BIT MANIPULATION FOUNDATIONS

* Bit Manipulation
→ Foundations
→ Masking
→ Application (5-6 Types)

- Foundations
  &, <<, >>, ~, ^, |, -

Exa :
→ cout << (1<<2+3) <<endl;
output = 32

→ variables (e.g., int x=5) are stored in a memory as a sequence of bits. The binary representation is stored in the form of 32 bits (4 bytes).

$x = 5 = (0101)_2$

0	1	2	3	4	5	...	26	27	28	29	30	31
0	0	0	0	0	0	....	0	0	0	1	0	1

→ Bit manipulation uses operators to work directly on these individual bits rather than using 'x' or its value.

- Bitwise Operators

> X|Y (X or Y)

→ sets a bit to 1 if either of the corresponding bits is 1.

→ Exa : X : 0101
Y : 1001
X|Y → 1101

X	Y	X|Y
0	0	0
0	1	1
1	0	1
1	1	1

> & (AND)

→ sets a bit to 1 only if ~~either~~ both of the corresponding bits are 1.

→ Exa: X : 0101
Y : 1001
X&Y : 0001

X	Y	X&Y
0	0	0
0	1	0
1	0	0
1	1	1

> What is the output of ~x ?

Flip every bit
So, X = 5 = 0101
~X = 1010 = $(-6)_{10}$

$(1010)_2 = (10)_{10}$
But the ans will be calculated on the basis of all of 32 bits.
All 32 bits will flip.

X: | 0 | 0 | 0 | 0 | 0 | 0 | .... | 0 | 0 | 0 | 1 | 0 | 1 |

~X: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | --------- | 1 | 1 | 1 | 0 | 1 | 0 |

> EXCLUSIVE OR

→ sets a bit to 1 only if the corresponding bits are different.

→ An odd no. of 1s results ~~likes~~ in 1. An even no. results in 0.

Exa: 0 1 0 1
 ^ 1 0 0 1
 $(1100)_2 = (12)_{10}$

> shift operators

• left shift : x << y

→ shift the bits of x to the left by y posⁿ. New posⁿ ~~are~~ on the right are filled with 0s.

Exa: 1) 2 << 2 = 0 0 1 0

shift this 1 by 2 ⇒ $2 << 2 = (1000)_2 = (8)_{10}$

2) 5<<3 : shifts the bits of 5 by 3 towards the left.

5 = 0 0 (0 1 0 1)

shift this by 3

$5<<3 = (101000)_2 = (40)_{10}$

• Right shift

(101) >> 2

shift this by 2 towards right

⇒ (001) 01 → vanishes as 32 bits are full

$= (001)_2$

When we do, $x<<y \Rightarrow X * 2^Y$

& $x>>y \Rightarrow \frac{X}{2^Y}$

Q: Give output of the following:

1. cout << 1 << 2+3 << endl;

Binary representation of 1

$(1)_{10} = (0001)_2$

Performing left shift ⇒ 1<< (2+3)

= 1 <<5

00(0001)

shift this 1 by 5 towards left

⇒ $(100000)_2$

Bin→Dec: $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$= (32)_{10}$

Bit operator has lower precedence, so put explicit brackets.

2. $10 >> 2$

$10 = (1010)_2$

$10>>2 = \underset{\hookrightarrow 2}{1010} = (0010)\underset{\hookrightarrow \text{vanishes}}{10} = (0010)_2 = (2)_{10}$

$2^1 2^0$

3. $5 << 4$

$5 = (0101)_2 = 0000(0101)_2 = (1010000)_2 = (80)_{10}$

$4 \leftarrow$

Bin → Dec $2^6 2^5 2^4 2^3 2^2 2^1 2^0$

4. $10 \& 4 | 2$

$10 \& 4$

$10 = (1010)_2$

$4 = (0100)_2$

$10\&4 = (0000)_2$

$2 = (0010)$

$10\&4|2 = (0010)_2 = (2)_{10}$

$(11)_{10} = 00001011$

↳ decides if no. is +ve or -ve.

If first bit 0 → positive no.

If first bit 1 → negative no.

Representation of -11

$-11 = ((\sim 11) + 1)$

$\sim 11 = 11110100$

$\sim 11 + 1 = \quad +1$

$-11 = 11110101$

2's complement

$X = 5 = (0101)$

$\sim X = (1010)$ 1's complement of X

$\downarrow +1$

$= (1011)$ 2's complement of X

$-X =$ 2's complement of X

$-X = ((\sim X) + 1)$

Q: What is the decimal value of 11110000 ?

$X \rightarrow 11110000$

↳ Indicates -ve no.

$\sim X = 00001111$

$+1 \downarrow$

$\sim X + 1 = 00010000$

$-X = -16$

Q: Find the value of $x = 1 << 31$

Initial state : $\underbrace{00000....00001}_{32 \text{ bits}}$

shift 31 places towards left

X : $10000 \cdots 0000$

↑ -ve

$\sim X$: $0111111 \cdots 1111$

$\sim X + 1$: $1000000 \cdots 0000 \Rightarrow 2^{31}$

$-X = \sim X + 1$

$\boxed{X = -2^{31}}$ → Ans

• Masking — it is a technique to compress information.

Application : It allows us to represent sets and subsets using integers (masks) and perform set operation.

Boolean Array: bool arr [ ] = | 0 | 1 | 1 | 0 | 1 | 1 |

Ex : {2, 5, 7, 8}
0 1 2 3

Subset : {2, 5, 8}
Position : {0, 1, 3}

Now we can store this info. in a boolean array where the index with the value of 0, 1, 3 will have 1 and other 0

| 1 | 0 | 1 | 1 | $= (1011)_2$
3 2 1 0

What no. will be store in x if we want to build subset {5, 7} ?

~~{2, 5, 7, 8} Ma~~

{5, 7} = {1, 2}

Boolean Array = | 0 | 1 | 1 | 0 | $= (0110)_2 = (6)_{10}$
3 2 1 0

• Mask

Ex : {2, 5, 7, 8}
0 1 2 3

Mask = 6 = 0110 ⇒ {5, 7} ~~Insert 8 here~~
3 2 1 0
1 (Flip)

① Goal : Inserting 8

→ 8 is at index 3. To add it we need to "flip" the bit at index 3 from 0 to 1.

Original Mask : 0110 → {5, 7}
Desired Mask : 1110 → {5, 7, 8}

This can be done using Bitwise operations.

→ To flip the 3rd bit we can use (1<<3) and take OR b/w (1<<3) and X

∴ X = 0110
↳ 1<<3 → 1000

→
  0110
  1000
OR = 1110
↳ Inserted 8

* If there is a set if 8 is present remove it or else add it

⇒ change the OR operation to XOR (^). It handles both cases of insertion & deletions. It basically flip the ith bit.