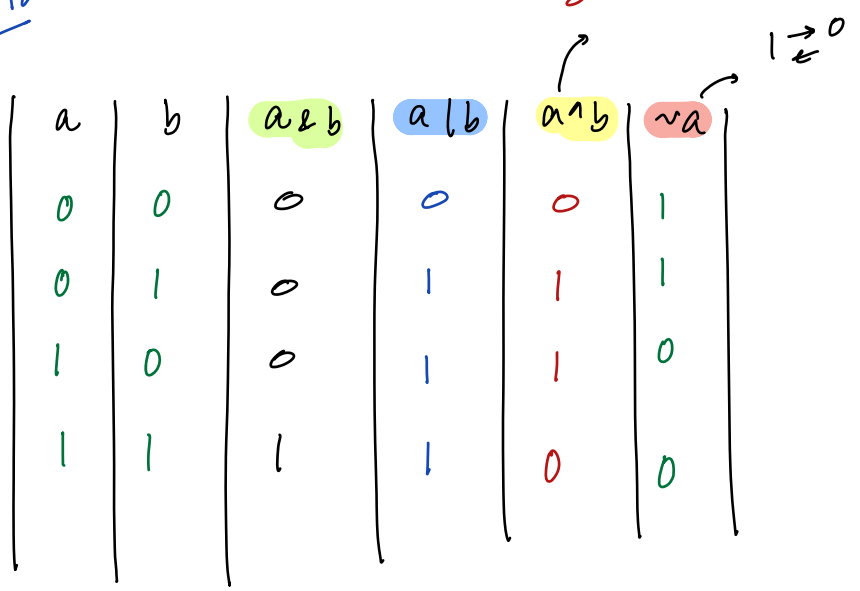
Content:

- → Bitwise operators & properties
- → Check Bit
- → Unique element
- → Single Element - II
- → Single Element - I

Bit wise operators → ( &, |, ^, ~, <<, >> )

Same Same puppy Shame = 0

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 |

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

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 0 1 1 1 1 = 15

Properties:

$a \& b = b \& a$

$a \wedge b = b \wedge a$

$a | b = b | a$

// Commutative property

$a \& b \& c = b \& c \& a = c \& a \& b$

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

$a \wedge b \wedge c = b \wedge c \wedge a = c \wedge a \wedge b$

// Associative property

$a = 10$

```
    3 2 1 0
a : 1 0 1 0
1 : 0 0 0 1
-----------
a&1 : 0 0 0 0
```

$a = 11$

```
    3 2 1 0
a : 1 0 1 1
1 : 0 0 0 1
-----------
a&1 : 0 0 0 1
```

// If $(a \& 1) == 0$

: a is even

: $0^{th}$ bit is unset

else

: a is odd

: $0^{th}$ bit is set

properties

$a \& 0 = 0$

$a \& a = a$

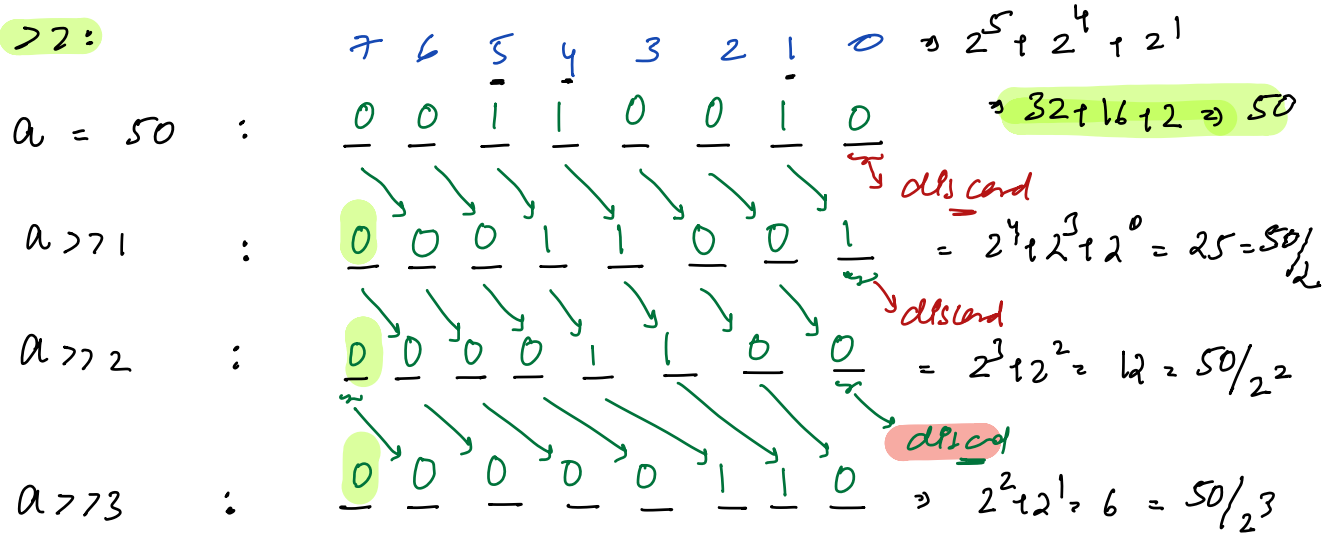
$a \wedge 0 = a$

$a \wedge a = 0$

$a | 0 = a$

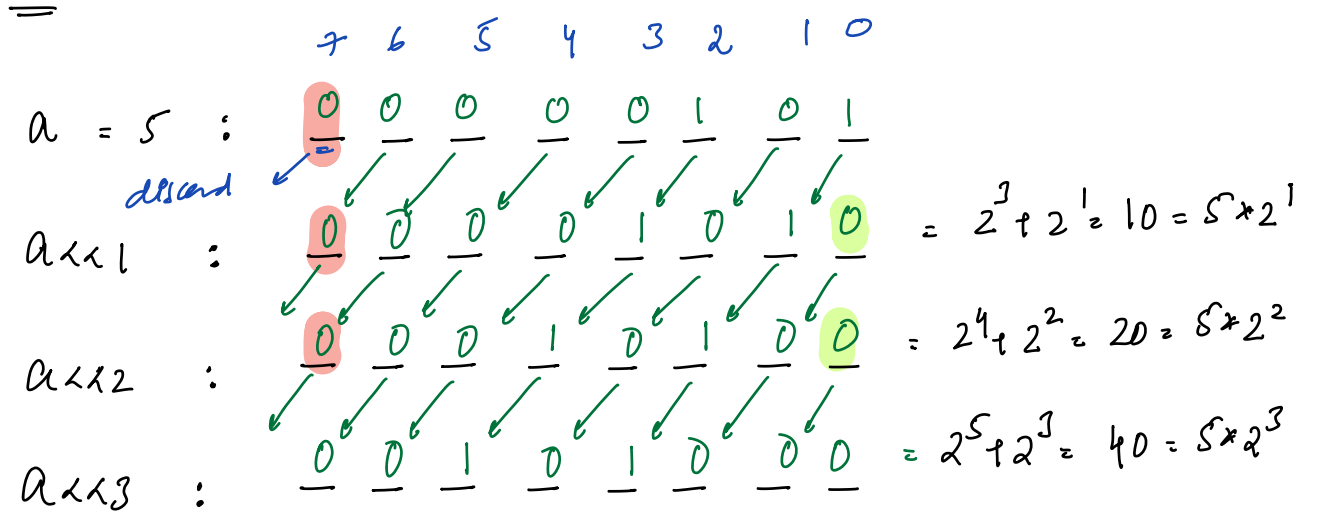
$a | a = a$

>>:




a >> N : $\frac{a}{2^N}$

<<:




a << N : $a * 2^N$ // Assume no overflow

↓ a = 1

1 << N : $1 * 2^N$ ⇒ 1 << N = $2^N$ // Assume no overflow

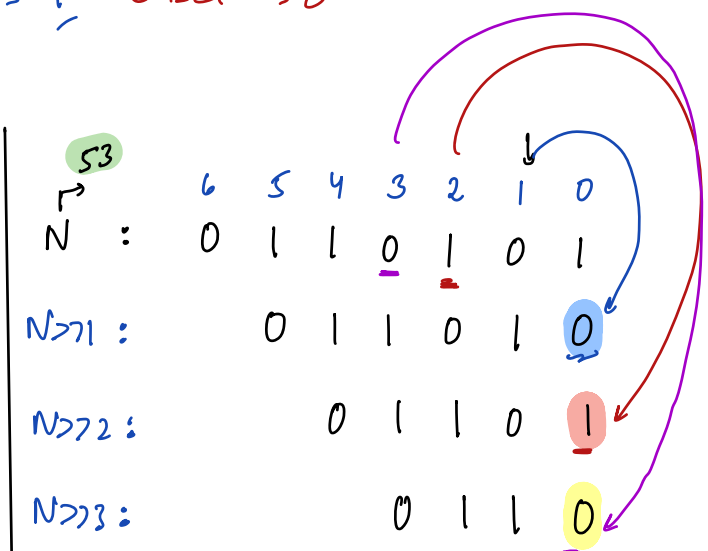
// Problems: int → 4B → 32 bits → pos: [0 31]

0 <= N <= $10^9$
0 <= i <= 30
} for given N check if $i^{th}$ bit is set?
Set → 1 Unset → 0

```
bool checkBit(N, i){
    N = 29 : 1 1 1 0 1
    i = 2  : (True)

    N = 29 : 1 1 1 0 1
    i = 1  : (False)
}
```

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




i=0 : (N&1) != 0 : set / unset
i=1 : (N>>1)&1 != 0 : set/unset
i=2 : (N>>2)&1 != 0 : set/unset
i=3 : (N>>3)&1 != 0 : set/unset

TODO:
(Try with left shift):

38) Given N array elements, every element repeats twice except 1, find unique element ?

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

// ans = { xor of all Elements }

TC : O(N)

SC : O(1)

{ 9:50 PM }

48) Every element repeats thrice except 1 (≥ 1) unique element

find unique element

**Constraints**

1) $1 <= N <= 10^6$
2) $1 <= ar[i] <= 10^9$

$ar[7] = \{ 3\ 7\ 6\ 7\ 7\ 3\ 3 \}$

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

Ideal: For every element get frequency by iterating in array

TC: $O(N * N)$ SC: $O(1)$

(N: Iterate & get freq)

Idea2: For every element get frequency using hashmap

TC: $O(N)$ SC: $O(N)$

Idea3: Sort the array + (Iterate & check every 3, get unique Ele)

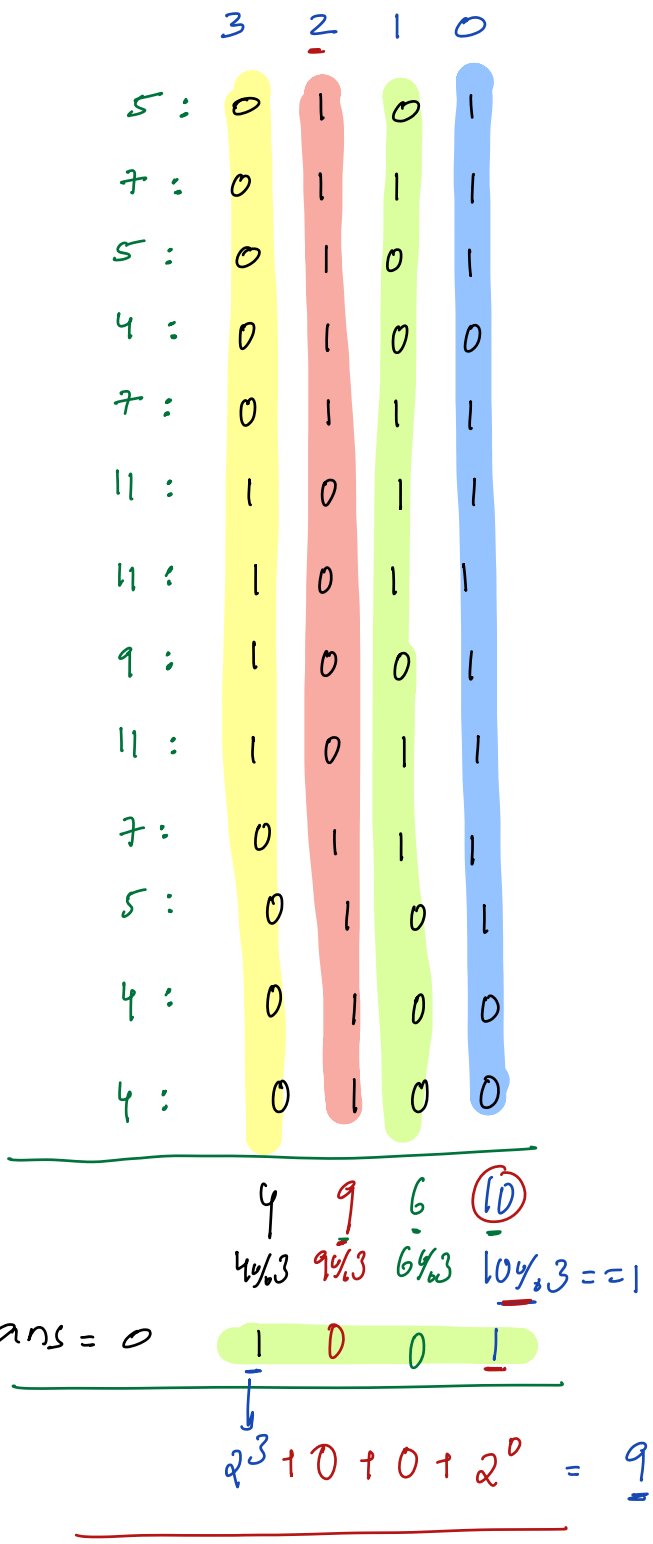
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 4 | 5 | 5 | 5 | 7 | 7 | 7 | 9 | 11 | 11 | 11 |

i

→ final ans

TC: $O(N \log N + N)$ SC: $O(1)$

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



Idea: For every bit position, iterate in array & count number of set bits = cnt

if ( cnt % 3 == 0 ) { In unique element $i^{th}$ bit unset }

else { // set }

Pseudocode

```
ans = 0
for (i = 0; i < 30; i = i + 1) {
    // for i'th bit get no. of set bits
    c = 0
    for (j = 0; j < N; j++) {
        if (checkBit(A[j], i)) { c = c + 1 }
    }
    if (c % 3 == 1) {          // In unique elem i'th bit set
        ans = ans + 2^i (1 << i)
        ans = ans | (1 << i)
    }
}
return ans;
```

Both approaches are correct?

TC: 31 * N ⇒ O(N)  SC: O(1)

Versions:

$V_1$: Every element repeats three except 1:

$V_2$: Every element repeats three except 1, (Comes 2 times)

c%3 == (2)

$V_3$: Every element repeats 5 except 1, (Comes 1 time)

c%5 == 1

$V_3$: Every element repeats 5 except 1, (Comes 2 times)

c%5 == 2

$V_3$: Every element repeats 5 except 1, (Comes 3 times)

c%5 == 3

38) Given N Elements every element repeats twice except 2 unique elements, find the 2 unique elements

(will comes = 1)

Ex: ar[6] = {3 6 4 4 3 8} : {6 8}

ar[4] = {4 9 9 8} : {4 8}

Idea1: for every element, iterate & get frequency == 1

TC: N * O(N) → O(N²) SC: O(1)

optimize ↓

Idea2: For every element, get frequency using Hashmap

no extra space

TC: O(N) SC: O(N)

Idea3: Sort the array + (Iterate & get unique Elements)

{same elements come together} TC: O(NlogN + N) SC: O(1)

Idea4: XOR of all Elements?

Ex: ar[8] = {3^7^6^7^3^8^9^9} = 6^8 = 14

using 14 we cannot 2 unique Elements?

ar[12]

3 2 1 0

(1010) (1000) (11100) (01110) (1010) (11100)

10 8 8 9 12 9 6 11 10 6 12 17

(1000) (1001) (1001) (1011) (0110) (10001)

1) xor of all Element

val = 11 ^ 17

= (xor of 2 unique)

| | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| 11 : | 0 | 1 | 0 | 1 | 1 |
| 17 : | 1 | 0 | 0 | 0 | 1 |
| (11^17) : | 1 | 1 | 0 | 1 | 0 |

obs:

At bit pos (1): diff :

At bit pos 3 : diff

At bit pos 4 : diff

Take any set bit position as xor value

// Say Split array based on 1st bit pos

1st bit

set → {10, 6, 11, 10, 6,} xor of all : 11

unset → {8, 8, 9, 12, 9, 12, 17} xor of all : 17

// Say split array based on 3rd bit pos

3rd bit

set → {10, 8, 8, 9, 12, 9, 11, 10, 12} xor of all : 11

unset → {6 6 17} xor of all : 17

Code:

Step 1: (xor of all elements)

```
v = 0;
for (i = 0; i < N; i++) { v = v ^ ar[i] }
```
→ O(N)

Step 2: for v get a set bit position

```
for (i = 0; i < 30; i++) {
    if (checkBit(v, i)) {
        pos = i; break
    }
}
```
O(1)

At pos both unique elements have diff set bits

Step 3: Split array based on Set & Unset based on pos

```
set = 0, unset = 0
for (i = 0; i < N; i++) {
    if (checkBit(ar[i], pos)) {
        set = set ^ ar[i]
    }
    else {
        unset = unset ^ ar[i]
    }
}
```
O(N)

TC: O(N + 1 + N) ≈ O(N)

Step 4: 2 unique elements : (set / unset)

SC: O(1)

Q. Given N Array Element, array contains all elements from [1 N+2] except 2 elements, find 2 missing elements

ar[4] = { 3 6 1 4 } → [1-6] → missing elements [2, 5]

ar[5] = { 1 6 4 7 5 } → [1 7] → missing elements [2 3]

Ideas:
- Try of N → Hashmap → keeping an element on it's position
- → bool ch[N+2]
- → Sort
- → BF

ar[5] = { 1 6 4 7 5 } → [1-7] → { 1 2 3 4 5 6 7 }

N

(Idea: Every element is repeating except 2 unique elements)

a → xor of all array elements

b → xor of all Elements from [1 7]

(a ^ b) : {xor of all Elements}

Pseudocode // Given ar[N] : {1, (N+2)}

Step1 :
- a = (xor of all array Elements)
- b = (xor of all Elemets [1 N+2] : (Iterate again)
- V = a ^ b

Step2 : Step bit position on V = pos }

Step3 . set = 0 , unset = 0

Splitting array values

```
for(i=0; i< N; i++){
    if(checkBit(ar[i], pos){ set = set ^ ar[i]; }
    else { unset = unset ^ ar[i]; }
}
```

Splitting all [1 N+2]

```
for(i=1; i<=(N+2); i++){
    if(checkBit(i, pos){ set = set ^ i; }
    else { unset = unset ^ i; }
}
```

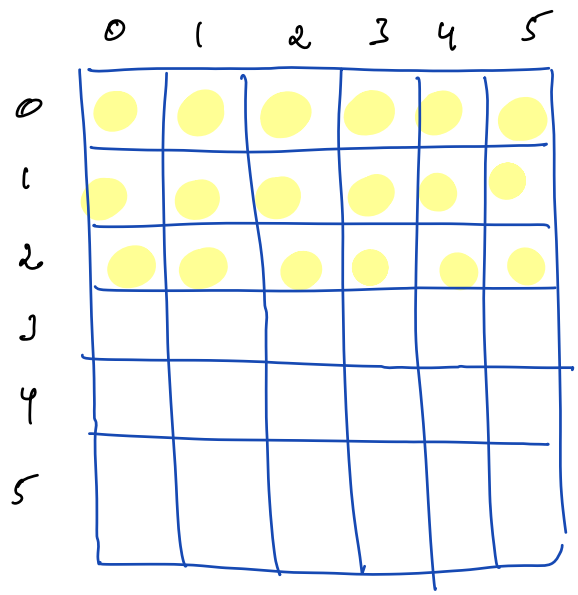
// set & unset are missing Elements

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

{ Repeat & Missing Number } → very similar to above question

→ {Doubts}

pf[][] :




Sq =