Intro Bits 2

Tuesday, October 10, 2023 10:30 AM

Left shift operations

-128 64 32 16 8 4 2 1

byte a = 0 0 0 0 1 0 1 0 : 10

a<<1 : 0 0 0 1 0 1 0 0 : 20 : 10*2

a<<2 : 0 0 1 0 1 0 0 0 : 40 : 10*2^2

a<<3 : 0 1 0 1 0 0 0 0 : 80 : 10*2^3

a<<4 : 1 0 1 0 0 0 0 0 : -128 : 10*2^4

$[-2^8, 2^8 - 1]$ → -128 byte range $[-2^8, 2^8]$

a<<4 a<<4 = 10*2^4 → overflow

Right shift

-128 64 32 16 8 4 2 1

byte a : 0 1 0 1 0 1 0 0 → Data

a>>1 : 0 0 1 0 1 0 1 0 : 20 : 10/2

a>>2 : 0 0 0 1 0 1 0 1 : 10 : 10/2^2

a>>3 : 0 0 0 0 1 0 1 0 : 5 : 10/2^3

(Right shift with Negative Numbers) →

Right Shift : Note : When >> this value never change.

-128 64 32 16 8 4 2 1

byte a : 1 0 0 0 1 1 1 0 → Data

a>>1 : 1 1 0 0 0 1 1 1 → Data

-128 64 + 16 + 2 + 1 = -70 → -128 + 70 = -58

Quiz 5

byte a = 56

-128 64 32 16 8 4 2 1 → Data

a>>2 : 0 0 0 0 1 1 1 0 : 8 = 56/2 = 56/4 = 8

Quiz 6 : a>>3 = 56/2^3

Quiz 7 : 256>>6 = 256/2^6 = 256/64 = 4

Quiz 8 : byte a = -1

-128 64 32 16 8 4 2 1 → -1

a>>1 : 1 1 1 1 1 1 1 1 → -1

a>>2 : 1 1 1 1 1 1 1 1 → -1

= -128+(127) = -1

byte a = -1

while(a != 0){

a = a>>1

}

3 loop

N is Integer : 4B

$[-2^{31}, 2^{31}-1]$

1 ≤ N ≤ 10^9

0 ≤ k ≤ 30

Given N, check if 1 bit in N is set in k

check 5th (k=4) {

N=21 : 10101

k=4

1 set

N=25 : 11001

k=2

clear

Quiz 9 :

N=24 : 11000

k=2

unset

Quiz 10 :

N=14 : 01110

k=2

set

N = 106 : 1 1 0 1 0 1 0

k=3 : 1 1 0 1 0 1 0

N&(1<<k) : 0 0 0 1 0 0 0

0 0 0 1 0 1 0 1 = 8

N = 106 : 1 1 0 1 0 1 0

k=1 : 1 1 0 1 0 1 0

N&(1<<k) : 0 1 0 0 0 0 0 0 = 32

N = 106 : 1 1 0 1 0 1 0

k=0 : 1 1 0 1 0 1 0

N&(1<<k) : 0 0 1 0 0 0 0 0

N = 106 : 1 1 0 1 0 1 0

k=3 : 1 1 0 1 0 1 0

N>>3 : _ _ _ _ 1 1 0 1 0 : (N>>3)&1 == 1

N = 106 : 1 1 0 1 0 1 0

k=4 : 1 1 0 1 0 1 0

N>>4 : _ _ _ _ _ 1 1 0 1 0 : (N>>4)&1 == 1

preferred from Right Shift

Generalize, N & 1 : (N>>k)&1 == 1

Generalize N & 2 : N&(1<<k)!=0

N&(1<<k) == 2

N&(1<<k) == (1<<k)

(N>>k)&1 == 1

(y>>k)&1 == 1

y = (N&(1<<k))

N&(1<<k)>>k == 1

Count Set Bit :

Given N get how many set bits to N

N = 30 : 3 2 1 0 : 1 1 1 1 0 = 4

1 ≤ N ≤ 10^9

Mask

Count Set Bit(N) {

int c=0;

k=0; k<31; k++{

if(check Bit(N, k)){

c++

}

}

return c;

}

N=16

Implement on your own

Iterative : [0, 30] → 31 Iterations

Indexed & Mask → 21

Quiz :

N = 29 : 1 1 0 1 1 : 29

5 3 2 1 0

N&1 : 1 0 1 1 1

N>>1 : 1 1 0 1 1

int c=0;

while(N>0){

if(N&1){

c++

}

N = N>>1

}

return c;

N=10

N&10 : 20

pow(2, 10) : 10

N>>1 : N value never get updated

Iterating : Least Bit → Left Shift

N = 16 → 5 (2^5)

Left is better

4 Iterations

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Given N array elements. Every element repeat twice except 2 unique elements, find 2 unique elements.

Ex: 3 4 5 4 7 5 : 3, 7

Ex: 4 7 9 8 : 4 8

Idea: Xor of all elements

: 11^10

: 01011

: 10001

: 11010

: 2

All bit pos = 1, both 11 & 10 are different

Step: (Find 4th bit)

set → 0

unset → 1

10^0^11^10

All repeat than Except 10

{xor of all elements} & pos=1

{9, 8, 7, 12, 10, 14}

{7, 11, 10, 12}

{4, 5, 13}

19

11

Step: Divide array based on 4th bit

Step: Iterator & XOR all

ans=0; → answer

N

T=0; T<N; T++{

ans = ans ^ ar[T]

}

ans : xor of 2 unique elements

Get any bit pos which is set

T=0; T<=30; T++{

if(check Bit (ans, T)){

pos = T; break

}

}

For one pos bit set

int set=0, unset=0

T=0; T<N; T++{

if(check Bit (ar[T], pos)){

set = set ^ ar[T]

}

else{

unset = unset ^ ar[T]

}

}

→ set & unset are 2 unique numbers

Today Issue : N ≤ 2^31

→ 32B → 2^32 numbers (0-2^32) → Integer / Float

byte : 8 bits

Signed

Unsigned

-2^7 ... 2^7-1

Default Signed

char : 1B : 2^8 bits

[-2^8, 2^8-1]

long long :

[-2^63, 2^63-1]

1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0

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

Max : 2^0+2^1+2^2+...+2^7 = 2^8-1

[0, 2^8-1]

Unsigned int : 2B bits

Positive

[0, 2^16-1]

Unsigned long long : 8B = 64 bits : [0, 2^64-1]

1 << 20 : 2^20

1 << 21 : 2^21 → overflow

signed int : [-2^21, 2^21-1]

1u << 21 → 2^21 ✓

unsigned int : [0, 2^22-1]

1u << 21 → 2^22 overflow

[0, 2^22-1]

1ll << 62 → 2^62

long long : [-2^63, 2^63-1]

1ll << 63 → 2^63 overflow

[-2^63, 2^63-1]

1ull << 63 → 2^63

[0, 2^64-1]

1ull << 64 → 2^64 overflow

[0, 2^64-1]

N	N-1	N & N-1
4 : 100	011	0
8 : 1000 →	0111	0
16 : 10000 →	01111	0
20 : 10100 →	10011	10000
24 : 11000 →	10111	10000
28 : 11100 →	11011	11000
47 : 101000	101011	101000

observation

In N & N-1

Repeat Mark left bit

left become 0

int c=0;

while(N>0){

N = N&(N-1)

c++

}

return c; → Number of set bits to N

For every iteration 1 set bit becomes 0, so loop iterations N=30

→ long → 8B → 4B

long long : 8B

Integer