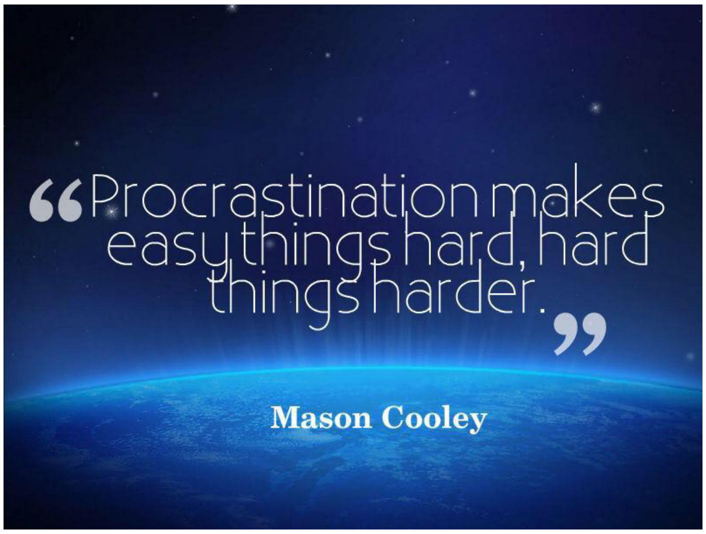
Bits 1



Good Morning

Content

01. Truth Table
02. Basic AND, OR, XOR properties
03. Left & Right shift operator
04. check i^{th} bit
05. Count set bits
06. Unset i^{th} bit
07. set bits in Range

Very Easy

Truth Table

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 = Both A & B has to be set.

A | B = One of the bit has to be set

A ^ B = Both the bits has to be different.

Same same puppy shame

* ## Basic AND Properties

01. A & 1 → 0 (even), 1 (odd)

A = 10

$$
\begin{array}{rrrrrr}
 & A = & 1 & 0 & 1 & 0 \\
\& & 1 = & 0 & 0 & 0 & 1 \\
\hline
 & & 0 & 0 & 0 & 0
\end{array}
$$

A = 11

$$
\begin{array}{rrrrrr}
 & A = & 1 & 0 & 1 & 1 \\
\& & 1 = & 0 & 0 & 0 & 1 \\
\hline
 & & 0 & 0 & 0 & 1
\end{array}
$$

02. A & 0 = 0

03. A & A = A

* Basic OR properties

01. $A \mid 0 = A$

02. $A \mid A = A$

03. $A \mid 1 = \begin{cases} A & \text{(odd)} \\ A+1 & \text{(even)} \end{cases}$

$A = 10$

$$\begin{array}{rcl} & A = & 1\ 0\ 1\ 0 \\ \mid & 1 = & 0\ 0\ 0\ 1 \\ \hline & & 1\ 0\ 1\ 1 = A+1 \end{array}$$

$A = 11$

$$\begin{array}{rcl} & A = & 1\ 0\ 1\ 1 \\ \mid & 1 = & 0\ 0\ 0\ 1 \\ \hline & & 1\ 0\ 1\ 1 = A \end{array}$$

* Basic XOR property

01. $A \wedge 0 = A$

$A = 10$

$$\begin{array}{rcl} & A = & 1\ 0\ 1\ 0 \\ \wedge & 0 = & 0\ 0\ 0\ 0 \\ \hline & & 1\ 0\ 1\ 0 = A \end{array}$$

$A = 11$

$$\begin{array}{rcl} & A = & 1\ 0\ 1\ 1 \\ \wedge & 0 = & 0\ 0\ 0\ 0 \\ \hline & & 1\ 0\ 1\ 1 = A \end{array}$$

02. $A \wedge A = 0$ (V.V.V.I)

03. $A \wedge 1 = \begin{cases} A+1 & \text{(A is even)} \\ A-1 & \text{(A is odd)} \end{cases}$

$A = 10$

$$\begin{array}{r} A = 1\ 0\ 1\ 0 \\ \hat{}\quad 1 = 0\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 1 = A+1 \end{array}$$

$A = 11$

$$\begin{array}{r} A = 1\ 0\ 1\ 1 \\ \hat{}\quad 1 = 0\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 0 = A-1 \end{array}$$

* **Commutative property**

$A \,\&\, B = B \,\&\, A$

$A \mid B = B \mid A$

$A \wedge B = B \wedge A$

* **Associative property**

$A \,\&\, B \,\&\, C \Rightarrow (A \,\&\, B) \,\&\, C = A \,\&\, (B \,\&\, C) = (A \,\&\, C) \,\&\, B$

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

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

Quiz 1

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

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

$\Rightarrow 0 \wedge 0 \wedge d$

$\Rightarrow d$

Quiz 2

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

$$= (1 \wedge 1) \wedge (3 \wedge 3) \wedge (5 \wedge 5) \wedge 2$$

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

$$= 2$$

* Left Shift Operator (<<)

Assume → int = 8 bits

A = 10		0	0	0	0	1	0	1	0	= 10	$= A * 2^0$
A<<1	x	0	0	0	1	0	1	0	0	= 20	$= A * 2^1$
A<<2	x	0	0	1	0	1	0	0	0	= 40	$= A * 2^2$
A<<3	x	0	1	0	1	0	0	0	0	= 80	$= A * 2^3$
A<<4	x	1	0	1	0	0	0	0	0	= 160	$= A * 2^4$
A<<5	x	0	1	0	0	0	0	0	0	= ~~320~~	$= A * 2^5$

Ideally, it should be 320 but we are getting 64.

Overflow

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

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

Right shift operator

A = 10	0	0	0	0	1	0	1	0	$= 10 = 10/2^0$
A >> 1	0	0	0	0	0	1	0	1	× $= 5 = \frac{10}{2^1}$
A >> 2	0	0	0	0	0	0	1	0	× $= 2 = \frac{10}{2^2}$
A >> 3	0	0	0	0	0	0	0	1	× $= 1 = \frac{10}{2^3}$

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

Quiz 3

$1 << 3 = 1 * 2^3 = \underline{8}$

7:49 → 7:59 AM

* Power of left shift operator <u>with AND</u>

A = 45

45	=	1	0	1	1	0	1	
& 1<<2	=	0	0	0	1	0	0	
		0	0	0	1	0	0	⇒ (1<<2)

45	=	1	0	1	1	0	1	
& 1<<4	=	0	1	0	0	0	0	
		0	0	0	0	0	0	= 0

$$
\begin{array}{lcl}
45 & = & 1\ 0\ 1\ 1\ \boxed{0}\ 1 \\
\& & & \\
1<<1 & = & 0\ 0\ 0\ 0\ \boxed{1}\ 0 \\
\hline
 & & 0\ 0\ 0\ 0\ \boxed{0}\ 0 = 0
\end{array}
$$

$$
N \,\&\, (1<<i) \begin{cases} \to 0 & (i^{th} \text{ bit is unset in } N) \\ \to 1 & (i^{th} \text{ bit is set in } N) \end{cases}
$$

* Power of left shift with OR

$$
\begin{array}{lcl}
45 & = & 1\ 0\ 1\ \boxed{1}\ 0\ 1 \\
| & & \\
1<<2 & = & 0\ 0\ 0\ \boxed{1}\ 0\ 0 \\
\hline
 & & 1\ 0\ 1\ \boxed{1}\ 0\ 1
\end{array}
$$

$$
\begin{array}{lcl}
45 & = & 1\ \boxed{0}\ 1\ 1\ 0\ 1 \\
| & & \\
1<<4 & = & 0\ \boxed{1}\ 0\ 0\ 0\ 0 \\
\hline
 & & 1\ \boxed{1}\ 1\ 1\ 0\ 1
\end{array}
$$

$N \,|\, (1<<i) \to$ set the i^{th} bit

$$
\begin{array}{lcl}
45 & = & 1\ 0\ 1\ 1\ \boxed{0}\ 1 \\
| & & \\
1<<1 & = & 0\ 0\ 0\ 0\ \boxed{1}\ 0 \\
\hline
 & & 1\ 0\ 1\ 1\ \boxed{1}\ 1
\end{array}
$$

* Left shift operator with XOR

```
45   =  1 0 1 1 0 1
^
1<<2 =  0 0 0 1 0 0
-------------------
        1 0 1 0 0 1
```

```
45   =  1 0 1 1 0 1
^
1<<4 =  0 1 0 0 0 0
-------------------
        1 1 1 1 0 1
```

N^(1<<i) → toggle the i bit

```
45   =  1 0 1 1 0 1
^
1<<1 =  0 0 0 0 1 0
-------------------
        1 0 1 1 1 1
```

* Check whether i bit is set or not.

```
boolean checkbit(N, i){
    if((N & (1<<i)) > 0){
        return true;
    }
    else{
        return false;
    }
}
```

TC: O(1)
SC: O(1)

Q2 Given an integer N, count the total no. of set bits in N.

N = 10 = 1010   Ans = 2

N = 13 = 1101   Ans = 3

$2^i \leq n$

```
ans = 0
for ( int i=0 ; i<32; i++) {
    if ( checkbit (N, i) == true ){
        ans++;
    }
}
return ans;
```

TC: O(32)

Issue

01. Hard coding it for 32 bits integer
02. Code will run for 32 times irrespective of the no.

$1 = \underbrace{0\,0\,0\,0\,0\ldots0}\,\cancel{1}$

for 30 unset bits, the loop run

ans = 1

* 2^{nd} idea

$$N \& 1 \begin{cases} \rightarrow 0 \text{ (last bit is 0)} \\ \rightarrow 1 \text{ (last bit is 1)} \end{cases}$$

		count
N=10	1010 & 1 ⇒ 0	0
N = N>>1	0101 & 1 = 1	1
N = N>>1	0010 & 1 = 0	1
N = N>>1	0001 & 1 = 1	2
N = N>>1		

N = 0000 ⟶ stop

```
ans= 0
while (n>0)
{
    if ((n&1)==1) ans= ans+1
    n= n>>1;          // n= n/2
}
return ans;
```

TC: $O(\log N)$

SC: $O(1)$

* Unset the i^{th} bit of a no., if it is set.

$$N = \overset{3}{1}\,\overset{2}{0}\,\overset{1}{1}\,\overset{0}{0}$$

$i = 3$

Ans= 0010 = 2

```
if (checkbit(N, i) == true){
    N = (N ^ (1<<i));
}
```

TC : O(1)
SC : O(1)

Set Bits in Range

A group of computer scientists is working on a project that involves encoding binary numbers. They need to create a binary number with a specific pattern for their project. The pattern requires A 0's followed by B 1's followed by C 0's. To simplify the process, they need a function that takes A, B, and C as inputs and returns the decimal value of the resulting binary number. Can you help them by writing a function that can solve this problem efficiently?

Constraints:

```
0 <= A, B, C <= 20
```

Example:

```
A = 4
B = 3
C = 2
```

$\Rightarrow$ 000011100 $\rightarrow$ $\underline{\underline{28}}$

A = 1
B = 6
C = 3

9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	0	0

A = 4
B = 3
C = 2

8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0

Start from C^{th} index $\rightarrow$ B+C−1

O(B)

```
long ans = 0
for (i=C ; i ≤ B+C-1 ; i++){
    ans = ans | (1<<i);          // ans = ans + 2^i
}
```

HW

* We can solve this question in **O(1)** as well.

Doubts

4 3 6 7 3 2

← ← ← ←

←

- 1 2 3

- 1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

n = 321

Ans = 123