Bit Manipulation - 1

Truth Tables For Bitwise Operators

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

A & B = 1 → both the bits are set

A | B = 1 → if any bit is set

A ^ B = 1 → if opposite bits then set

Basics And Xor Or Properties

(A) Basics AND Properties

(1) Even / Odd properties

↳ In a Binary Representation.

If LSB → 0 = Even

↳ 1 = Odd

LSB

Ex:- 10 → 1 0 1 0

9 → 1 0 0 1

⇒ A & 1

```
A = x y z a b c
& 1 = 0 0 0 0 0 1
-----------------
      0 0 0 0 0 □
```

← Depends on C

if C is 0 then Ans is 0, else 1

CONCLUSION

↳ (For odd)

if ((A & 1) == 1) & odd No.}

↳ (For even)

if ((A & 1) == 0) & even No.}

⇒ A & 0 = 0

```
A = x y z a b c
& 0 = 0 0 0 0 0 0
-----------------
      0 0 0 0 0 0
```

⇒ A & A = A

```
A = 1 0 1 1 0
& A = 1 0 1 1 0
---------------
      1 0 1 1 0
```

=> Basics of OR properties

=> A | 0 = A

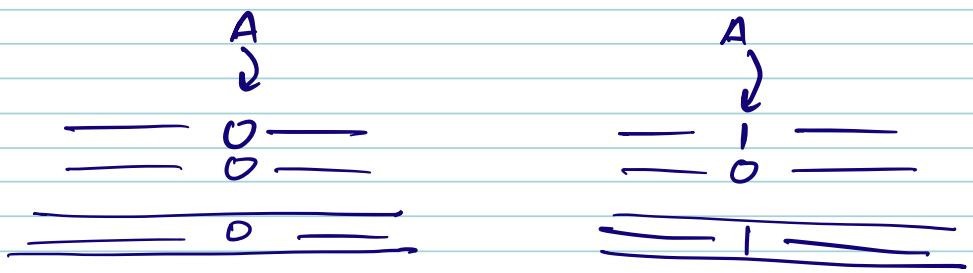
```
A = 1 0 1 1 1
0 = 0 0 0 0 0
-------------
    1 0 1 1 1
```

=> A | A = A

```
A = 1 0 1 1 1
A = 1 0 1 1 1
-------------
    1 0 1 1 1
```

=> Basics of XOR properties

=> A ^ 0 = A




```
Ex:- A = 1 0 1 1 1
     0 = 0 0 0 0 0
---------------------
       1 0 1 1 1
```

$\Rightarrow A \wedge A = 0$

Eg: $A = 1\ 0\ 1\ 1\ 1$

$A = 1\ 0\ 1\ 1\ 1$

$0\ 0\ 0\ 0\ 0$

(D) COMMULATIVE PROPERTY

↳ The orders of Operands doesn't affect the result of bitwise operation.

$A \& B = B \& A$

$A \mid B = B \mid A$

$A \wedge B = B \wedge A$

(E) ASSOCIATIVE PROPERTY

↳ Grouping operands does not affect the Result of the operation.

→ $(A \& B) \& C = A \& (B \& C)$

→ $(A \mid B) \mid C = A \mid (B \mid C)$

→ $(A \wedge B) \wedge C = A \wedge (B \wedge C)$

Note :- operators Needs to be Same.

Quiz 1 :- Evaluate the expression

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

↓ Commutative property

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

↓

$0 \wedge b \wedge b \wedge d$

↓

$b \wedge b \wedge d$

↓

$0 \wedge d$

↓

d ⟸ Ans.

Quiz 2 :- Evaluate the expression.

$1 \wedge 3 \wedge 5 \wedge 3 \wedge 2 \wedge 1 \wedge 5$

↓ Commutative property.

$1 \wedge 1 \wedge 3 \wedge 3 \wedge 5 \wedge 5 \wedge 2$

↓

$0 \wedge 0 \wedge 0 \wedge 2$

↓

2 ⟸ Ans

LEFT SHIFT OPERATOR (<<)

↳ The left shift operator (<<) shifts the bits of a number to the left by a specified No. of positions.

Lets assume we have 8 bits.

Ex: a = 10

* Till the term 0 is waste we are fine but as soon as 1 goes waste it leads to overflow which means Given bits can't store the No. formed with Left shift.

CONCLUSION

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

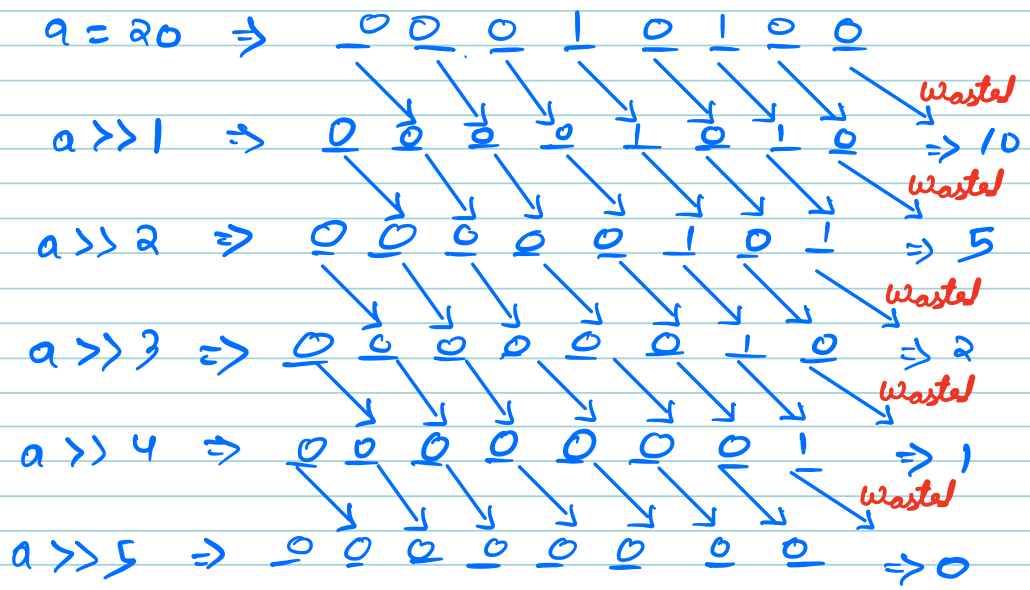
$$1 << n = 2^n$$

RIGHT SHIFT OPERATOR (>>)

↳ The Right shift operator (>>) shifts the bits of a Number to the right by a specified No. of positions.

↳ when we Right Shift a binary Number, the most significant bit (MSB) is filled with 0.

Lets assume there are 8 bits.



CONCLUSION

$$a >> n = \frac{a}{2^n}$$

$$1 >> n = \frac{1}{2^n}$$

Power Of Left Shift Operator

① OR (|) operator

↳ Let Say we want to set 4th Bit

Ex:- N = 1 0 1 1 0 1

OR = 0 1 0 0 0 0 (1<<4)

1 1 1 1 0 1

↳ Let Say we want to set 3rd Bit.

Ex:- N = 1 0 1 1 1 1

OR = 0 0 1 0 0 0 (1<<3)

1 0 1 1 1 1

Conclusion

↳ Left Shift operator can be used with OR to Set ith Set.

↳ If the bit we are trying to Set is already Set, the nothing will happen

EXPRESSION :-

$N = N \mid (1 << i)$

② XOR (^) operator

↳ If i^{th} Bit is 0

Ex! N = 1 0 1 1 0 1
XOR 0 0 1 0 0 0 (1<<4)
1 0 1 1 0 1

↳ If i^{th} Bit is 1

Ex :- N = 1 0 1 1 1 1
XOR 0 0 1 0 0 0 (1<<3)
1 0 0 1 1 1

Observation

↳ Left shift operator can be used with XOR to Flip (Toggle) the i^{th} Bit.

EXPRESSION

↳ N = N ^ (1<<i)

(3) AND (&) operator

Ex:- N = 1 0 1 1 0 1
& 0 1 0 0 0 0 (1<<4)
0 0 0 0 0 0

Ex:- N = 1 0 1 1 1 1

AND 0 0 1 0 0 0 <(1<<3)

0 0 1 0 0 0

Observation

↳ Left Shift operator can be used with AND to check i'th Bit.

EXPRESSION

↳

```
if (N & (1<<i)) == 0) {
        // i'th Bit is unset
} else {
        // i'th Bit is set
}
```

Quiz 3 :- what will we get if we do 1 << 3 ?

1 << 3 ⇒ 2^3 ⇒ 8 Ans

Question 1 :- check whether i^{th} Bit in N is Set or not.

PSEUDO CODE

```
Function checkBit ( N , i ) {
    if ( (N & (1<<i)) == 0 ) {
        return False;
    } else {
        return True;
    }
}
```

TC - O(1)
SC - O(1)

Question 2 :- Given an Integer N, count the total Number of SET bits in N.

Ex:- N = 12 → 1100

Ans = 2

APPROACH :- We can iterate over 32 bit in a integer & can check if the bit is Set or Not.

PSEUDO CODE

```
function CountSetBits ( N) {
    ans = 0 ;
    for (i = 0 ; i < 32 ; i++){
        if ( checkBit (N, i)){
            ans ++;
        }
    }
    return ans;
}
```

TC → O(32) ≈ O(1)

SC → O(1)

APPROACH 2

N = 10 → 1 0 1 0

Let assume there are 8 bits.

```
  N  = 0 0 0 0 1 0 1 0
& 1  = 0 0 0 0 0 0 0 1
-----------------------
       0 0 0 0 0 0 0 0   ← Last Bit is unset
```

```
 N>>1 = 0 0 0 0 0 1 0 1
& 1   = 0 0 0 0 0 0 0 1
-----------------------
       0 0 0 0 0 0 0 1   ← set Bit
```

N >> 2 = 0 0 0 0 0 0 1 0

& 1 = 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 ← unset Bit

N >> 3 = 0 0 0 0 0 0 0 1

& 1 = 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 1 ← Set Bit.

⇒ Stop N >> 4 = 0 0 0 0 0 0 0 0 → 0

& 1 = 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0

IDEA :- one by one Making each bit of N as LSB & then checking using AND.

PSEUDO CODE

```
ans = 0
while (N > 0) {
    if (N & 1) != 0) { // Set Bit
        ans++;
    }
    N = N >> 1;
}
```

$N = 10 \rightarrow 5 \rightarrow 2 \rightarrow 1 \rightarrow 0$

TC →	$O(\log(N))$
SC →	$O(1)$

Quiz 4 :- what is time complexity to count set bits?

$O(\log(N))$

Question 3 :- Unset the i^{th} Bit of the Number N.

Ex : N = 6 → 1 1 0 , i = 2

↓

0 1 0 ≡ 2

OBSERVATIONS

↳ XOR is used to toggle,

0 → 1

1 → 0

↳ APPROACH → check i^{th} Bit

↳ if Set, then use XOR

PSEUDO CODE

```
func  unset ( N , i ) {
      if ( check Bit (N, i )) {
            N = N ^ (1<<i) ;
      }
      return N;
}
```

TC → O(1)
SC → O(1)

Question 4 :- A Group of computer Scientists is working on a project that involves encoding binary Nos. They Need to create a binary Number with a specific pattern for their project. The pattern require A 0's followed by B 1's followed by C 0's. To simplify they need a function which take A, B & C as input & gives Decimal No. corresponding to binary formed.

Ex :- A = 4 , B = 3 , C = 2

(1+C)th

Ans: 0000 1 1 1 00 => 28

(2+C)th (0+C)th

APPROACH :- → we can ignore A, its of No Use
→ set B Bits starting from C distance from LSB.

PSEUDO CODE

```
ans = 0
for (i = 0 ; i < B ; i++){
    ans = setBit (ans, c+i);
}
```

setBit → N = N | (1<<i)

TC — O(B)
SC — O(1)