Bit - Manipulation

- AND Operators,

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

- & 1 with any bits, bits remain same.

- OR Operators,

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

- || with any bits, bits becomes 1.

X-OR (if and only if)

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

- $a \wedge 1 = \bar{a}$

  $a \wedge 0 = a$

  $a \wedge \bar{a} = 0$

- Complement ($\sim$)

  $\sim a = \bar{a}$

Left shift operator (<<)

$10 << 1$

$a << 1 = 2a$

$a << b = a * 2^b$

Right shift operator (>>)

$a >> 1 = \frac{a}{2}$

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

Odd - Even

- 10011 — Leaving this, every other is a power of 2

$$\underbrace{1\times2^4+0\times2^3+0\times2^2+1\times2^1}_{\text{Even.}}+\underbrace{1\times2^0}_{\text{this is odd}}$$

- If 2^0 place == 1 — odd

  2^0 place == 0 — Even

- odd & 1 = 1

  Even & 1 = 0

⊞ Use X-OR we can find unique elements.
cause $a \wedge a = 0$

⊞ Find i^{th} bit of a number:

Ex: 10110110 → we want 5th bit

```
  10110110   ← num
& 00010000   ← mask
----------
  00010000
```

⊛ If we want (n-1) zeros after 1 then we can do $\boxed{1 << (n-1) = m}$

⊛ Then, (num & m) → 10000

⊛ To get ans we have right shift,

$10000 >> (n-1) = 1$ (Ans)

$\therefore \boxed{Ans = (num \;\&\; m) >> (n-1)}$

⊞ Set the i^{th} bit into 1

Ex: 10110110 → we want 4th bit 1

```
  10110110  → num
(OR) 00001000  → m
-------------
  10111110
```

$m = 1 << (n-1)$ [For bring zeros after 1]

Ans = num | m

⊞ Set the i^{th} bit into 0.

Ex: 10110110 → we want 5^{th} bit 0

```
  10110110  ← num
& 11101111  ← m
-------------
  10100110
```

$m = \sim (1 << (n-1))$

Ans = num & m

Reset ith bit:

1. First Find ith bit.
2. If it's 1 convert into 0
3. If it's 0 convert into 1

Find the position of right most set bit (1).

Ex: 10110110 → Ans → 2

a 1 b

N = 10110110

$N = a\,1\,b$

$-N = \bar{a}\,1\,b$

$-N = 01001010$

$N \;\&\; -N = 00000010$ → 2

$$Ans = (N \;\&\; -N)$$

Negative of a number in Binary form:

1 byte = 8 bits

10 → | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

MSB ↓ (sign bit)

LSB ↓

sign bit — 1 → (-ve)
0 → (+ve)

2's complement,

$-10 = 1111\,0110$

why this:

$1000 = 111 + 1$

$8 = 7 + 1$

$10000 = 1111 + 1$

$16 = 15 + 1$

we know,

$0 - N = -N$

$$\begin{array}{r} 00000000 \\ (-)\ 00001010 \\ \hline \end{array}$$

then,

$$100000000 = 11111111+1$$

Now,

$$11111111+1 - 00001010$$

$$= (11111111 - 00001010) + 1$$

(complement +1)

$$= 11110101+1$$

$$= 11110110$$

Range of Numbers,

1. 1 byte totally 256 numbers can store

1 byte = 8 bits ; , total no. = 2^8

actually no. store in (n-1) bits

So, for 1 byte no. store in 7 bits.

$2^7 = 128$

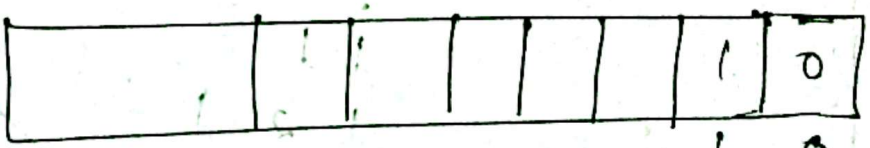
∴ range of no : -128 to 127

∴ Range for n bits;

$$-2^{n-1} \text{ to } 2^{n-1} - 1$$

In array every numbers appears p times and only one numbers appears 1 time. Find the numbers..

Ex: arr = [2, 2, 3, 2, 7, 7, 8, 7, 8, 8]



1 0 → 2

1 0 → 2

1 1 → 3

1 0 → 2

1 1 1 → 7

1 1 1 → 7

1 0 0 0 → 8

1 1 1 → 7

1 0 0 0 → 8

1 0 0 0 → 8

total bits arr → | 3 | 3 | 7 | 4 |

-5> total bits : array each element % repeation time (p)

| 0 | 0 | 1 | 1 |

convert it into decimal → 3 (Ans)

No. of digits in base b.

$$\text{no. of digits} = \text{int}\left(\log_b^n\right) + 1$$

Pascal Triangle,

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5   10  10  5   1
```

Find the sum of n^{th} row.

Sum of each row $= {}^nC_0 + {}^nC_1 + {}^nC_2 + \cdots + {}^nC_n = 2^n$

For n^{th} row, $\boxed{\text{sum} = 2^{n-1}}$

$$\boxed{1 \times 2^{n-1} = 1 << n-1}$$

Power of 2 or not:

→ A number is power of 2 if it has 1 in only 1-bit, rest of has to be zero.

$(100000)_2$ → is a power of 2

$10000000 = 11111111 + 1$

$$\begin{array}{l} 10000000 \rightarrow n \\ \&\ 01111111 \rightarrow n-1 \\ \hline 0 \end{array}$$

$$\begin{array}{l} 10010 \rightarrow n \\ \&\ 01111 \rightarrow n-1 \\ \hline 00010 \end{array}$$

So, we get if $n \,\&\, (n-1) == 0$ then it's power of 2.

Calculate a^b

$3^6 = 3*3*3*3*3*3$ → O(b)

$3^6 = 3^{(110)_2} = 3^4 * 3^2 * 3^0$

ans = 1

base = 3

n = 110

n & 1 == 0 (ignore)

n & 1 == 1 }

ans *= base;

base *= base

n = n >> 1

Given a number n find the no. of set bits in it.

→ $n \,\&\, (-n)$ → rightmost set bits.

→ $n = 9 = (1001)_2$ → Ans = 2

way one

```
while(n>0){
    ans++
    n-=(n & -n)
}
```

$n \,\&\, (-n)$ =

another:

```
while (n>0) {
    ans++
    n = n & (n-1)
}
```

c=1 { n = 1001, n-1 = 1000, n & (n-1) = 1000 }

c=2 { n = 1000, n-1 = 111, n & (n-1) = 0000 }

no. of sets bits = no. of iterations

Find X-OR of numbers from 0 to 9

a	X-OR from 0 to 9
0	0
1	1
2	3
3	0
4	4
5	1
6	7
7	0

If, $a \text{ X.OR } 4 = 0$ $\xrightarrow{\text{X.OR } 0 \to a}$ a

$a \text{ X.OR } 4 = 1$ 1

$a \text{ X.OR } 4 = 2$ a+1

$a \text{ X.OR } 4 = 3$ 0

⊛X-OR from a to b

a = 3 b = 9

3^4^5^6^7^8^9

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

↑ this extra

↳ to removing this we have to do X-OR again with extra (0 to a-1)

Sieve of Eratosthenes: Get prime numbers in range.

2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~
11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~
~~21~~, ~~22~~, 23, ~~24~~, ~~25~~, ~~26~~, ~~27~~, ~~28~~, 29, ~~30~~
31, ~~32~~, ~~33~~, ~~34~~, ~~35~~, ~~36~~, 37, ~~38~~, ~~39~~, ~~40~~

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37

Time complexity:

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \frac{n}{5} + \cdots$$

$$= n\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots\right)$$

$$= \log(\log n)$$

Total Time Complexity: $O(n * \log(\log n))$

Finding square root of a number:

* Binary search

$\sqrt{36}$, s = 0 , e = 36

mid = (s+e)/2

mid * mid > n → e = mid - 1

mid * mid < n → s = mid + 1

mid * mid == n → return mid

we can add precision,

ans = 0.0

ans + = mid incr = 0.1

keep increasing ans + = incr until

ans * ans <= n

ans - = incr

and incr /= 10

- At first assign x to N
- then minimize error
- update x

Time Complexity: $O(\log N * f(N))$

$f(N)$ → complexity of calculating $\frac{f(x)}{f'(x)}$ with a integer precision

Square root using Newton Raphson method

$$sqrt(N) = \left(x + \frac{N}{x}\right)/2$$

x → Assuming sqrt.

If my guess was actual ans: then

$$sqrt(N) = \left(x + \frac{N}{x}\right)/2$$

$$= \left(\sqrt{N} + \frac{N}{\sqrt{N}}\right)/2.$$

$$= \left(\frac{N+N}{\sqrt{N}}\right)/2$$

$$= \frac{N}{\sqrt{N}}$$

$$= \sqrt{N}$$

trying to minimize the error as much as possible.

$$error = |sqrt - x|$$