Bit Manipulation -3

Jun 13, 2022

AGENDA

- Some problems on Bit Manipulation
- Brief on how to represent -ve nos.
  (in binary form)
- Overflow

Bitwise operators.

Some more properties.

Commutative. $\rightarrow$ $a \text{ op } b = b \text{ op } a$

Associative. $\rightarrow$ $a \text{ op } (b \text{ op } c) = (a \text{ op } b) \text{ op } c$

* Addition

$a+b = b+a$ $\Rightarrow$ Commutative

$a+(b+c) = (a+b)+c$ $\rightarrow$ Associative

* Subtraction/Division

$a-b = b-a$ ✗ $\rightarrow$ Not commutative

$a-(b-c) = (a-b)-c$ ✗ $\rightarrow$ Not associative

* Multiplication

↳ Both comm. & associative

$a \wedge b$ $\rightarrow$ No

$a \% b$ $\rightarrow$ No

OR → Comm. & Assoc.

$a | b = b | a$ $\quad$ $0 \wedge 1 = 1 , \; 1 \wedge 0 = 1$

$a | (b | c) = (a | b) | c$

AND

$a \& b = b \& a$

$a \& (b \& c) = (a \& b) \& c$

XOR

$a \wedge b = b \wedge a$ ←

$a \wedge (b \wedge c) = (a \wedge b) \wedge c$

Q. Given N array elements, where each no. appears twice except one no. → which appears once. Find that no.

{1, 1, 2, 0, 0, 1, 2, 2, 3, 3, 1, 2, 4}

Brute force → $O(N^2)$

Hash Map/Dictionary → $O(N)$, $O(N)$ space

Sort → $O(N \log N)$

$\boxed{XOR}$

$a \wedge a = 0$
(XOR cancels itself).

{1, 1, 2, 0, 0, 1, 2, 2, 3, 3, 1, 2, 4}

$1 \wedge 1 \wedge 2 \wedge 0 \wedge 0 \wedge 1$

0

$2 \wedge 1 = ?$

$2 \wedge 1 \wedge 2 = 1$

ans = XOR of all nos.

All nos. appear thrice, one no. appear once.

HW — Using Bit Manipulation

Q. Given N elements where each no. appears twice, except 2 nos. which appear only once.
Find the 2 nos.

Inp: {2, 2, 1, 1, 0, 1, 6, 0, 1, 6, 3, 4}

Out: 3, 4

A B

$A \wedge B$

H $A \wedge B = \begin{array}{ccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$

XOR

0 0

0 1

1 0

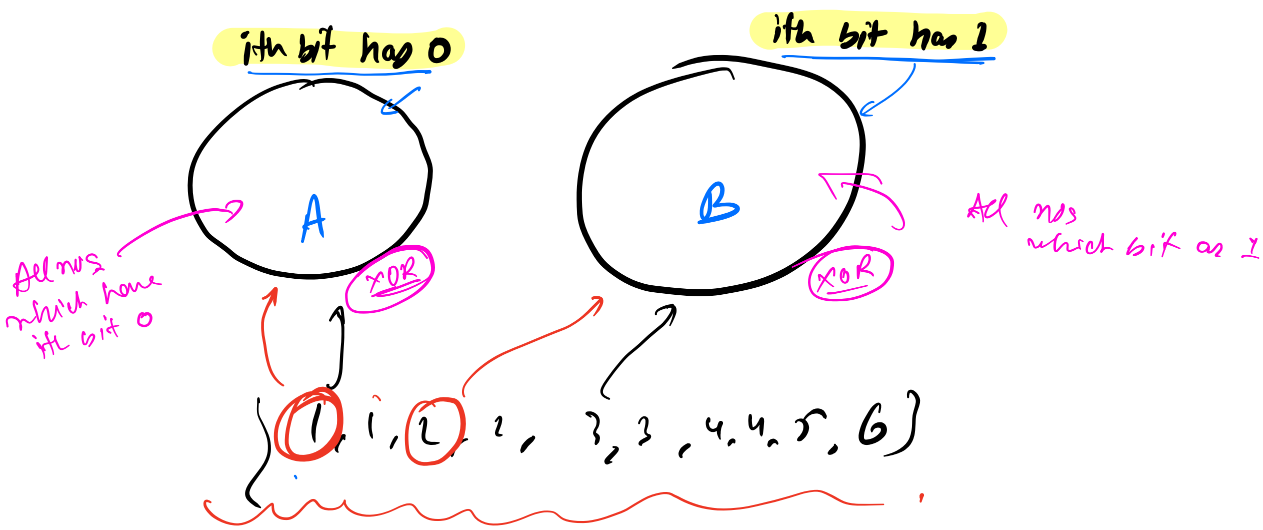
1 1

A and B will differ at
1th, 4th and 6th place.

$\boxed{A \wedge B}$ = XOR all no. in an array

Let

** A and B differ with each other at ** ith bit (right)



$\{1, 1, 2, 2, 3, 4\}$

XOR $= 3 \wedge 4$

$= 7$

$\begin{array}{ccc} 2 & 1 & 0 \\ 1 & 1 & \boxed{1} \end{array}$

XOR $3 \wedge 4$

$$\begin{array}{r} 0011 \\ 0100 \\ \hline 0111 \end{array}$$

A an B differ with each other at 0, 1, 2 indexes. (bit) (Binary repr.)

right most set bit from this XOR value?

Clear the right most set bit → N & N-1

11000100 → 11000000

(N & N-1) ^ N =

$$\begin{array}{rll} & 111 & N \\ \wedge & 110 & N \& N\text{-}1 \\ \hline & 001 & \end{array}$$

Mask → 001

$$\begin{array}{rl} N & 1100100 \\ \wedge\ N\&N\text{-}1 & 1100000 \\ \hline & 0000100 \end{array}$$

Mask → 0000100

val = XOR of all nos. in array

Mask = (val & val-1) ^ val

000000.....1000

bit position where A & B differs with each other.

* ith bit → one of the bit positions where A & B differs with each other.

{1, 1, 2, 2, 3, 4}

5

7

0

3 ^ 5

$$\begin{array}{r} 011 \\ 101 \\ \hline 110 \end{array}$$

it = 1

bit position = i

Group 1

1 1
3

where 0th index is set = 1

0001
0010

Group 2

2 2
4

where 0th index is unset.

Observations

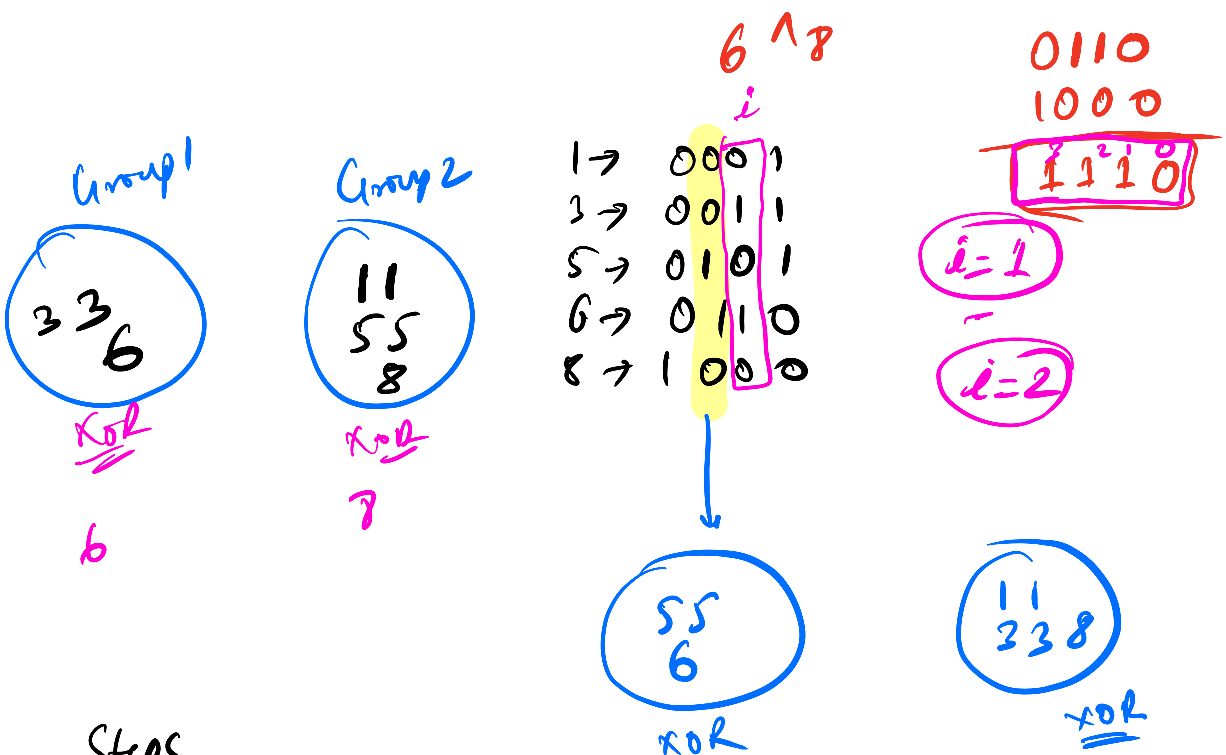
1. A and B (no. that we have to return) will always lie in different groups.

2. All the repeated nos. in same group.

1, 1, 3, 3, 5, 5, , 6, 8

1 1
3 3
5 5

6
8



Steps.

① Take XOR of all nos.

② Choose an 'i' where XOR value has a set bit

$0 \le i \le N-1$

③ Mask → $000000\ldots\overset{i}{1}\ldots0000$

④ mask & A[i] $== 0$ → Group 1

$== 1$ → Group 2

How to come up with this mask?

10000111000

① Find out any 1 bit which is set, keep it set, and clear all other bits.

$$
\begin{array}{lll}
 & N & 1000111\,1000 \\
N \& N-1 & & 1000111\,0000 \\
\hline
 & & 00000\,10000
\end{array}
$$

val = XOR of all nos.

Mask = val ^ (val & val-1)

Pseudo-code

{1, 1, 3, 3, 5, 5, 2, 4}

```
xOR_val = 0
for (int i=0 ; i< n ; i++)
    xOR_val = xOR_val ^ arr[i]
```

010
100
110

```
mask = xOR_val ^ (xOR_val & xOR_val-1)
```

110
100
000010

```
// Divide into 2 groups,
int A = 0, int B = 0.
for(int i=0 ; i<n ; i++)
```

```
{
        if (arr[i] & mask == 0)
            A = A ^ arr[i]

        else
            B = B ^ arr[i]

}
return A, B;
```

Break till 10:13

* No. of subarrays with OR as 1.

{0 1 0 0 1 0 0 0 1 1 0 0 1 1 0}

Brute force → For every subarray, check whether OR is 1.

↳ $O(N^2)$ subarrays.

$O(N^3)$

Ans → Count all subarrays which has atleast 1 in it.

= Total no. of subarrays − No. of subarrays with all 0's.

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

{0 1 [0 0] 1 0 0 0 1 1 0 0 1 1 0}

1 3 6 3 1

Total no. of subarrays = $\frac{N(N+1)}{2}$ = 15 × 8 = 120

2,3 → $\frac{2 \times 3}{2} = 3$

$\frac{3 \times 4}{2} = 6$

1+3+6+3+1 = 14

No. of subarrays with at least one 1 = 120 − 14
= 106

Another approach.

Count all subarrays with 1 in it.

0 2 2 2 5 5 5 5 9 10 10 10 13 14 14 =

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

{0 1 0 0 1 0 0 0 1 1 0 0 1 1 0}

[0, 1, 0
1, 0]

0+2+2+2
+5+5
5+5

```
ans = 0
last = 0      // No. of subarrays ending at that point.
for( i = 0; i<n ; i++ ){
    if (A[i] == 1)
        last = i+1
    ans += last
return ans
```

1+2 4+4 1

{1, 1, 2, 3, 4, 6, 8, 1, 2, 0, 4, 20} → {0}

Merge any 2 nos. using XOR.
Split any 2 nos. using +.

{1, 2, 3, 4, 5, 6}

1+1 1+2

All even nos. can be converted to 0?
↳ split into 2 equal parts.
↳ Merge it using XOR.

All odd nos. can be converted to 1?
↳ split into 1 and n-1
↳ Convert n-1 to 0 because n-1 is even.

{1, 2, 3, 4, 5, 6}

{1, 0, 1, 0, 1, 0}

if I have even 1's → XOR → 0
odd 1's → 1 will remain.

Ans. → Count no. of odd nos.
if count is even
return True,
if count is odd
return false.

Bit compression

For every pair. $A[i] = A[i] \& A[j]$
$A[j] = A[i] \mid A[j]$

$$(A[i] \& A[j]) \wedge (A[i] \mid A[j]) \quad \forall \; i \& j$$

[1, 2, 3, 4, 5, 6, 7, 8]

3&4 3/4

All pairs of i,j ,(k,j)

(3&4&7) (3&4)|7

a, b.

** $(a \& b) \wedge (a \mid b) = a \wedge b$

	$a \& b$	$a \mid b$	$a \wedge b$
	0	0	0
	0/1	1/0	→ 1
when a and b = 1	1	1	→ 0

XOR

when a and b both are same/different

$a \& b$ → 1, $a|b$ → 1 ⇒ a and b both are 1

$a \& b = 0$
$a|b = 0$ } ⇒ a and b both are 0.

a & b
a | b
→ if both are same, a and b are also same
→ if both are diff, a, b are are also diff.

$$(a \& b) \wedge (a|b) = a \wedge b.$$

Truth table.

a	b	a & b	a | b	a ^ b	(a&b)^(a|b)
0	0				
0	1				
1	0				
1	1				

Ans → XOR all the originals present in the array.

Represent Negative nos.

8 bit

1/0

MSB → sign bit → 1 → negative no.

0 → positive no.

4 bit

0000	→	0
0001	→	1
0010	→	2
$\boxed{0011}$	→	3
0100	→	4
0101	→	5
0110	→	6
0111	→	7
1000	→	

~~8~~ ~~X~~

0011 → 3

-3

X 1 011 → -3 → Sign magnitude form

sign. magnitude

X

2 0's possible

0000
1000 } XX -0 +0

0011 → 3

1100

Add 1 to it.

↓

$\boxed{1101 \rightarrow -3}$

2's complement.

0 → 0000

1 → 0001

−1 → 1110 = 1111

+1

2 → 0010

−2 → 1

101

+1

110

= 1110

* Take the complement and add 1.

// 2's complement form of representation of −ve nos.

// Widely accepted

0000	→ 0
0001	→ 1
0010	→ 2
$\boxed{0011 \rightarrow 3}$	
0100	→ 4
0101	→ 5
0110	→ 6
0111	→ 7 — 8
1000	→ −8
1001	→ −7
1010	→ −6
1011	→ −5
1100	→ −4
1101	→ −3
1110	→ −2
1111	→ −1

−8 to 7

1000 — 8 or 8 ?

1111 → 15 or −1 ?

① Always know the no. of bits in your system.

② Check whether the data is signed/unsigned.

if its is a 4 bit no. and signed.

1111 → −1

4 bit no. but unsigned,

1111 → 15

if it is a 8 bit no and signed,

00001111 → 15

" " 8 bit no. and unsigned.

00001111 → 15

int 32 unsigned → 0 to $2^{32} - 1$

int 32 signed → -2^{31} to $2^{31} - 1$

signed 4bit 1111 → -1

-1

-1 into binary.

0001

1110

+1

1111

-1 → 1111

0000

+1

0001

-4 :

0100

1011
$+1$
1100 = -4

1001 → -7

0110
1
0111 → 7

$$\begin{array}{r} 10001101 \\ (-)\ 00010111 \\ \hline 01110110 \end{array}$$

89000101027
8999878899

000000000 → 000111010010
0
+1
+1
A →

× 2 → No charg
+ 1 → charge.

Count no. of set bits → Ans

00101010

0000 0000

+1

0000 0001

×4

00000100

+1

0000 0101

×4

0001 0100