**Bit Magic**

Java stores integers in 32 bits. First bit is used as signed bit, hence integer range is :

-2^31(2147483648) to 2^31 -1(2147483647)

In Java negative numbers are stored in 2’s compliment.

2^32 -1=111111…..11(32 ones)

Find 2’s compliment of a number: Take 1’s compliment and represent it in 2^32-x form.

e.g. 5 = 0000000….0101

1’s compliment= 1111111111111…1010

=((2^32-1)-5)=2^32-6

2’s compliment of 5 is -6

Bitwise operators in java:

OR: |

AND: &

XOR: ^

Compliment: ~ (2’s compliment in java)

Left Shift: << fill the trailing bit by 0(multiplication by 2 power until int

range is reached, it might become negative)

Signed Right Shift: >> (division by 2 power for positive only) for positive number fill the leading bits by 0, for negative fill by 1. So it preserves the signed bit)

E.g.: -2 >>3

-2 two’s compliment representation: 1111111111……110

Right shift 3 digits and fill leading bits by 1: 11111111…….111

=2^32-1

=-1 in 2’s compliment

Unsigned Right Shift: >>> (division by 2 power for positive and negative both) fill leading with 0’s for both positive and negative numbers. Negative numbers become positive. It does not prevent the signed bit.)

**XOR Properties:**

1. *Commutative*: A ^ B = B ^A
2. *Associative*: A^ ( B ^ C ) = ( A^ B ) ^ C
3. *Identity element*: A ^ 0 = A
4. *Self-inverse*: A ^ A = 0

**Gray Code:** Gray code has property that two successive numbers differ in only one bit. Following is 2-bit sequence (n = 2)

00 01 11 10

n-bit Gray Codes can be generated from list of (n-1)-bit Gray codes using following steps.

1. Let the list of (n-1)-bit Gray codes be L1. Create another list L2 which is reverse of L1.
2. Modify the list L1 by prefixing a ‘0’ in all codes of L1.
3. Modify the list L2 by prefixing a ‘1’ in all codes of L2.
4. Concatenate L1 and L2. The concatenated list is required list of n-bit Gray codes.

**Binary to Gray conversion:**

* The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
* Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.

**Gray to binary conversion:**

* The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given binary number.
* Other bits of the output binary code can be obtained by checking gray code bit at that index. If current gray code bit is 0, then copy previous binary code bit, else copy invert of previous binary code bit.