// Content:

- → Number system basics
- → Binary ⇄ Decimal
- → String -ve numbers
- → MSB & ranges
- → Bitwise operators & properties
- → CheckBit / CountSetBit
- → Unique Element
- → Triple Trouble.

// Maths:

Sum of N Terms in G.P

→ $\frac{a*(r^N-1)}{r-1}$

↳ G.P ⇒

$2^0+2^1+2^2+2^3+2^4 = 2^5-1 < 2^5$

N = 5, a = 1, r = 2 ⇒ $\frac{1*(2^5-1)}{2-1}$ ⇒ 2^5-1

- ⇒ a - 1st Term
- ⇒ N - No: of terms
- ⇒ r - comm ratio

⇒ $2^0+2^1+2^2+\dots 2^6 = 2^7-1 < 2^7$

N = 7, r = 2, a = 1

Number system → Decimal
Base 10
↳ Digits: [0-9]
↳ powers: powers of 10

8 4 2 1 5 = $8*10^4 + 4*10^3 + 2*10^2 + 1*10^1 + 5*10^0$

$10^4\ 10^3\ 10^2\ 10^1\ 10^0$

// Binary Number System:
base: 2
↳ digits: [0-1]
↳ power: power of 2

Eg1:

5 4 3 2 1 0
1 0 1 1 0 1
$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

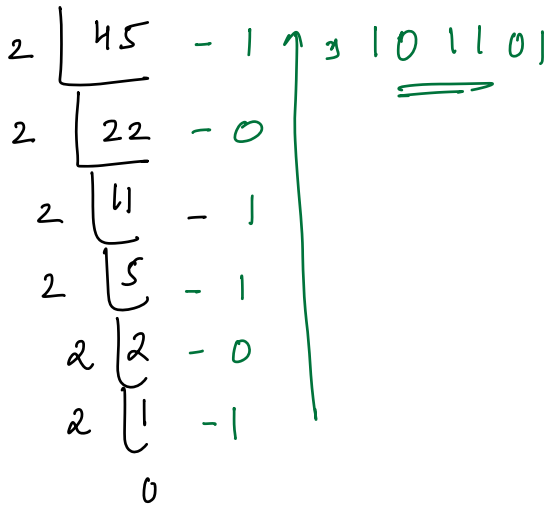
→ Decimal Number?
→ $1*2^5 + 1*2^3 + 1*2^2 + 1*2^0$
→ 32 + 8 + 4 + 1 → 45

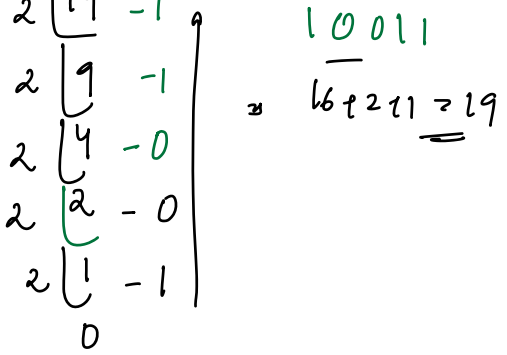
Eg2:

4 3 2 1 0
1 0 1 1 0
$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

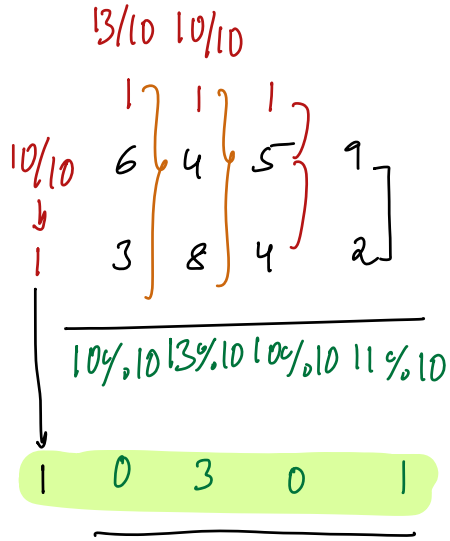
→ $1*2^4 + 1*2^2 + 1*2^1$
→ 16 + 4 + 2 → 22

// Decimal → Binary



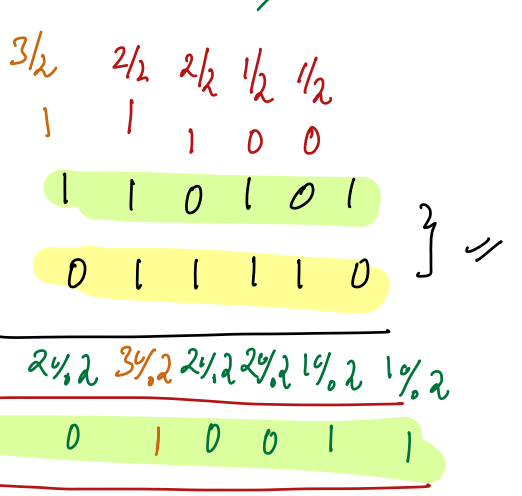


// Add 2 Decimal numbers ? => d = sum %10 , c = sum /10



Add a Binary numbers

d = sum %2 , c = sum /2



8 bit ⇒ 8 bits

10 : 0 0 0 0 1 0 1 0

-10 : 1 0 0 0 1 0 1 0

Sign bit ?

4 : 0 0 0 0 0 1 0 0

-4 : 1 0 0 0 0 1 0 0

1's))))))) 1

10 : 0 0 0 0 1 0 1 0

6 = 1 0 0 0 1 1 1 0

-ve ⇒ -14

0 : 0 0 0 0 0 0 0 0

-0 : 1 0 0 0 0 0 0 0

// 2 same values are having

2 different Binary representation.

$-a$ = 2's of a = is a + 1

↳ 1 ⇄ 0

10 : 0 0 0 0 1 0 1 0

is 10 : 1 1 1 1 0 1 0 1

+1 : 0 0 0 0 0 0 0 1

-10 : 1 1 1 1 0 1 1 0

-2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

7 6 5 4 3 2 1 0

-2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

↳ → left most bit

MSB → Most Significa Bit

MSB, bak valu ge negative → =

→ $2^7 + 2^6 + 2^5 + 2^4 + 2^2 + 2^1 = 246$

→ $-2^7 + 2^6 + 2^5 + 2^4 + 2^2 + 2^1 = -128 + 118 = -10$

// 4 bit number

3 2 1 0

-2^3 2^2 2^1 2^0

// N bit Number ?

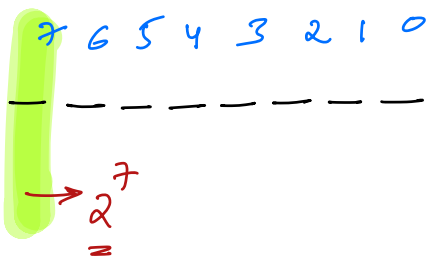
N-1 . . . 0

-2^{N-1}

// Is MSB always negative?

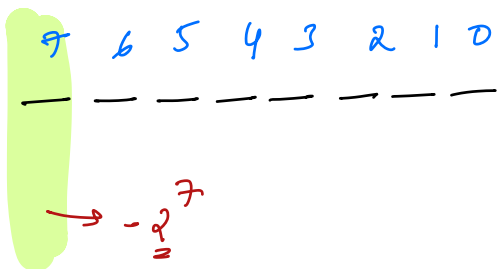
MSB bare value +ve ← unsigned

unsigned 8 bit number



signed ⇒ MSB bare val is -ve

signed 8 bit number



// How will system signed / unsigned

- byte n
- short n
- int n
- long n

Default it is signed.

- unsigned int n
- unsigned long n
- unsigned short int

⇒ MSB bare value is +ve

Only in C/C++

In other language MSB bare value = -ve

// Ranges

unsigned | signed

Min | Max | Min | Max

4 bit : 1 1 1 1 → $[0, 2^4 - 1]$ | 0 1 1 1 $[-8, 7]$

$2^3\ 2^2\ 2^1\ 2^0$ | -2^3 $2^2\ 2^1\ 2^0$

N bit : 1 1 1 1 1 1 → $[0, 2^N - 1]$

2^{N-1} ... $2^1\ 2^0$

0 1 1 1 1 1 1 1

$2^2\ 2^1\ 2^0$

2^{N-2}

-2^{N-1}

$[-2^{N-1}, 2^{N-1} - 1]$

$2^0 + 2^1 + 2^2 \cdots 2^{N-2}$

$a = 1$, $r = 2$, $N = N-1$ → $\frac{2^{N-1} - 1}{2 - 1}$ → $2^{N-1} - 1$

// n bit signed : $[-2^{N-1}, 2^{N-1} - 1]$ → $[-128, 127]$ → $[-2^7, 2^7 - 1]$

// n bit unsigned : $[0, 2^N - 1]$ → $[0, 2^8 - 1]$ → $[0, 255]$

// byte : 1 Byte = 8 bits : int → 4 Bytes = 32 bits

short int : 2 Byte = 16 bits

long / long long = 8 B = 64 bits.

// int : → 4B ⇒ 32 bits. : $[-2^{31}, 2^{31}-1]$ ≈ $\{-2*10^9, 2*10^9\}$

$2^{10} = 1024 \approx 1000 \approx 10^3$

$2^{10} \approx 10^3$ → cube on both sides

$2^{30} \approx 10^9$ } Multiply by 2 on both sides

$\boxed{2^{31} \approx 2*10^9}$

⇒ Bit wise operators : ({ &, |, ^, ~ } <<, >>)

Truth Table

Same Same puppy Shame

ps

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 = 29 : 0 0 0 1 1 1 0 1

b = 18 : 0 0 0 1 0 0 1 0

a & b : 0 0 0 1 0 0 0 0 → 16

a | b : 0 0 0 1 1 1 1 1 → 31

a ^ b : 0 0 0 1 1 1 1 1 → 15

~a : 1 1 1 0 0 0 1 0 → ? → -a-1

-30

2's a = -a = 1's a + 1 = ~a + 1

-a = ~a + 1

~a = -a-1

a & b = b & a

a ^ b = b ^ a

a | b = b | a

a & b & c = a & c & b = b & c & a

a ^ b ^ c = a ^ c ^ b = b ^ c ^ a

a | b | c = a | c | b = b | c | a

set = 1

unset = 0

// property

a : 10

a : 1 0 1 0

1 : 0 0 0 1

a&1 : 0 0 0 0

a : 11

a : 1 0 1 1

1 : 0 0 0 1

a&1 : 0 0 0 1

if (a&1 == 1) {

a is odd

0th bit is set

}

else {

a is even

0th bit is unset

}

// a & a = a | a ^ a = 0 | a | a = a

a & 0 = 0 | a ^ 0 = a | a | 0 = a

→ 10 : 30 break

→

// << >>

7 6 5 4 3 2 1 0

byte a = (5) : 0 0 0 0 0 1 0 1 : $2^2+2^0 \Rightarrow 5$

remove

a<<1 : 0 0 0 0 1 0 1 0 : $2^3+2^1 \Rightarrow 10 \Rightarrow 5*2^1$

a<<2 : 0 0 0 1 0 1 0 0 : $2^4+2^2 \Rightarrow 20 \Rightarrow 5*2^2$

a<<3 : 0 0 1 0 1 0 0 0 : $2^5+2^3 \Rightarrow 40 \Rightarrow 5*2^3$

a<<4 : 0 1 0 1 0 0 0 0 : $2^6+2^4 \Rightarrow 80$

a<<5 : 1 0 1 0 0 0 0 0 : $-2^7+2^5 \Rightarrow -96$

$-128+32$

a<<6 : 0 1 0 0 0 0 0 0 : $2^6 \Rightarrow 64 \Rightarrow 5*64 \Rightarrow 320$

$\Rightarrow$ a<<5 $\Rightarrow 5*2^5 \Rightarrow 5*32 \Rightarrow 160$ → * overflow

[-128, 127]

a << N : $a*2^N$

1 << N : $1*2^N$

} If no overflow, Then Then formulas hold.

>>:

a : 50:

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

0 0 1 1 0 0 1 0 → *

a>>1 : 0 0 0 1 1 0 0 1 → $2^4+2^3+2^0$ → 25 → $50/_{2^1}$

a>> 2: 0 0 0 0 1 1 0 0 → 2^3+2^2 → 12 → $50/_{2^2}$

a>> 3: 0 0 0 0 0 1 1 0 → 2^2+2^1 → 6 → $50/_{2^3}$

a>> 4: 0 0 0 0 0 0 1 1 → 2^1+2^0 → 3

a >> 5 → $50/_{2^5}$ → $50/_{32}$ → 1

a >> 6 → $50/_{2^6}$ → $50/_{64}$ → 0

a >> 7 → $50/_{2^7}$ → 0

a >> N : $a/_{2^N}$

↳ >> won't cause overflow

TODO!

-ve n >> ?

↳ Saturday

// Problems:

$0 <= N <= 10^9$

$0 <= p <= 30$

} for given N check if i^{th} bit is set or Not

```
bool checkBit(N, p)
{
    N = 29 => 1 1 1 0 1   (4 3 2 1 0)
    p = 2 : set => True

    N = 29 => 1 1 1 0 1
    p = 1 : Unset => False
}
```

i^{th} bit:

```
bool checkBit(N, i) {
    if((N >> i) & 1 == 1)
        return True
    else
        return False
}
```

TODO: Try with &

N = 53

6	5	4	3	2	1	0
0	1	1	0	1	0	1

N: 0 1 1 0 1 0 1

N>>1: 0 1 1 0 1 0

N>>2: 0 1 1 0 1

```
if(N & 1 == 1)        } p == 0
    return True

if((N >> 1) & 1 == 1) } p == 1
    return True

if((N >> 2) & 1 == 1)
    return True
```

$1 <= N <= 10^9$

// countSetBits( )

N = 29 ; 1 1 1 0 1

cnt = 4

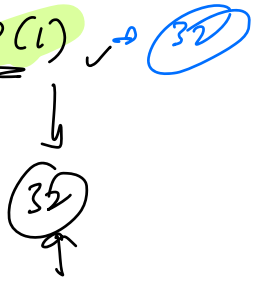
N = 25 ; 1 1 0 0 1

cnt = 3

```
countSetBits(N) {          // 32 bit
                           // i = [0, 31]
    cnt = 0;
    for(i = 0; i < 32; i++) {
        if(checkBit(N, i)) {
            cnt++
        }
    }
    return cnt;
}
```

32 iterations

⇒ O(1) → 32



	5	4	3	2	1	0
N = 50	1	1	0	0	1	0
N >> 1 :		1	1	0	0	1
N >> 2 :			1	1	0	0

cnt = 0 ⇒

```
while(N > 0) {
    if(N & 1 == 1) {
        cnt++
    }
    N = N >> 1
    N = N / 2
}
```

$\log_2 N$

$O(\log_2 N)$

2^{nd} approach better

30) Given N Array element every element repeats twice except 1 find unique element?

ar[7] = {3 2 3 7 2 8 7} 3 8

Idea: XOR of all Elements

4Qs) Every element repeats thrice except 1 unique element

find unique element

1) $N <= 10^6$

2) $1 <= ar[i] <= 10^7$

ar[13]: 5 7 5 4 7 11 11 9 11 7 5 4 4

Idea1: For every element get frequency, & get Element freq = 1

- Nested loops : $O(N^2)$
- Using Hashmaps : $O(N)$, SC: $O(N)$

*

Idea2: Sort the array, iterate & get unique elements

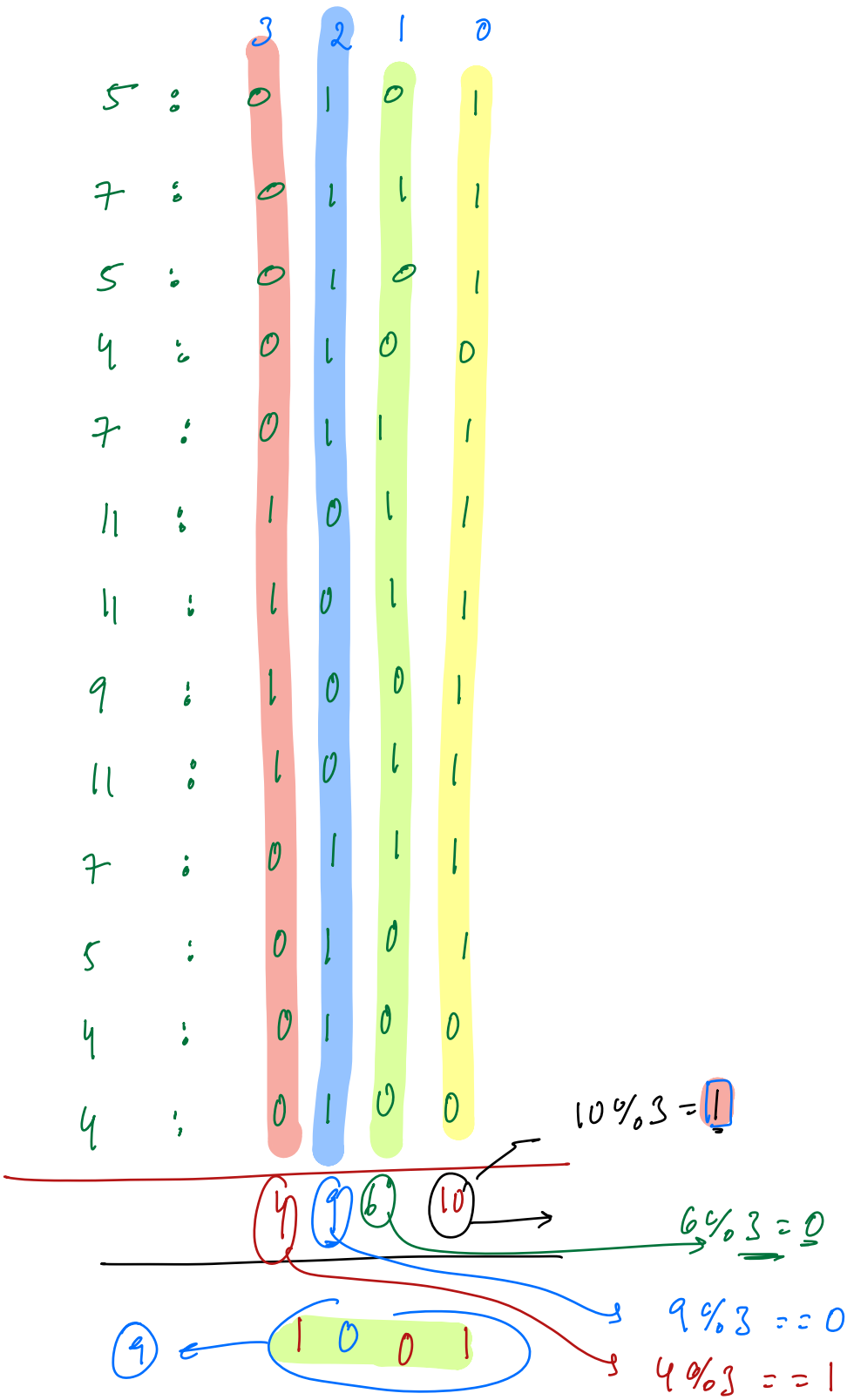
4 4 4 5 5 5 7 7 7 9 11 11 11

↳ ans = 9

- $O(N \log_2 N + N)$

* No External Libraries

ar[13]: 5 7 5 4 7 11 11 9 11 7 5 4 4



Idea: For every bit pos, iterate in entire array & get no of set bits are there = cnt

```
if ( cnt % 3 == 1 ) {
    | At that bit position unique number is set
}
```

* Pseudo Code: ans = 0

i^{th} → (2^i)

i^{th} → $(1 << i)$

```
for(i = 0; i < 30; i++) {
    cnt = 0;
    for(j = 0; j < N; j++) {
        if( checkBit(ar[j], i) {
            cnt++
        }
    }
    if( cnt % 3 == 1 ) {
        ith bit is set in unique Element
        // set ith bit in your ans
        ans = ans | (1 << i)   or
        ans = ans + (1 << i)
    }
}
return ans;
```

TC → 31 * N

TC → O(N)

SC → O(1)

Doubts:

↳ every repeat thrice, except element repeat 2 times

↳ if (cnt % 3 == 2) { unique i^{th} bit set }

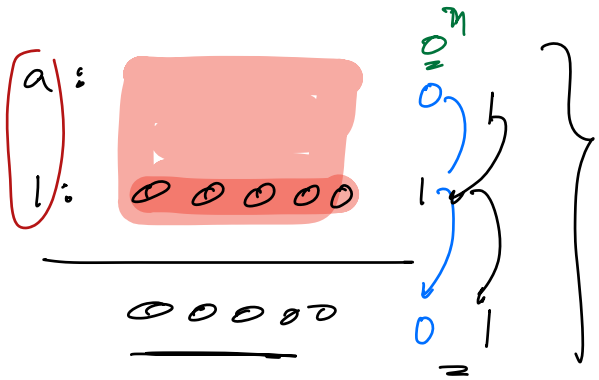
every repeat 4

↳ except 1 element repeat (1) — %4 == 1

↳ except 1 elem repeat (2) — %4 == 2

↳ except 1 elem repeat (3) — %4 == 3

// a & 1 == 1 } a is odd



if (a & 1 == 1)
| 0^{th} bit is set in a
}
else
| 0^{th} bit unset
}

// 6 5 4 3 2 1 0

a: 0 1 0 1 1 0 1

a>>3: 0 1 0 1

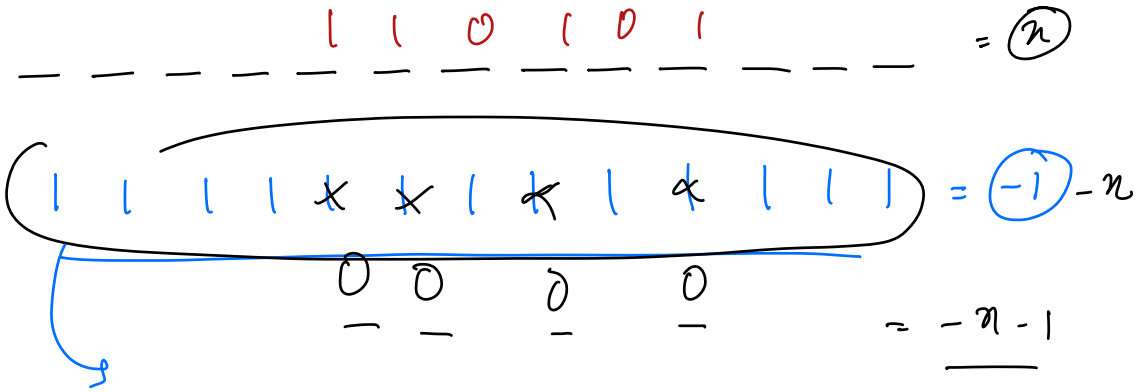
(a>>3) & 1 == 1 }

(a >> i) & 1 == 1

// i^{th} bit in a is set

//

32 bits



$\sim n = -n - 1$

$\sim n + 1 = -n$