**Basic Theory**

-5 is Stored as 2’s Complement in the Hardware

int → 32 bit (4 byte)... First Bit is for “Pos OR Neg” and Rest of 31 Bit is for Numbers

INT\_MAX = 2^31-1 (because there is 0 so we have to Minus “1” in it)

INT\_MIN = 2^31

**the “-1” in the Integer is Because we have “0” is Considered to be Positive. As Such, it is Included in the POSITIVE Side… And as such there is a LESS 1 in the Positive Side, as 0 is in the Positive Side**

* Set union A | B
* Set intersection A & B
* Set subtraction A & ~B
* Set negation ALL\_BITS ^ A or ~A
* Set bit A |= 1 << bit
* Clear bit A &= ~(1 << bit)
* Test bit (A & 1 << bit) != 0
* Extract last bit A&-A or A&~(A-1) or x^(x&(x-1))
* Remove last bit A&(A-1)
* Get all 1-bits ~0

**Left & Right Shift**

Basically,

Left Shift → Multiply by 2 → x << k = x\*(2^k)

Right Shift → Divide by 2 → x >> k = x/(2^k)

**XOR**

XOR → Number of 1’s **EVEN** then **ZERO**

→ Number of 1’s **ODD** then **ONE**

→ for All 0’s then ZERO

XOR (^) of SAME Number = ZERO (caz, Number of 1’s are EVEN)

XOR is mainly used to know if in **TWO Numbers i'th** Bits are **Different or Not**

if,

Two Number **i’th Bit** **COMPARE** korar Jonno XOR use hui, jodi **SAME SAME** bit hui taile nijeder **CANCEL OUT** kore dey that means 0 hoy. But jodi **DIFFERENT DIFFERENT** Bit hui in that case “**1**” Output

**\*\*\*Same Bit huile = 0\*\*\***

**\*\*\*Different Bit huile = 1\*\*\***

**0 xor N = N (any number XOR with 0 is will be that Number)**

**N xor N = 0 (any number XOR with Himself is 0)**

**SWAP Variables** with XOR

a = a xor b

b = a xor b

a = a xor b

**Famous Problems**

**\*\*\***

**(1<<i)** → 1 0 0 → **eivabe “1” ke Prepare korte hobe and then GIVEN Number sathe** (AND, OR, XOR). korte hobe to Set, Unset & Toggle Bit… **(1<<i) is a very POWERFUL Technique**

**\*\*\***

n | (1<<i) → **SET** the i’th Bit

n & ~(1<<i) → **UNSET/CLEAR** the i’th Bit

n ^ (1<<i) → **TOGGLE** the i’th Bit

**(n&1)** → to Check if the Number is **ODD** (if, 1) or **EVEN** (if, 0)

Example:

10 = 1 0 1 0

&

1 = 0 0 0 1

––––––

0 0 0 0

# **Do Remember:**

* **Single “&” is for Bitwise “AND”**
* **Single “|” is for Bitwise “OR”**

# **Ques: Check if the i’th bit is 0 or 1?**

n=13 and i=2

**Trick:** for these types of Questions **→ if any BIT is Set, Unset 1 etc.** Then Try to Think of

**“Left Shift” and “Right Shift”** the **GIVEN Number first** and then **“OR(to Set Bit), AND(to Clear bit), XOR(to Toggle bit)” with “1” to** Know that Particular Bit is Set or Not Set.

OR, Do Reverse Order → Left Shift and Right Shift “1” then Try to Compare with “GIVEN” Number

***Left Shift Method***

Now, 13 is: 1 1 0 1

now, if we Right Shift by 2 → then it will be Only “1”

and Now, if check the Result and “&” with “1” we can Know if the Bit was 1 or 0

**((n>>i) & 1) == 1** , then the i’th bit was “1”

***Right Shift Method (Preferred)***

Now, 13 is: 1 1 0 1

let, if we left shift Single “1” by 2 places → 1 0 0

Now, if we do OR, with 13

1 1 0 1 (n)

&

0 1 0 0 **(1<<i)**

––––––

0 1 0 0

(n & (1<<i)) == 1

**(n & (1<<i)) != 0** → This Means i’th Bit is “1”... “**>**” Because Think, if the 3rd Digit is 1… Then **(n & (1<<i))** will be “4”... So, we have to check **i’th Bit** is “**Greater Than**” or “**Less Than**” Equal “0”

"**!=0**" Because **Numbers** can be **NEGATIVE**

# **Ques: Remove/Unset Right Most bit to 1?**

n-1 = korle amra “Right Most 1” soho baki Right diker shb 0 “Toggle” hoye 1 hoye jabe, karon amra “1” Carry korte korte sheita “Right Most 1” Bit er sathe giye Cancel out hobe

Example:

n = 40 → 1 0 **1** 0 0 0 → here, observe the “Right Most 1 Bit (in Bold)”

n = 39 → 1 0 **0** 1 1 1

to Remove the “Right Most Bit”:

**n & (n-1)**