Competitive Programming

Bit Manipulation

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

Binary Representation of a Number

- In our daily life, we deal with numbers in the decimal system. However, computers deal with numbers in the binary format.
- In the decimal system, all our numbers are represented using the following digits:
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- In the binary system, the numbers are represented using only 2 digits:
- 0, 1
- Any binary number represents the coefficients of the powers of 2.

For Example

- The binary number 11101 actually represents:
- 1 * (2 ^ 4) + 1 * (2 ^ 3) + 1 * (2 ^ 2) + 0 * (2 ^ 1) + 1 * (2 ^ 0)
- = 16 + 8 + 4 + 0 + 1
- = 29
- Usually, the integers we work with a 32 bit integers. Of this, the most significant bit (the first bit on the left), is used to store the sign of the integer (1 for negative, and 0 for positive).
- Thus, the range of a signed integer is usually given as -2^31 + 1 to 2^31 -1

Truth Tables

- In bit manipulation, we shall often deal with the following operations:
  - AND, OR, XOR and NOT.
- For each of these operations, we generally use the bitwise operations.
- This mean that, we apply the operations to each of the corresponding bits of the operands.

AND Truth Table (X = A & B)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Example

- If we apply the same to 5 & 7, we get 5

101

111

___

101

OR Truth Table (X = A | B)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Example

- If we apply the same to 5 | 7, we get 5

```
101
111
___
111
```

XOR Truth Table (X = A ^ B)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Example

- If we apply the same to 5 ^ 2, we get 6

101

011

___

110

Shift Operators

- We often use the left shift << and the right shift >> operators in bit manipulation.
- These operators shift the bits of a number to the left, or the right respectively.
- The most significant bit (or the leftmost bit) in a number is used to denote the sign of the number. When we apply the arithmetic left/right shift operator, this bit is NOT shifted.

Example

- The binary representation of 5 is 101
- Therefore,
  - 5 >> 2 = 001
  - 5 << 2 = 10100

Example

- Note that, left shifting multiplies the number with a power of 2 and right shifting divides the number by a power of 2.
  - a >> b = a / (2 ^ b)
  - a << b = a * (2 ^ b)

Checking / Setting the i^{th} bit

- We use the above, very useful property, to help us in finding the value of the i^{th} bit of the number. And also, to set/unset the i^{th} bit of a number.

Example

- For example, given the number a:
  - 10111100X0001
- We don’t know the value of the 5th bit, hence it is denoted by X.
- Using the shift operators, we can and this value with 1:

```
c = 1 << 5 // now c is 10000
value = a & c
if(value == 0)
   print 0
else
   print 1
```

Example

- Effectively, we are running the following calculation:

```
  10111100X0001
& 0000000010000
_______________
  00000000X0000
_______________
```

- This is because the result of any number AND with 1 is the number itself.
- Now, if the value is 0, this means that X has to be 0. Otherwise, X has to be 1.
- Similarly, when we use the OR operator, we can set the value of any bit to 1. And if we use the AND operator with 0, then we can unset any value to 0.

Queries?

Thank You…!