31/03/2024

PAGE NO.
DATE:

Day 24 of DSA.

Task)

CheckBox	Bit Manipulation
☑	Decimal to Binary (Theory) → Bitwise operators
☑	Binary to Decimal
☑	Solving Bit Manipulation problems.

PAGE NO.
DATE:

* Decimal to Binary :-

Approach :-

Pseudocode :-

```
int ans = 0;
int i = 0;
while ( n != 0)
{
    int bit = n & 1;
    ans = ( bit * pow(10, i) + ans;
    n = n >> 1;
    i++;
}
```

n = 5

i=1 bit = 101 & 1 = 1

ans = (1 * 10^0) + 0 = 10

n = 101 >> 1 = 010

i++

i = 2 bit = 010 & 1 = 0

ans = (0 * 10^1) + 10 = 10

n = 010 >> 1 = 001

i++

i = 3 bit = 1 & 1 = 1

ans = (1 * 10^2) + 10 = 100 + 1 = 101

n = 000 >> 1 = 0

break

PAGE NO.
DATE:

* Binary to Decimal

Approach:-

```
int i=0, ans=0;

while( n!=0)
{
    int digit = n %10;

    if (digit == 1)
    {
        ans = ans + pow(2,i);
    }

    n = n/10
    i++;
}
```

n = 5 101

i=0, ans=0

digit = 1015 %10 = 1

(1==1) = ans = 0 + 2^0 = 1

i=1, ans=1

digit = 10%10 = 0

(0==1) X

i=2, ans=1

digit = 1%10 = 1

(1==1)

ans = 1 + (2^2)

= 1+4 = 5

PAGE NO.
DATE:

Another Approach For Decimal to Binary.

Pseudocode :-

```
String decToBin(int n)
{
    String res = "";
    while (n != 0)
    {
        if (n % 2 == 1) res += '1';
        else res += '0';
        n = n/2
    }
    reverse(res);
    return res;
}
```

Time Complexity = $O(\log_2 n)$
Space complexity = $O(\log_2 n)$

PAGE NO.
DATE:

4. Another Approach For Binary to Decimal :

Approach :-

```
int BinToDeci(string x)
{
    int len = x.length();
    int p2 = 1, num = 0;

    for(i = len-1, i >= 0; i--)
    {
        if(x[i] == '1')
            num += p2;
        p2 *= 2;
    }

    return num;
}
```

Time Complexity : O(n)
space Complexity : O(1)

PAGE NO.
DATE:

* 1's Complement

(13) → $(1101)_2$

↓ Flip

$(0010)_2$ ✓

* 2's complement

1) 1's complement
2) add 1 to it

$(0010)_2$ $\xrightarrow{+1}$

$$\begin{array}{r} 0010 \\ +0001 \\ \hline (0011)_2 \end{array}$$ ✓

* AND :-

All true → true
, False → False

$$\begin{array}{r} 1101 = 13 \\ \& \; 0111 = 7 \\ \hline 0101 \\ \hline 5 \end{array}$$

* OR :-

1 true = true
all False = False

$$\begin{array}{r} 1101 = 13 \\ \| \; 0111 = 7 \\ \hline (1111) \\ \hline 15 \end{array}$$

* XOR :-

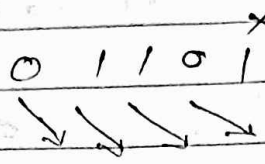
no. of 1s → odd → 1
no. of 1s → even → 0

$$\begin{array}{r} \text{XOR } 1101 = 13 \\ \quad\; 0111 = 7 \\ \hline (1010) \\ \hline 10 \end{array}$$

PAGE NO.
DATE:

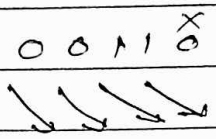
* Right shift :- >> $x >> k = \frac{x}{2^k}$

① 13 >> 1



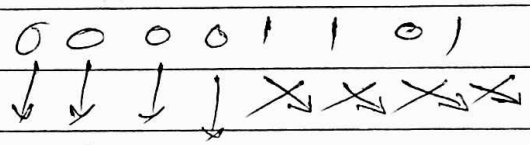
6

② 13 >> 2



3

③ 13 >> 4



0

* longest 2^{31}

+ve sign → 0 1 1 1 1

$(2^{31}-1)$ → INT-MAX

* smallest

→ 1 0 0 . . . . . 0 -2^{31}

-ve sign

INT_MIN

PAGE NO.
DATE:

* left shift : << num * 2^k

12 << 1

0 1 1 0 0

1 1 0 0 0 = 26

* Not (~)

1) Flip
2) check -ve

+ve -ve

2's stop

eg.

0 0 0 0 . . . 1 0 1

1 1 1 1 . . . 0 1 0

0 0 0 0 1 0 1

+ 1

1 . . . 1 1 0

PAGE NO.
DATE:

* Swap two Numbers:-

a = 6 b = 5

(∵ 5 ^ 5 = 0)

a = a ^ b

b = a ^ b = (a ^ b) ^ b = a

a = a ^ b = (a ^ b) ^ a = b

* Check if the ith bit is set or not.

N = 13 i = 2

& 1 1 0 1

1 << 2

0 1 0 0

0 0 1 0 0

← 0's = 0's →

① if ((N & (1 << i) != 0) Logic

{

set

}

else

{

Not set

}

PAGE NO.
DATE:

Logic

② `if ( (N >> i) & 1 == 0)`

```
{
    Not set
}
else
{
    set
}
```

* set the ith bit

N = 9 i = 2

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

or

Logic

```
if(N | (1<<i))
{
    setted
}
else { Not set }
```

* clear the ith bit

N = 13 i = 2

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

~(1<<2)
~(0100)
(1011)

Logic = (N & ~(1<<i)) ✓

PAGE NO.
DATE:

* Toggle the ith bit :-

N = 13 i = 2

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

Logic = N ^ (1 << i) ✓

* Remove the last set bit (rightmost)

N = 40

```
  1 0 1 0 0 0    N
& 1 0 0 1 1 1    N-1
---------------
  1 0 0 0 0 0   ✓
```

Logic = N & (N-1) ✓

* Check if the number is a power of 2

N = 16 = 2^4

Logic (N & N-1 == 0) ✓

```
  1 0 0 0 0
& 0 1 1 1 1
-----------
  0 0 0 0 0   ✓
```

PAGE NO.
DATE:

* Count set of bits.

(1) int csb(int n)

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

odd ke liye

```
cnt += n&1
n = n>>1
```

↓

n/2

*

(2)

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

Time complexity : $O(\log_2 n)$