Bit - Manipulation

Kinds of Bits?

0 : False / unset bit

1 : True / set bit

Least Significant Bit

LSB :

MSB : Most Significant Bit

3 2 1 0

8 : 1 0 0 0

11 : 1 0 1 1 => 1 0 0 1 => $\boxed{9}$

3 2 1 0

5 => 1 0 1 => 3 bits

0 0 0 0 0 1 0 1

unset the 1st bit in 11,

index

Convert integer to binary number?

11 : 1 0 1 1 => $\boxed{4 \text{ bits}}$

2nd 1st 0

11%2 => $\frac{11}{2} = 5$ => $\frac{5}{2} = 2$ -> $\frac{2}{2} = 1$ -> $\frac{1}{2} = 0$

N = 11

$\boxed{1 \mid 1 \mid 0 \mid 1}$

Reverse

```
vector<int> bits;
  while( n > 0) {
    bits.push( n%2);
    n = n/2;
  }
return reverse(bits)
```

1 0 1 1

11 => 1 0 1 1

3 2 1 0

(11)%2)

$\Rightarrow N = N/2 =$ $\boxed{N = 5}$

$N = 2$

$N = 1$

Convert Binary to Integer

0 1 1 0 1 1

5 4 3 2 1 0

$= 2^0 \times 1 + 2^1 \times 1 + 2^2 \times 0 + 2^3 \times 1 + 2^4 \times 1 + 2^5 \times 0$

$= 1 + 2 + 0 + 8 + 16 + 0 = \boxed{27}$

0 1 1 0 0 1 0

6 5 4 3 2 1 0

$\Rightarrow \boxed{2^4} + 2^1 \times 1$

$= 16 + 2 = 18$

$= 18 + 2^5 = 18 + 32 = 50$

Question: No. of bits required to store a number N.

_ _ ⇒

0 0 → 0
0 1 → 1
1 0 → 2
1 1 → 3

No. of bits	Max Value	
1	1	$\Rightarrow (2^1 - 1)$
2	3	$\Rightarrow (2^2 - 1)$
3	7	$\Rightarrow (2^3 - 1)$
4	15	$\Rightarrow (2^4 - 1)$
$\boxed{K}$		$\Rightarrow 2^K - 1$

least/minimum K such that

-) Find the least

$$\boxed{N \le 2^K - 1}$$ ✓

$$N+1 \le 2^K$$

$$\log_2(N+1) \le K \log_2 2$$

$$\boxed{K \ge \log_2 (N+1).}$$

$\log_2 4 = 2$

$\log_2 9 = 3$

N = 0	1
N = 1	1
N = 2	2
N = 3	2
4	3
5	3
6	3
7	3
8	4
9	4
10	4
11	4
12	4

N = 5.

$K \ge \log_2 6 = \lceil 2.\cancel{x}\cancel{x} \rceil$

$\longrightarrow$ N = 7

$K \ge \log_2 8 \Rightarrow \boxed{K \ge 3}$

We need $\text{ceil}(\log_2(n+1))$ bits.

$$\boxed{T.C: O(\log n) \text{ bits.}}$$

---

int $x = 10;$

4 bytes => 32 bits

B1	0000
B2	0000
B3	0000
B4	00 1010

$$\underbrace{000000001010}_{32 \text{ bits}}$$

Since, computers understands bits very well, all the bitwise operations are very fast

Bitwise Operators

1) AND &
2) OR |
3) XOR ^
4) NOT ~

Truth table

A	B	A&B	A|B	A^B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

If all bits are set, then result is set

If atleast 1 bit is set, the result is set.

If we have odd no. of set bits, then result is set.

$\boxed{1 \wedge 1 \wedge 0} \Rightarrow$

$(1 \wedge 1) \wedge 0$

$0 \wedge 0 \Rightarrow \boxed{0}$

$0 \wedge 1 \wedge 0 \Rightarrow$

$1 \wedge 0 = 1$

$1 \wedge 1 \wedge 0$

$(1 \wedge 0) \wedge 1$

+, ~~-~~

C++ : 4 byte

Java : 4 byte

(&, |, ^) (>, +, /)

`int a = 5 & 8;`

0 0 1 1 0 1

0 0 1 0 1 1

$$\begin{array}{r} 000101 \\ 001000 \\ \hline 000000 \end{array} \Rightarrow 0$$

001001 ↓ $\boxed{9}$

$\boxed{CSO}$

A = 000000 101 . . . . }
B = 0 00101 . . . . . } 32 bits

A & B = $\boxed{1 \text{ clock cycle}}$ (Instruction)

$\boxed{1 \text{ operation}}$

More Operators

1) Left Shift : <<

$= (2 \times 10^9) \times 2^3 \approx \boxed{16 \times 10^9}$

1 0 0 0 0 0 0 0 .

5 << 2

5 => 0 0 0 0 1 0 1

$$\begin{array}{ccccccc} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}$$

$2^4 + 2^2 = \boxed{20}$

$\Rightarrow \underline{5} \times 2^2 = 5 \times 4 = 20$

$x << (i) \Rightarrow \boxed{x \times 2^i}$

2) Right Shift : => the last 2 bits will get lost.

5 >> (2)

$$\begin{array}{cccccc} 0 & 0 & 0 & 1 & 0 & 1 \\ 5 & 4 & 3 & 2 & 1 & 0 \end{array}$$

=> 0 0 0 0 1

$5 \quad \frac{5}{\ } = \boxed{1}$

$\boxed{x >> i \Rightarrow \frac{x}{2^i}}$ $\Rightarrow$ 2^2 4

=> For odd numbers : LSB

5 : 010(1) →

9 : 0100(1) →

LSB : Always 1

For even numbers: LSB => 0

Question : check if a number is odd?

Approach 1: if (N%2 == 1) return true..

Approach 2 :

```
  100.....10111
& 0000000    1
-----------------
  0000000    1 → odd
             0 → even
```

Faster

```
if(N & 1 == 1) return true;
else return false;
```

$\boxed{\begin{array}{l} N = N >> 1; \\ (N \& 1) \end{array}}$

Question : Given an integer, check if ith bit is set or not.

i = 3 => False

True

N = 18

i = 4 =>

1 0 0 1 0
4 3 2 1 0

1 0 0 1 0
1 0 0 1 1

Approach 1:

i = 2

(i = 2)

N = 1 0 0 1 0
4 3 2 1 0

2)

LSB

$\boxed{N >> i}$

$\boxed{\text{Right shift}}$

N >> 2 = 0 0 0 1 0 0

if(((N >> i) & 1) == 1)
SET

else
UNSET

Approach 2.

i = 2

N = 18

(1 << i)

0 1 2 3 4
N = 1 0 0 1 0
& 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0

①

N = 1 0 0 1 0
1 0 0 0 0
1 0 0 0 0

i = 2

N = 0 0 0 0 0 1 => 0 0 0 1 0 0 => 4

$\boxed{1 << 2}$

=1

1 > 0

if((N & (1 << i)) == (1 << i))
SET

$(N \& 2^i)$

$((1<<i) \Rightarrow 2^i)$

$(1<<i)$ 2^i

Little Faster

else

UNSET

$\boxed{T.C: O(1)}$

Question: Check if a number is a power of 2.

N = 8 True

N = 15 False

N = 32 True.

1100

WRONG APPROACH:

If LSB = 0 ⇒ Power of 2

LSB = 1 ⇒ Not a Power

$\overset{4}{1}\ \overset{3}{0}\ \overset{2}{1}\ \overset{1}{0}\ \overset{0}{0} \Rightarrow \boxed{20}$ ✗

0

⇒ 16 ⇒ 0 0 0 0 0 0 0 0 1 0 0 0

MSB: 0

Approach 1:

No. of set bits ⇒ 1

4: 100

8: 1000

16: 10000

32: 100000.

$\boxed{16}$

N = 16

&1

```
set_bits = 0;
while (N > 0) {
    if (N & 1)
        set_bits ++;
    N = N >> 1;
}
if (set_bits == 1)
    TRUE;
else
    FALSE
```

if(set_bits > 1)
return false

N = 0010000

N = N >> 1

N = 001000

N = N >> 1

N = 00100

N = 0010

N = 0001

N = 00000

(32 times)

(No of bits in the number)

O(32) → O(64)

T.C : O(log n)

Approach 2:

001011

12 : 001100

11 : 001011

10 : 001010

9 : 001001

8 : 001000

7 : 000111

6 : 000110

5 : 000101

12 and 11

(n and n-1)

11 & 10

001010

001001

000111

Claim: Subtracting 1 from a number flips all the bits after the rightmost set bit including it.

① N = 00....101001:....0100

N-1 = 00....101001.....0011

00....101001.....0000

② N & N-1

$\boxed{N \& (N-1)}$ => Unset the Right most set bit $x \& x = x$

n = 1 0 1 1 1

n-1 = 1 0 1 1 0

n_1 = 1 0 1 1 0

$n_1 - 1$ = 1 0 1 0 1

$(n_1) \& (n_1 - 1)$ = 1 0 1 0 0

If N is a power of 2

! (N & (N-1)) == 0)

return True;

else false;

N = 000...100.....0

N & (N-1) = 0000000000

$\boxed{T.C: O(1)}$

Question: No. of set bits in a number.

Approach1: O(log n)

N =

Approach2:

N = 0 1 0 0 0 0 0 1 0

N1 = (N&(N-1)) = 0 1 0 0 0 0 0 0 0

N & (N1-1) = 0

N = 0

```
set_bits = 0;
while ( N > 0) {
        set_bits++;
        N =   N & (N-1);
}
```

T.C: O(# set bits)

worst case: O(log n)

Brian Kernighan's Algo

N = 0 0 0 1 0 0 0 1 0 0 0 0 1 0

set_bits = 1

N = N & (N-1)

N = 0 0 0 1 0 0 1 0 0 0 0 0 0

set_bits = 2

N = N&(N-1)

integer array,

Question : Given twice except every number is repeated this no. without 1 numbers. Find extra space.

A = 1 1 2 2 3 5 5 7 7

Naive Approach :

→ Hash map ⇒ O(n)

→ Return number with frequency 1 : →

T.C : O(n)

S.C : O(n)

Approach 2 : Sorting

A = 1 1 2 2 3 3 5 7 7

Find 'i' such that $a[i-1] \,!= a[i]$ && $a[i] \,!= a[i+1]$

T.C : O(n log n)

S.C : O(1)

Approach 3 :

XOR all elements

$x \wedge x = 0$

$$\begin{array}{r} 1\,1\,0\,0\,1\,0 \\ 1\,1\,0\,0\,1\,0 \\ \hline 0\,0\,0\,0\,0\,0 \end{array}$$

A = [2, 7, 1, 2, 1, 7, 5]

$= 2 \wedge 7 \wedge 1 \wedge 2 \wedge 1 \wedge 7 \wedge 5.$

$= 0 \wedge 5 \Rightarrow \boxed{5}$

```
ans = 0;
for(i=0; i<n; i++){
    ans = ans ^ a[i];
}
return ans
```

T.C: O(n)

S.C: O(1)

Question: Given an integer array, every number appears thrice except 1 number.

-> Find that number without extra space.

A = 1 3 12 1 12 4 3 12 1 3

Ans = 4

A = 4 4 2 4 7 7 7

Ans = 2

ans = 0

ans = ans ^ a[i]

Solution:

$\Rightarrow \underbrace{4 \wedge 4}_{0} \wedge 2 \wedge 4 \wedge 7 \wedge \underbrace{7 \wedge 7}_{0}$

=> 2 4 7

Hint: Look at the Binary Representations

No of 0's => 4
No. of 1's => 3

7 7 7   3 3 3

A = 4 4 4 2

	2	1	0
4 :	1	0	0
4 :	1	0	0
4 :	1	0	0
2 :	0	1	0
7 :	1	1	1
7 :	1	1	1
7 :	1	1	1

6

3

3y+1

3y+1

4
↓
3y+1

0,0,0, 1,1,1, 1,1,1
0's => 3
1's => 6

0's = 3x → x is any integer
1's = 3y

0th bit = 3

0 0 1 0
3 2 1 0

Case 1: ith bit of number is 0

No. of set bits (1's) = 3y → Multiple of 3

Case 2: ith bit of number = 1

No. of 1's = 3y+1

3   0' 1st 2nd

Ex

	2	1	0
4	1	0	0
4	1	0	0
4	1	0	0
6	1	1	0
7	1	1	1
7	1	1	1
7	1	1	1

ans = 0 0 0
2 1 0

3

log(max(A))

i = 0   set bits = 3

i = 1 , set_bits = 4

ans = 0 0 0
OR ← 0 1 0 ← (1<<i)
ans = 0 1 0

2^i

i = 2. set_bits = 7

(set_bits) % 3 == 0)

if(set_bits % 3 == 1)

ans = 0 1 0

1 0 0 ← (1<<2)

ans 1 1 0

```
ans = 0;
for(i = 0; i <= 31; i++){
    count = 0;
    for(j = 0; j < n; j++){
        if((A[j] >> i) & 1 == 1)
            count++;
    }
    if(count % 3 == 1){
        ans = ans | 1<<i;
                  or
              ans + (1<<i)
    }
}
```

32

T.C: O(log(MAX(A)) × N)

i = 4

$$\begin{array}{ccccccccc} & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ A[3] = & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}$$

(x >> 4) & 1

⇒ (0 0 0 0 1 1)
& 1
①

T.C: O(N. log(MAX));

S.C: O(1)

⇒

All numbers repeat 5 (any N) times except one number

1s = 5x or 5x+1