Today's Content :

→ Number system basics

→ Binary to decimal & Viceversa

→ Adding 2 Binary numbers

→ Bitwise operators

  → Basic properties

  → Basic problems

→ Left shift & Right shift

  → check Bit

Number system Basics → Decimal Number System

- [0–9]
- power of 10

$734 - 700 + 30 + 4 = 7*10^2 + 3*10 + 4$

$6594 - 6*10^3 + 5*10^2 + 9*10^1 + 4*10^0$

$2456 - 2*10^3 + 4*10^2 + 5*10^1 + 6*10^0$

$10^3 \; 10^2 \; 10^1 \; 10^0$

Number Systems:

Octal

- every digit [0 – 7]
- every power [8]

$-(125)_8$

$-(125) = 1*8^2 + 2*8^1 + 5*8^0 = 64 + 16 + 5$
$= 85$

$8^2 \; 8^1 \; 8^0$

Binary

- every digit [0 1]
- every power (2)

$(10100)_2 = \begin{array}{ccccc} 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{array} = 16 + 4 = 20$

$2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

Binary Number System

1) $(\overset{4}{1}\overset{3}{0}\overset{2}{1}\overset{1}{1}\overset{0}{0})_2 = 2^4 + 2^2 + 2^1 \Rightarrow 16 + 4 + 2 \Rightarrow 22$

2) $(1\,2\,0)_2 =$ Invalid Binary? If binary [0 1]

hence not binary?

Decimal to Binary:

2	37	1
2	18	0
2	9	1
2	4	0
2	2	0
2	1	1
	0	

$(\overset{5}{1}\overset{4}{0}\overset{3}{0}\overset{2}{1}\overset{1}{0}\overset{0}{1}) \Rightarrow 2^5 + 2^2 + 2^0 \Rightarrow 32 + 4 + 1 \Rightarrow 37$

2	25	1
2	12	0
2	6	0
2	3	1
2	1	1
	0	

$(11001)_2$

$\Rightarrow 16 + 8 + 1 = 25$

2	19	1
2	9	1
2	4	0
2	2	0
2	1	1
	0	

$(\overset{4}{1}\overset{3}{0}\overset{2}{0}\overset{1}{1}\overset{0}{1})$

$\Rightarrow 16 + 2 + 1 \Rightarrow 19$

Add 2 Decimal Numbers? → Base : 10

$d = sum \% 10$, $c = sum / 10$

$$
\begin{array}{ccc}
1 & 1 & \\
7 & 8 & 9 \\
1 & 4 & 2 \\
\hline
9 & 3 & 1
\end{array}
$$

$$
\begin{array}{cccccc}
 & & 17/10 & 8/10 & & \\
 & & 1 & 0 & 1 & \\
11/10 & & 7 & 8 & 3 & 9 \\
1 & & 3 & 9 & 4 & 8 \\
\hline
 & & 11 & 17 & 8 & 17 \\
 & 1 & 1 & 7 & 8 & 7
\end{array}
$$

Add 2 Binary Numbers?

$d = sum \% 2$, $c = sum / 2$

Ex:

$$
\begin{array}{cccccc}
 & 2/2 & 3/2 & 2/2 & 1/2 & \\
 & 1 & 1 & 1 & 0 & \\
 & 1 & 0 & 1 & 1 & 0 \\
2/2 & 0 & 1 & 1 & 1 & 1 \\
1 & & & & & \\
\hline
 & 2\%2 & 2\%2 & 3\%2 & 2\%2 & 1\%2 \\
1 & 0 & 0 & 1 & 0 & 1
\end{array}
$$

$$
\begin{array}{cccccc}
 & 1/2 & 1/2 & 2/2 & 2/2 & \\
 & 0 & 0 & 1 & 1 & \\
a = & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 \\
\hline
 & 1\%2 & 1\%2 & 1\%2 & 2\%2 & 2\%2 \\
\cancel{0} & 1 & 1 & 1 & 0 & 0
\end{array}
$$

Bitwise operators? → { and, OR, XOR, ~, <<, >> }

- and → &
- OR → |
- XOR → ^
- ~ → Inverse
- << → left
- >> → rightshift

[Addition without carry]

Truth Table?

- If both bits are 1 = 1 (a & b)
- If both bits are = 0 = 0 (a | b)
- Same Same Puppy Shame (a ^ b)
- flip every bit of 0 ⇒ 1, 1 ⇐ 0 (~a)

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

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1: 2/2 = 1, 2%2 = 0

// Basic problems in Bitwise operators?

a = 29, b = 19

	4	3	2	1	0	
a :	1	1	1	0	1	
b :	1	0	0	1	1	
print (a & b) :	1	0	0	0	1	→ 17
print (a | b) :	1	1	1	1	1	→ 31
print (a ^ b) :	0	1	1	1	0	→ 14

Bitwise and
print func

a = 13, b = 10

a :	1	1	0	1	
b :	1	0	1	0	
a & b :	1	0	0	0	= 8
a | b :	1	1	1	1	= 15
a ^ b :	0	1	1	1	= 7

// Properties

8 & 2 & 1

$a = 10$: 1 0 1 0

print(a&1) : 0 0 0 1

= 0 ; 0 0 0 0

$a = 11$: 1 0 1 1

print(a&1) : 0 0 0 1

0 0 0 1

$a = 14$: 1 1 1 0

a&1 1 : 0 0 0 1

0 0 0 0

$a = 13$: 1 1 0 1

print(a&1) : 0 0 0 1

0 0 0 1

```
if ( a&1 == 1) {
    a is odd
    0th bit in a = 1
}
else {
    a is even
    0th bit in a = 0
}
```

//

$a \& 0 = 0$

$a \& a = a$

$a \mid 0 = a$

$a \mid a = a$

$a \wedge 0 = a$

$a \wedge a = 0$

a&1 == 1

a is odd

0th bit in a is 1

else

a is even

0th bit in a is 0

//

Bitwise Arthimetic & Associativity?

$a \& b = b \& a$

$a \mid b = b \mid a$

$a \wedge b = b \wedge a$

$a \& b \& c = c \& b \& a$

$\underbrace{a}_{P_1} \& \underbrace{(b \& c)}_{P_2}$

$\underbrace{(b \& c)}_{P_1} \& \underbrace{a}_{P_2}$

$(c \& b) \& a = c \& b \& a$

// $a \& b \& c = c \& b \& a = b \& a \& c =$

// $a \mid b \mid c = c \mid b \mid a = b \mid a \mid c =$

// $a \wedge b \wedge c = c \wedge b \wedge a = b \wedge a \wedge c =$

Ex1: $a \wedge b \wedge d \wedge b \wedge a$

$= a \wedge a \wedge b \wedge b \wedge d$

$0 \quad 0 \quad d$

Ex2: $a \wedge d \wedge e \wedge a \wedge e =$

$= a \wedge a \wedge e \wedge e \wedge d$

$0 \quad 0 \quad = d$

1Q) Given N array elements, every elements repeats twice except 1, find unique element?

ar[5] = { 6, 9, 6, 10, 9 } = 10

ar[7] = { 12, 9, 12, 8, 7, 9, 8 } = 7

ar[5] = { 2, 9, 7, 2, 7 } = 9

res = 0 → 2 → 11 → 12 → 14 → 9

2 : 0 0 1 0
9 : 1 0 0 1
11 : 1 0 1 1
7 : 0 1 1 1
12 : 1 1 0 0
2 : 0 0 1 0
14 : 1 1 1 0
7 : 0 1 1 1
: 1 0 0 1

// idea: Take xor of all Elements

```
res = 0;
i = 0; i < N; i++) {
    res = res ^ ar[i]
}
return res
```

```
res = ar[0]
i = 1; i < N; i++) {
    res = res ^ ar[i]
}
return res
```

TC: O(N) SC: O(1) → 9:42am

// left shift <<

↳ move all bits to left

8 bit number } → Only store 8 bits

	7	6	5	4	3	2	1	0	Decimal
a = 10 :	0	0	0	0	1	0	1	0	= 10
a<<1 :	0	0	0	1	0	1	0	0	= 20 = 10*2
a<<2 :	0	0	1	0	1	0	0	0	= 40 = $10*2^2$
a<<3 :	0	1	0	1	0	0	0	0	= 80 = $10*2^3$
a<<4 :	1	0	1	0	0	0	0	0	= 160 = $10*2^4$
a<<5 : 1	0	1	0	0	0	0	0	0	= $10*2^5$ = 320

dis

dis

dis

discard

discard

↳ overflow

→ In general No Overflow

a<<1 = a*2

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

a<<3 = $a*2^3$

a<<N = $a*2^N$

a<<N = $a*2^N$

// a = 10, N = 3

10<<3 = $10*2^3$ → 80

// a = 10, N

10<<N = $10*2^N$

$\boxed{a << N = a * 2^N}$

$1 << 1 = 1 * 2^1 = 2$

$1 << 2 = 1 * 2^2 = 2^2$ ⇒ N = 10, Calculate 2^{10} : (1 << 10)

$1 << 3 = 1 * 2^3 = 2^3$ ⇒ TC → for all bitwise operators : O(1)

$1 << 4 = 1 * 2^4 = 2^4$

If overflow not then formulae holds.

$\boxed{1 << N = 1 * 2^N}$

$a << N = a * 2^N$

$5 << N = 5 * 2^N$

Calculate $5^N = 5 << N$ ✗

only 2 powers we can get with left shift.

// Right Shift : move bits to it's right

	7	6	5	4	3	2	1	0		Decimal
N = 10 :	0	0	0	0	1	0	1	0		= 10 =
N>>1 :	0	0	0	0	0	1	0	1	dis	= 5 = $10/2$
N>>2 :	0	0	0	0	0	0	1	0	dis	= 2 = $10/2^2$
N>>3 :	0	0	0	0	0	0	0	1	dis	= 1 = $10/2^3$
N>>4 :	0	0	0	0	0	0	0	0	dis	= 0 $10/2^4$

N>>5 : $10/2^5$ ⇒ 0

$a >> 1 = a/2$ $\quad a >> 3 = a/2^3$

$\boxed{a >> 2 = a/2^2 \quad a >> N = a/2^N}$ → no need to worry about overflows.

// Naming convention : → 1 : set or ON , 0 : unset or OFF

1 Byte = 8 bits

	size	Bits	Bit posit
// Byte	= 1 Byte	→ 8	[0 - 7]
short int	= 2 Byte	→ 16	[0 15]
int	= 4 Bytes	→ 32	[0 31]
long long or long	= 8 Bytes	→ 64	[0 63]

Given N integer, check if i^{th} bit pos in N is set or Not?

(bool) → If set return True else return False

$N = 21$:
$$\begin{array}{ccccc} 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{array}$$
$i = 2$ → set

$N = 34$:
$$\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$
$i = 3$ → unset

```
bool checkBit(N, i){
```

[82]

$$\begin{array}{lccccccc} & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ // \; N = & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

$i = 0$: if((N & 1) == 1) { 0^{th} bit in a N is set

$$\begin{array}{lccccccc} & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ N = & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

}

$N >> 4 = 1 \; 0 \; 1$

(N >> 4) & 1 == 1 { 4^{th} bit in N is set }

```
if((N >> i) & 1 == 1) {
    i^th bit in N is set
    return True
}
else { return False }
```

TODO: Try with left shift

Doubts?

$$\left\{ \begin{array}{l} 5 \ll N = \\ a \ll N = a * 2^N \\ \downarrow \\ 5 \ll N = \boxed{5 * 2^N \neq 5^N} \end{array} \right\}$$