\sim N = -(N+1)$

$0 \rightarrow 1$

$1 \rightarrow 0$

(v) Left shift: (<<)

↳ shifting all bits to the left empty bits with 0.

↳ Equivalent to multiplying 2^n.

16 8 4 2 1

$5 \rightarrow 0101$

$01010 \Rightarrow 5 \times 2^1$

8 + 2 = 10

$10100 = 5 \times 2^2$

16 + 4 = 20 $= 5 \times 4$

$= 20$

(vi) Right shift (>>)

↳ shift all bits to right

↳ Equivalent to dividing by 2^k.

16. // 10000

00100

$\frac{16}{2^2} = \frac{16}{4} = 4$

Basic Questions.

(1) Swap two number

(i) Earlier we used a third variable to do so.

(ii) $a = a + b$
$b = a - b$
$a = a - b$

(iii) $a = a \wedge b$
$b = a \wedge b \Rightarrow (a \wedge b) \wedge b = a$
$a = a \wedge b \Rightarrow (a \wedge b) \wedge \underset{a}{b}$
$\Rightarrow \boxed{b}$

(2) Check if ith bit is set or Not.

0 0 1 1 → check 0th bit is set or not.
→ $1 << i$ ⟵ make a mask.
& 0 0 0 1 ⇒ $1 << 0$

0 0 0 1

$\boxed{((N \& (1 << i)) != 0)}$ → It means that particular bit is set.

(3) Set the ith bit

$N = 8$, $i = 2$

1 0 0 0

1 0 0 0
OR. 0 1 0 0

1 1 0 0

$\boxed{N \mid (1 << i)}$

4. Clear the ith bit

→ Turn 1 to 0, if it is 0 keep it 0.

1 1 0 1

↑

clear this.

So, we can bring 0 here and to the AND operation.

```
  1 1 0 1
  1 0 1 1      ⟶   (0 1 0 0)
  -------
  1 0 0 1
```

∴ $\boxed{N \,\&\, (\sim(1<<i))}$

5. Toggle the ith bit

↳ We need to simply ~~the~~ reverse the bit value from there.

```
  1 [1] 0 1
^ 0  1  0 0
  ---------
  1  0  0 1
```

(6) Remove the last set bit (Rightmost)

N = 16 ⇒ 1 0 0 0 0
0 0 0 0 0 &
N = 15 ⇒ 0 1 1 1 1

N = 40 ⇒ 1 0 1 0 0 0
N = 39 ⇒ 1 0 0 1 1 1 &
$\boxed{1\ 0\ 0\ 0\ 0\ 0}$

(N & N − 1)

(7) ~~Check~~ Count the number of setbits in a given number.

```
int countSetBit (int n) {
   cnt = 0;
   while (n > 1) {
     if (n%2 == 1) cnt++;
     n = n/2;
   }
   if (n==1) cnt++;
     return cnt;
   }
```

The last bit of odd number is always 1.

To check this

$(N \& 1) == 1 \Rightarrow N$ is odd
else N is even.

Second Method.

```
cnt = 0;
while (N != 0) {
    N = N & (N-1);
    cnt++;
}
```

Learn coding

THANK YOU