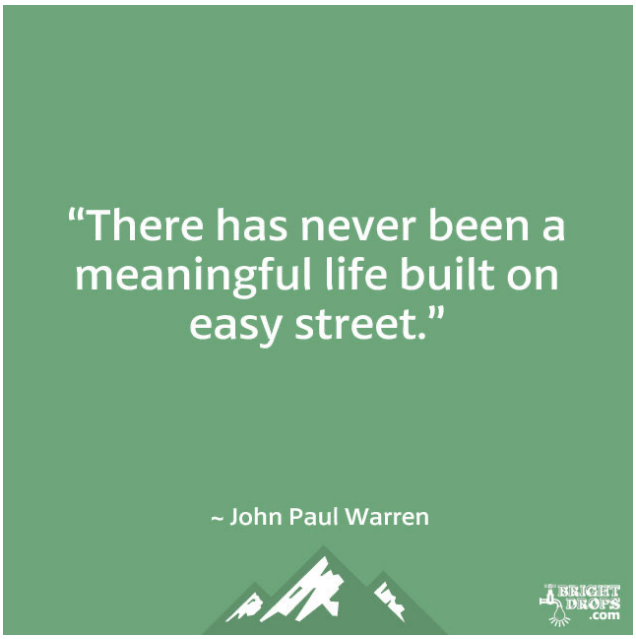
Bit Manipulation

Good Morning



NOTES:

* Basics of bit Manipulation
  - → Number system & conversions
  - → Addition of two binary numbers
  - → Bitwise operators
  - → Negative numbers
  - → Ranges of no.

Number System → Way of writing to express the number

01. Decimal no. system

→ base - 10 system

→ Digits = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$[7\ 3\ 2]_{10}$ 10^2 10^1 10^0

$= 7 * 10^2 + 3 * 10^1 + 2 * 10^0$

$= 700 + 30 + 2$

$= 732$ { dig * base value }

$[2\ 4\ 6\ 3]_{10}$ 10^3 10^2 10^1 10^0

$= 2 * 10^3 + 4 * 10^2 + 6 * 10^1 + 3 * 10^0$

$= 2000 + 400 + 60 + 3$

$\Rightarrow \underline{2463}$

* Binary No. system

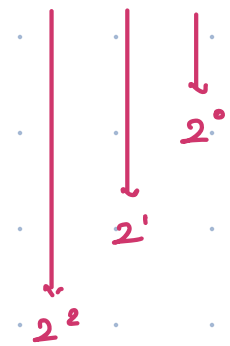
base - 2 system

Bits = 0 & 1

$[1\ 1\ 0]_2 = 1*2^2 + 1*2^1 + 0*2^0$

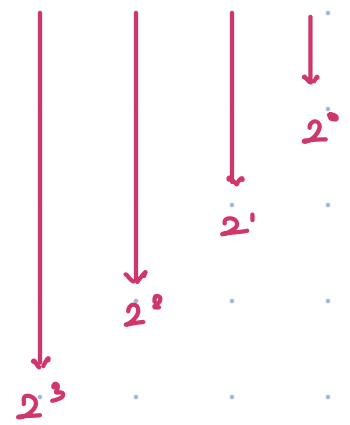
$= 4 + 2 + 0$

$= 6$

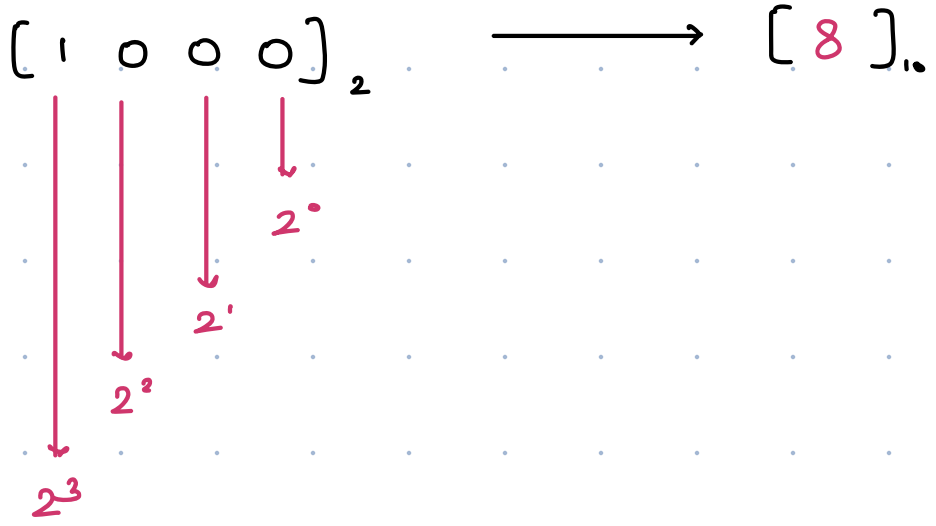


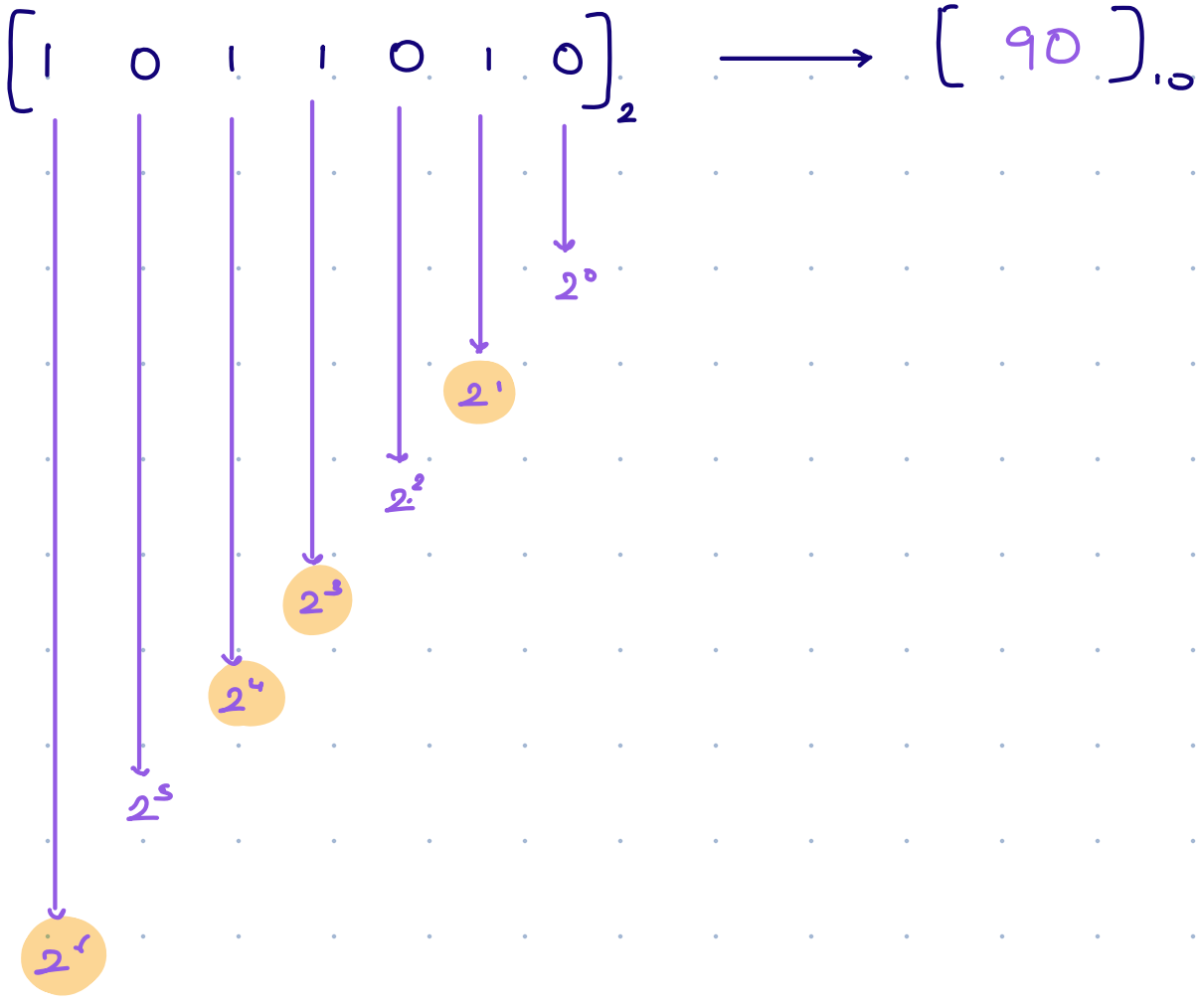
$[1\ 0\ 0\ 1]_2 = 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$

$= 8 + 1 = 9$



* Binary to Decimal conversion





Idea → Extract the digits one by one.
Multiply the extracted digit with its correct base value & add it to the ans.

```
int btod (int n)
{
    int ans = 0
    int mul = 1;
    while (n > 0) {
        int r = n % 10;
        n = n/10;
        ans = ans + r * mul;
        mul = mul * 2;
    }
    return ans;
}
```

TC: O(no. of digits)

TC: $O(\log_{10} n)$

SC: $O(1)$

TC lecture ⇒ $n \to \frac{n}{2} \to \frac{n}{4} \to \frac{n}{8} \cdots 1 \Rightarrow O(\log_2 n)$

$n \to \frac{n}{10} \to \frac{n}{100} \to \frac{n}{10^3} \cdots 1 \Rightarrow O(\log_{10} N)$

* Decimal to Binary

2	10	
2	5	0
2	2	1
2	1	0
	0	1

Remainders ↑

$[10]_{10} \longrightarrow [1010]_2$

02 $[45]_{10} \longrightarrow [101101]_2$

2	45	
2	22	$1 * 10^0 = 1$
2	11	$0 * 10^1 = 0$
2	5	$1 * 10^2 = 100$
2	2	$1 * 10^3 = 1000$
2	1	$0 * 10^4 = 0$
	0	$1 * 10^5 = 100000$

Remainders ↑

101101

Code → {TODO}

* Addition of Two Decimal no.

```
9/10   13/10   12/10
 (0)    (1)    (1)
  0      7      8      0
                       9
  0      1      4      3
```

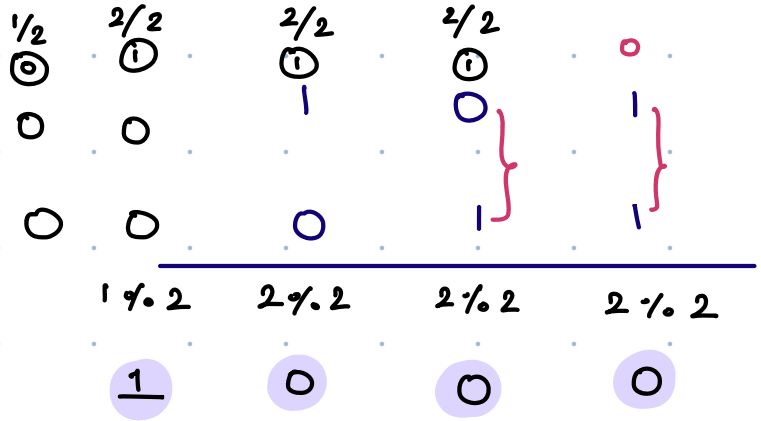
9%10 13%10 12%10

9 3 2

dig = sum % 10

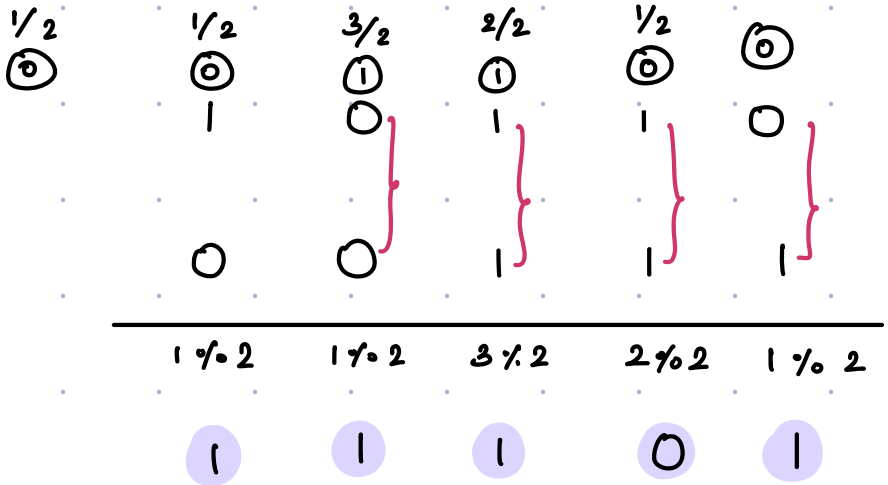
carry = sum / 10

* Addition two binary no.



dig = sum % 2

carry = sum / 2



*

carry = sum/2

rem = sum % 2

n_1	n_2	d_1	d_2	carry	d_1+d_2+c sum	sum % 10 rem	ans
89	143	9	3	0	12	2	$2 * 10^0$
8	14	8	4	12/10	13	3	$3 * 10^1$
0	1	0	1	13/10	2	2	$2 * 10^2$
0	0	—	—	2/10	—	—	232

```
int add (int n1, int n2)
{
    ans=0 , carry=0
    mul=1

    while (n1>0 || n2>0 || c>0)
    {
        int d1 = n1 % 10
        int d2 = n2 % 10

        n1 = n1 /10;
        n2 = n2 /10;

        sum = d1 + d2 + c;
        rem = sum % 10
        c = sum /10
        ans= ans + rem * mul;
        mul = mul * 10;
    }
    return ans
}
```

TC: $O(\log_{10} n)$

// rem = sum % 2

// c = sum/2

8:13 AM → 8:23 AM

Translate the number 15.9_{10} в binary like this:

the Integer part of the number is divided by the base of the new number system:

15	2		
-14	7	2	
1	-6	3	2
	1	-2	1
		1	

direction of gaze ←

the Fractional part of the number is multiplied by the base of the new number system:

0.	9*2
1	.8*2
1	.6*2
1	.2*2
0	.4*2
0	.8*2
1	.6*2
1	.2*2
0	.4*2
0	.8*2
1	.6*2

direction of gaze ↓

* Bitwise Operators → Perform all the operations on binary no. bit by bit

0 → false / unset

1 → true / set

01. AND &

02. OR |

03 XOR ^

04. NOT ~

05 Left shift <<

06 Righ Shift >>

A	B	A & B	A | B	A ^ B	~ A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

A & B = both the bits are set

A | B = Any one bit is set

A ^ B = same same puppy shame

↳ same bits = unset

↳ diff bits = set

Addition without carry

01. 5 & 6 = 4

$$
\begin{array}{rcl}
5 & = & 1\ 0\ 1 \\
\& & & \\
6 & = & 1\ 1\ 0 \\
\hline
 & & 1\ 0\ 0 \quad \Rightarrow 4
\end{array}
$$

02 20 & 45 = 4

```
20 = 0 1 0 1 0 0
&
45 = 1 0 1 1 0 1
-----------------
     0 0 0 1 0 0
```

03 20 ^ 45 = 57

```
20 = 0 1 0 1 0 0
^
45 = 1 0 1 1 0 1
-----------------
     1 1 1 0 0 1
```

04. 20 | 45 = 61

```
20 = 0 1 0 1 0 0
|
45 = 1 0 1 1 0 1
-----------------
     1 1 1 1 0 1
```

* Binary representation of Negative no.

$(-45)_{10} \rightarrow [\ ?\ ]_2$

int x = | 31 | 30 | 29 | 28 | ... | 3 | 2 | 1 | 0 |

2^{31} 2^{30} 2^{29} 2^{28} ... 2^3 2^2 2^1 2^0

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

$2^0 + 2^1 + 2^2 + \ldots + 2^{29} + 2^{30} \rightarrow$ GP

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

$$= \frac{1 * (2^{31} - 1)}{2 - 1}$$

$$= 2^{31} - 1$$

2^{31}

MSB

- 0 {no. is positive}
- 1 {no. is negative}

signed bit → means the MSB will tell if no. is positive or negative

unsigned bit → means the MSB doesn't have any significance

* $[-45]_{10} \longrightarrow [\quad\quad]_2$

Assumption int → 8 bits (Just for visual representation)

$(45)_{10}$ → 0 0 1 0 1 1 0 1

01. 1's compliment = Toggle every bit of n

02 2's compliment = 1's compliment + 1

$(45)_{10}$ → 0 0 1 0 1 1 0 1

1's complim → 1 1 0 1 0 0 1 0

+ 1 0 0 0 0 0 0 0 1

$[-45]_{10}$ → 1 1 0 1 0 0 1 1 (bits 7 6 5 4 3 2 1 0)

$-2^7 + 2^6 + 0 + 2^4 + 0 + 0 + 2^1 + 2^0$

$= -128 + 64 + 16 + 2 + 1$

$\Rightarrow -128 + 83$

$\Rightarrow \underline{-45}$

* $[-3]_{10} \longrightarrow [\quad\quad]_2$

$[3]_{10} \rightarrow$ 0 0 0 0 0 0 1 1

1's complm → 1 1 1 1 1 1 0 0

+ 1 0 0 0 0 0 0 0 1

$[-3]_{10} \rightarrow$ 1 1 1 1 1 1 0 1

* Range of ~~the~~ numbers

01. Byte = 8 bits

min num = 1 0 0 0 0 0 0 0 $\Rightarrow -128$

$= -2^7$

max num = 0 1 1 1 1 1 1 1 $= 127$

$= 2^7 - 1$

-128 to 127

-2^7 to $2^7 - 1$

02 int → 32 bits

min num = 1 0 0 0 0 . . . . . 0 0 0 = $-2^{31} \simeq -10^9$

max num = 0 1 1 1 1 . . . . . 1 1 1 = $2^{31}-1 \simeq 10^9$

Range = -2^{31} to $2^{31}-1$

= -10^9 to 10^9

03 long → 64 bits

Range = -2^{63} to $2^{63}-1$

= $-9*10^{18}$ to $9*10^{18}$

* Importance of constraints

Given two integers a & b. Return a * b

$a \leq 2*10^9$

$b \leq 2*10^9$

01. `int ans = a * b` ⟶ Incorrect (Overflow)

`return ans`

02. `long ans = a * b` ⟶ Incorrect (Overflow)

`return ans;`

`int * int`

`int` ⟶ garbage value

03 `long ans = long (a * b)` ⟶ Incorrect (Overflow)

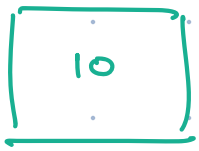
`return ans`

04 `long ans = ((long)(a)) * b` ⟶ correct

`long * int`

`long`

05
```
long ans = a;
ans *= b;
return ans;
```
} correct



ans

a

long * int

long

Leetcode Easy

long elecount = (n-1) * (i+1)

Problem

Constraints

Time complexity

TLE 10^7 to 10^8

$n = 1000$