Day 34

Bit Manipulation

Binary Numbers – Numbers in the form of 0 and 1.

Decimal to Binary $(13)_{10} \rightarrow$

2	13	1 ↑
2	6	0
2	3	1
	1	

$(1101)_2$

Addition in Binary Numbers →

```
  1 1
  1 0 1   → (5)
  1 1 1   → (7)
-------
1 1 0 0   → (12)
```

Subtraction in Binary Numbers →

$9 - 7 = 2$

$\Rightarrow 9 + (-7) = 2$

Now in binary

```
  1 1 0 0
- 1 0 1   ← find negative inverse of this number.
-------
    ?
```

↓

2's complement

2's complement → ① Invert all bits

② add 1

```
1100   (12)        -5 → 0000101        2's complement = 1111010
- 101  (-5)                                          +       1
                                                    ---------
                                                     1111011
```

Now add 5's complement,

```
  001100   (12)
+ 111011   (5's complement)
--------
  000111   (7)
```

Example 2 9 - 3 in binary.

9 = 1001
3 = 0011

2's complement of 3 = 1100 + 1 = 1101

add 9 & 3's complement

```
  1001   (9)
+ 1101   (3's complement)
-------
 00110   (6)
```

Bitwise Operator

a	b	& (AND)	| (OR)	^ (XOR)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

~ → Invert every bit

5 = 00101
~5 = 11010

Right Shift Operator

12 >> 2
1100 >> 2 ⇒ 0011 (3)

Left shift Operator

12 << 2 → 110000 (48)

1100

0000

- Right shift = dividing by 2
- Left shift = multiplying by 2

Benifit: So instead of using / and .* do left shift or right shift. Operations becomes faster and TLE will not occur.

Odd / Even Number

```
if (a % 2 == 0)
{
  Number is even
}
else
{
  Number is odd
}
```

```
if (a & 1 == 0)
{
  Number is even
}
else
{
  Number is odd
}
```

faster than previous one

Last Bit of Even number = 0

2 →	010
4 →	100
6 →	110

Last Bit of Odd number = 1

3 →	011
5 →	101
7 →	111

Swap 2 numbers

a = 2 b = 3	a = 2 (010) b = 3 (011)
a = a + b (5)	a = a ^ b = 001
b = a - b (2)	b = a ^ b = 010 ⇒ 2
a = a - b (3)	a = a ^ b = 011 ⇒ 3

Bit Masking

Find i^{th} bit

n → 1 0 0 1 1 0 1 0 1

mask → 0 0 0 1 0 0 0 0 0

n & mask → non-zero → 1

↳ 0

Let n = 100110101, find 5^{th} bit

Method 1 → Right shift n 5 times so that digit at 5^{th} position becomes last digit & check if it one or zero by even/odd method.

Method 2 → Masking, left shift 1 for 5 positions and then (&) and then them. If answer is non-zero, bit is 1 else 0.

8. Set Bit

Set bit at n^{th} position

```
          8 7 6 5 4 3 2 1 0
n →       1 0 0 1 1 0 1 0 1
mask →    0 0 0 0 0 1 0 0 0
          ---------------
          1 0 0 1 1 1 1 0 1
```

Set 3rd bit as 1, so left shift 1 to 3rd position and use OR operator

```
mask = 1 << i
n = n | mask
```

```
          8 7 6 5 4 3 2 1 0
n →       1 0 0 1 1 0 1 0 1
mask      1 1 1 1 1 0 1 1 1
          ---------------
          1 0 0 1 0 0 1 0 1
```

Set 4th bit as 0, for that we need a mask such that all other digits remain same & only 4th one changes. OR operator cannot be used since 1 OR anything is 1 & and we want 0. So using & operator, put 1 at every place & except 4th

```
mask = ~(1 << i)
res = n & mask
```