People who feel good
about themselves
Produce good results

Content Today:

1. Single number 1
2. Single number 2
3. single number 3
4. maximum & Pair

Quiz 1: if Last bit of number is One. the number is odd

Quiz 2: what is the Time Complexity of "checking ith bit set or unset
"Constant time"

```
A & (1<<i) > 0
# ith bit is set
```

Quiz 3: Total Cont of set bits in Integer. Time Complexity is O(log N)

Problem 1: Single Number 1

→ given N elements
→ every element repeats twice except one
→ find the unique element

A = 2 4 3 7 4 8 7 ans = 8

A = 4 5 5 4 1 6 6 ans = 1

```
ans = A[0]
for i in range(1, n-1):
    ans = ans ^ A[i]
return ans
```

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

idea 2 → sorting O(nlogn)

idea 3 → hashmap TC: O(N)
SC: O(N)

idea 4 → A[] = [2 3 5 6 3 6 2]

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

=> if element is repeating twice, number of set bit at every index will be even

```
ans = 0
for i = 0 → 32                          ---------->  O(32)
    count = 0
    for j = 0 → N-1                     ---------->  O(N)
        if (A[j] & (1 << i)) > 0
            count ++
    if count & 1 > 0  #odd
        # ans = 1
        ans = (ans | (1 << i))
```

TC = O(N)

SC = O(1)

Problem 2: Single Number 2

↳ Given an arr[N], all the elements present thrice except one, Find the unique one.

one → occuring once

A = [2, 3, 2, 2, 4, 3, 3] Ans = 4

1 → 2, 3 → 3

Brute force: $O(N^2)$

1. two for loop and calculate each element frequency.

Sorting approch: $O(n \log n)$

1. sort the array
2. iterat the array and keep the count of

```
Count = 0,
for (j = 0 -> N-1):
    if A[j] == A[i]
        count += 1
    if count == 1:
        return A[i]
```

Hash map approch : TC : O(N) SC: O(1)

Contribution Technique for Bit manipulation

1. Counting each index bit of every element and if it is multiple of 3 no action.

2. if

a = [15 7 5 4 7 11 11 9 11 7 5 4 4]

8 =	0	1	0	1
7 =	0	1	1	1
8 =	0	1	0	1
4 =	0	1	0	0
7 =	0	1	1	1
11 =	1	0	1	1
11 =	1	0	1	1
9 =	1	0	0	1
11 =	1	0	1	1
7 =	0	1	1	1
5 =	0	1	0	1
4 =	0	1	0	0
4 =	0	1	0	0
	4	9	6	10
	1	0	0	1 = 9

if % 3 != 0

10 is not multiple of 3
add 1

else:
# 6 is multiple of 3
add 0

ndm v2

```
Ans = 0
for i in range(32)                    --> O(32)
    count = 0
    for j in range(len(A))            --> O(N)
        num = A[j]
        (num & (1<<i)) > 0            => O(32 x N)
            count += 1                => O(32N)
    if count % 3 != 0
        ans = ans | (1<<i)
return ans
```

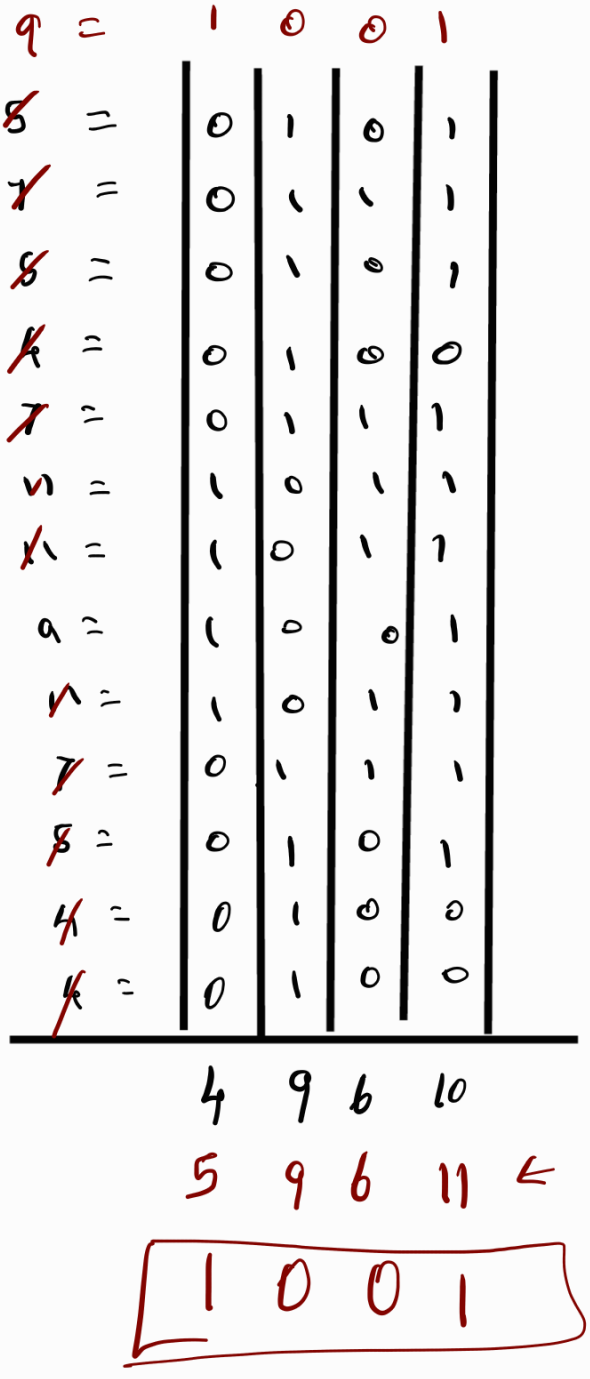
TC : O(N)

SC : O(1)

Extention:

1. Every element is repeating thrice except one element which is reapeting twice.

Same exact method of above work here.



2. Every element repeating 4 times except one element, which is repeating one time

=> XOR of all the elements in Array

=>

3. Every element repeating 4 times except one element which is repeating 2 times.

=> number of set bit in ith index is going to be multiple of '4'

```
if ele & (1<<i) > 0
    count++

if count % 4 != 0
    ans = ans | (1<<i)
```

4. Every element repeating 4 times except one element which is repeating '3' times.

XOR can be used

Single number 3

Given int[] A. all element repeats twice, except two no.

A = [4 5 4 5 2 1] ans = [2, 1]

X = X | A[i]

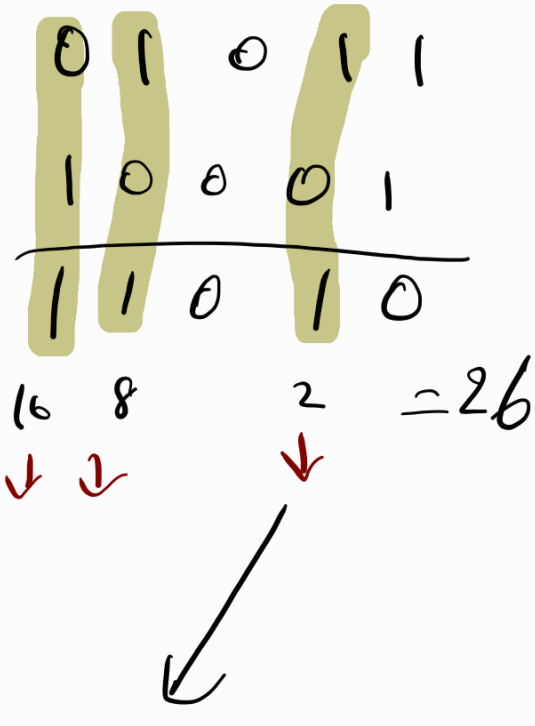
Later extract 2 elements of Answer from X

A = [4 5 4 1 1 6 5 2] ans = 2, 6

A = [10 8 8 9 12 9 6 11 10 6 12 17]

ans = 11 ^ 17

we have the repeating value ⇒ 11010



how to extract ans?

=> Split the entire array on the basis of 1st index.

=> "11 010" has 1 in 1st index

set | un-set

10, 6, 1, 10, 6 | 17, 8, 8, 9, 12, 12

↳ Take XOR of all left element to get any 1)

↳ Tak XOR of all right ele also.

```
# perfor xor operaton on array
x = A[0]
for i in range(1, len(A)):
    x = x ^ A[i]

# find first index of set bit in x
idx = -1
for i in range(32):
    if x & (1 << i) > 0:
        idx = i
        break

# split the idxth set bit element

Set bit = 0
unset bit = 0
for i in range(len(A)):
    if A[i] & (1 << i) > 0:
        setbit = setbit ^ A[i]
    else:
        unsetbit = unsetbit ^ A[i]
```

Problem — Maximum And Pair

↳ Given int[] of size N.

↳ choose two indices (i, j) such that (i != j) && A[i] & A[j] is maximum.

$A = \{27, 18, 20\}$

27 & 18 = 10010

27 & 20 = 10000

18 & 20 = 10000

✓

27 = 11011

18 = 10010

20 = 10100

Brute force : TC: $O(N^2)$ SC: $O(1)$

1. Consider all the pairs
2. Take bitwise AND
3. maintain the maximum.

```
max = 0
for i in rang(N)
    for j in rang(N)
        max = max(max, A[i] & A[j])
```

$A = \{ 26 \quad 13 \quad 23 \quad 28 \quad 27 \quad 7 \quad 25 \}$

26 =	1	1	0	1	0
13 =	0	1	1	0	1
23 =	1	0	1	1	1
28 =	1	1	1	0	0
27 =	1	1	0	1	1
7 =	0	0	1	1	1
25 =	1	1	0	0	1

→

Set bit Counts	5	4	1 (no discard)	2	1 (no discarding)
more_value	1	1	0	1	0

```
max_value = 0
for i in range(31, -1, -1):          → O(32)
    setbits = 0                       31, 30, 29 ... 1
    for j in range(len(A)):           O(N)
        ele = A[j]
        if(ele & (1<<i))>1:
            setbits += 1
    if setbits >= 2:
        max_vale = max_vale | (1<<i)
        for j in range(len(A)) :      O(N)
            if A[j] & (1<<i) == 0:
                A[j] = 0
```

TC : $(32 \times (N + N))$

$\Rightarrow (32 \times 2N)$

TC $\Rightarrow$ O(N)

SC $=$ O(1)

(*) Calculate the count of pairs of which bitwise AND is maximum.

→ following the same method above

→ get max_value number

→ Count the survives that are not zero.

→ return $\frac{x(x+1)}{2}$, the possible no. of pairs

Calculate the cout of of pairs

1 2 3 4 5

A B C D E

AB BC CD DE (10)

AC BD CE

AD BE

AE

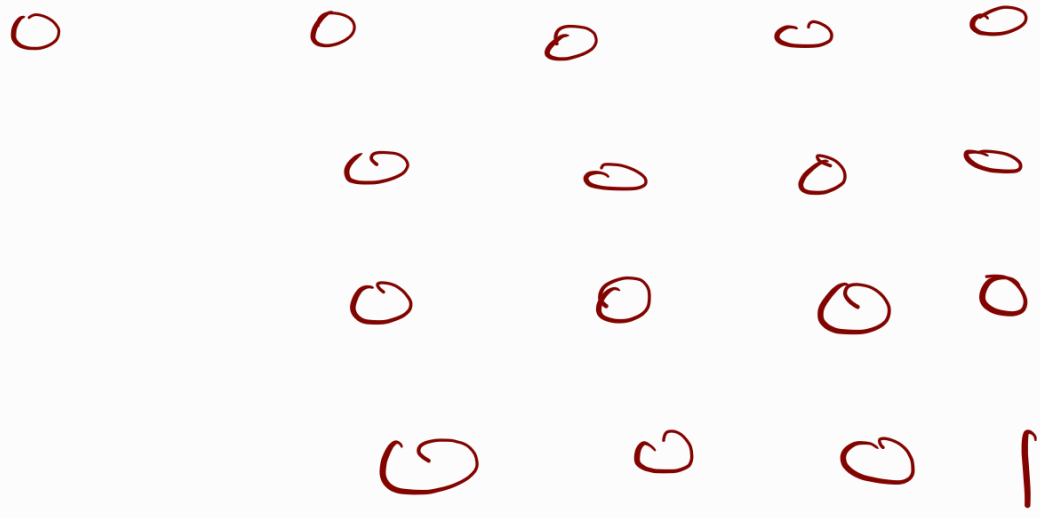
$$\Rightarrow \frac{5 \times (5-1)}{2} = \frac{5 \times 4^{2}}{2} = 10$$

Assignment problems :-

1. Given in[] A
   - every elements appears thrice
   - except one, which occurs once.

⇒ Find that element that does not appears thrice ?

A = [0 0 0 1]



1. Count set bits in each bit index.

ele binary form

0 = 1 0 0 0

0 = 1 0 0 0

0 = 1 0 0 0

1 = 0 0 0 1

---

setbits count } 3 0 0 1 1.3 = 1

32 bit times

```
if n%3 == 0:
    0%3 == 0:
    1%3 != 0
    # set ith bit
    Ans = Ans | (1<<i)
```

Ans = 0

return Ans

0 0 0 0

(1) 0 0 0 1

0 0 0 1

↓

1

(2) Single number II

int A[]

⇒ two integers appears once

A [X, X, 3, X, X, 4]

Ans = ans ^ A[i]

0 0 1 1 = 3

(^) 0 1 0 0 = 4

0 1 ^ 1 ⇒ 7

0th

0th index bit set

3 ✓

X

X

0th index bit unset

4 ✓

X ✓

X ✓

Sum of XOR of all pairs.

⇒ int [ ] A

⇒ find sum of bit wise XOR

⇒ return (ans % (10^9+7))

A = [1, 2, 3] ans = 6

	3	2	1	0	
(1, 2)	0	0	0	1	= 1
(1, 3)	0	0	1	0	= 2
(2, 3)	0	0	1	1	= 3
	—	—	—	—	
x	y	xy	xy	xy	
set, un-set ⇒ 0	3	0 3	2 1	2 1	
no. pairs that ith bit set	0×8	0×4	2×2	2×1	
	0	0	4	2	

(1^2^3^4^5)

1^2 1^3 1^4 2^3 2^4 2^5 4^5

1^5

6 1 1 0 = 6

3 4 2

3 0 0 1 1

4 0 1 0 0

2 0 0 1 0

zeroCount = 2

OneCount = 1

pair =

zc oc zcoc zcoc zcoc

https://github.com/syed460/Learning/blob/main/Daily%20Notes/Bit_Manipulation_2/README.md

pair 0x2 2x2 2x2 2x2

0x8 2x4 2x2 2x1

8 + 4 + 2 = 14

"min XOR value"

↳ given array int [] A

↳ find the pair that has min XOR value.

↳ return it.

A = [0 2 5 7]

0 0 0 0 0
2 0 0 1 0
0 0 1 0 - 2

0 0 2
2 0 1 0
5 0 1 0 1
0 1 1 1 = 7

2 0 1 0 1
7 0 1 1 1
0 0 1 0 = 2

5 0 1 0 1
7 0 1 1 1
0 0 1 0 = 2

(0,2) (2,5) (5,7)
(0,5) (2,7)
(0,7)

Steps:

1. Sort the array.
   → because the closer values should come to gether to get the pair.
2. Perform 'XOR' operation all element by iterating the array.
3. Maintain the min result in each iteration.

A = [12 4 6 2]

[2 4] 6 12]

```
def findMinXor(A)
    A.sort()  ------> nlogn
    N = len(A)
    ans = float('inf')
    for i in range(N-1) - O(N)
        x = A[i] ^ A[i+1]
        ans = min(ans, x)
```

TC = O(N log N) SC = O(1)

Strange Equality

→ given int A

→ x, y num are defined as follows

1. x is "greatest number smaller" than A such that

A = 5

Ans x = 2

5 = 8

^ 10

0 1 0 1 5

^ 0 0 1 0 2

0 1 1 1 = 7

0 1 0 1 5

& 0 0 1 0 2

0 0 0 0 = 7

0 1 0 1 5

^ 1 0 0 0 8

1 1 0 1 = 13

0 1 0 1 5

& 1 0 0 0 8

1 1 0 1 = 13

0 1 0 1 → 5

0 0 1 0 → 2

0 0 0 0

0 1 0 1

& 0 1 0 1 0 2

0 0 0 0 = 0

arithmatic sum of A&B &

XOR of A, B relation.

$$(A + B) = (A \wedge B) + 2\,(A \& B)$$

should be 0

our condition:

A & B == 0

↳ all set bit in numba A

should be unset in B

1 & 0 = 0

0 & 1 = 0

1 & 1 = 1

Gratest small than ← A → smallest grater than

X_2 3 4 5 7 Y_8

①

X greatest smaller than A (LSB) ←

↳ chang all unset bits of A to 1

↳ iterate over A, and change unset bit to set

(*) 5 | 0 1 0 1 → 5
5 | 0 1 0 = 2
0 0 0 0 = 0

②

y smallest grater than A →

↳ just grater power of 2 than A

is ans.

↳ because

8 4 2 1
0 1 0 1 = 5

MSB
1 0 0 0 = 8
0 0 0 0 = 0

A → y

1 0 1 1 0 1
1 0 0 0 → 0 1 0 = 2
0 0 0 0 0 0

(X)

↳ find the number of bits in A

↳ if number of bit i.

(Ans = 1 << i)

Example :- A = 5 0 1 0 1

5 = " 0 1 0 1 "

Find y

bitsCount = 0

N = A

while (N > 0) N = N >> 1

↓ bitsCount += 1

y = (1 << bitsCount)

find x

x = 0

for i in range(bitsCount)

if (A[i] & (1<<i)) == 0 # setbit

↓ x = x | (1<<i)

return [x, y]

```
  0 1 0 1
& 1 0 1 0
-------
  0 0 0 0
```

TC : O(bitsCount)

O(N)

SC : O(1)

Problem: SUBARRAY OR

↳ given int[] A of size N

↳ The value of Subarray is defind as "BITWISE OR" of all elements in it.

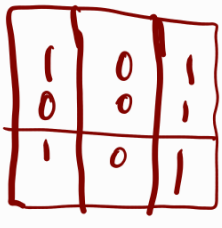
↳ Return the sum of all subarray of "A % 10^9+7".

$1 \le N \le 10^5$

$1 \le A[i] \le 10^8$

$A = [1, 2, 3, 4, 5]$

$\frac{N(N+1)}{2} \Rightarrow \frac{5 \times 6}{2} = 15$



1. For every Subarray OR

⇒ Brute force $O(N^3)$

⇒ Carry forward approch: $O(N^2)$

⇒ Brute force approch of Bitwise:

1. Traverse all subarray $\left(\frac{N(N+1)}{2}\right)$
   - 1.1 For each subarray
     - ↳ take bitwise OR of that subarray
   - 1.2. add it to the ans_sum.

TC: $= \left(\frac{N(N+1)}{2}\right)$

$= \frac{N^2+N}{2}$

$= N^2 + N/2$

$= O(N^2)$

$A = [1, 2, 3, 4, 5]$

Number of subarrays of Array.

of index 0 = N

of index 1 = N-1

of index 2 = N-2

⋮

1

Total number of subarray of A

$\Rightarrow N + (N-1) + (N-2) + \ldots 1$) revised the order

$\Rightarrow 1 + 2 + 3 \ldots N$

$\Rightarrow$ It is simple arithmatic progression with common difference of 1

$\Rightarrow$ It has number of elements N

It is sum of all natural numbers.

$\Rightarrow$ Formula for "sum of N natural number"

$\Rightarrow \frac{N(N+1)}{2}$

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

1 ⟶ 1 = 1

1 2 ⇒ 1 | 2 = 3

1 2 3 ⇒ 1 | 2 | 3 = 3

1 2 3 4 ⇒ 1 | 2 | 3 | 4 = 7

1 2 3 4 5 ⇒ 7 | 5 = 7

2 ⟶ 2 ⇒ 2

2 3 ⇒ 2 | 3 = 3

2 3 4 ⇒ 3 | 4 = 7

2 3 4 5 ⇒ 7 | 5 = 7

3 ⟶ 3 ⇒ 3

3 4 ⇒ 3 | 4 = 7

3 4 5 ⇒ 7 | 5 = 7

4 ⟶ 4 ⇒ 4

4 5 ⇒ 4 | 5 = 5

5 ⟶ 5 ⇒ 5

7 1	1	0	0	0	1	1	1	

13

total Subarr. 15

Zeros = 2

Zero bits = 1

Ones = 15 − 2 = 13

13

num. of Sub array = 15 - 2 = 13 1's

0th bit index ⇒ 13 × 2^0 = 0 1 1 0 1

1st bit index = 15 - 3 = 12 1's

⇒ 12 × 2^0 = 0 1 1 0 0

2nd bit index = 15 - 3 = 12 1's

⇒ 12 × 2^1 = 0 1 1 0 0

Steps:

1. calculate number of subarray of A.

$$\frac{N(N+1)}{2}$$

2. Itere every bit of 32

for each elements ith bit

2. calculate number of zeros present and if zeros in continously calculate possible its contribution all subarrays.

0 0 1 } 2 ⇒ $\frac{2(2+1)}{2} = 3$

0 0 0 } 3 ⇒ $\frac{3(3+1)}{2} = 6$

9 zeroes possible

15 - 9 =

N = len(A), ans = 0, totalSubarray = $\frac{N(N+1)}{2}$

for i in range(32)

zerosCount = 0

zerobitsCount = 0

for j in range(N)

if A[j] & (1 << i) == 0

zerosCount += 1

else:

zerobitsCount += $\frac{N(N+1)}{2}$

zeroCount = 0

zerobitsCount += $\frac{N(N+1)}{2}$

temp = (totalNumSubarray − zerobitsCount)

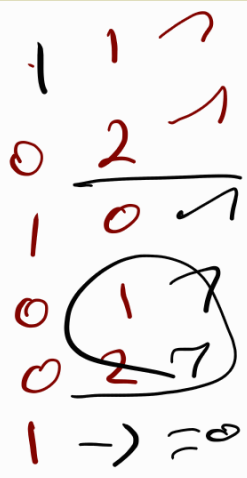
T(1's)

SUBARRAY OR

subarrayOR = (temp × 2^i)

ans += subarrayOR

return ans



next Problem: Find Two Missing Numbers

1. Given int[] A with distict elements in the range of $[1, N+2]$

$N = 3$ $\quad [1, , 3, , 5]$: Ans = 2, 4

2. Find the missing two numbers in A

$1 \leq N \leq 10^5$

$1 \leq A[i] \leq N+2$

Ex: $A = [3, 2, 4]$, $N = 3$

Brute force approch: $O(N^2)$

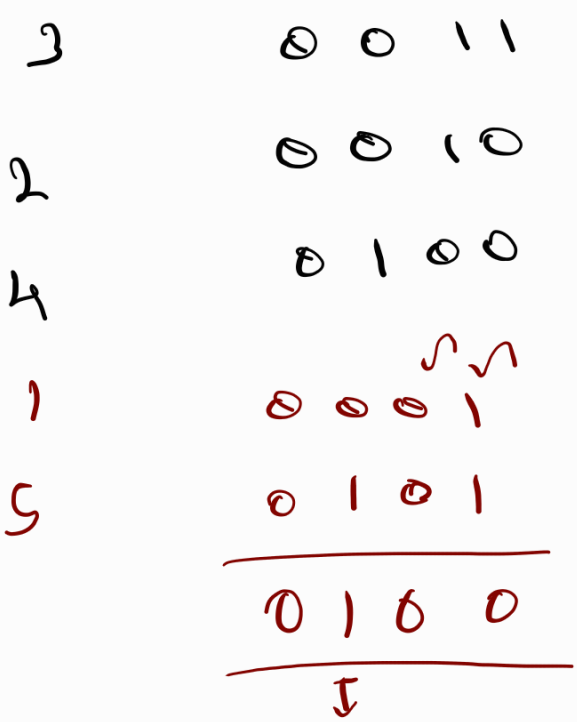
1. Iterate through $(N+2)$
2. check if i is present in A
2.1 else append it to ans
3. once len(Ans) has 2 return it

Sorting aprnch: $O(n \log n)$

1. sort the array
2. iterate the arry and update ans.

Hash mpping: $O(N)$ SC: $O(N)$

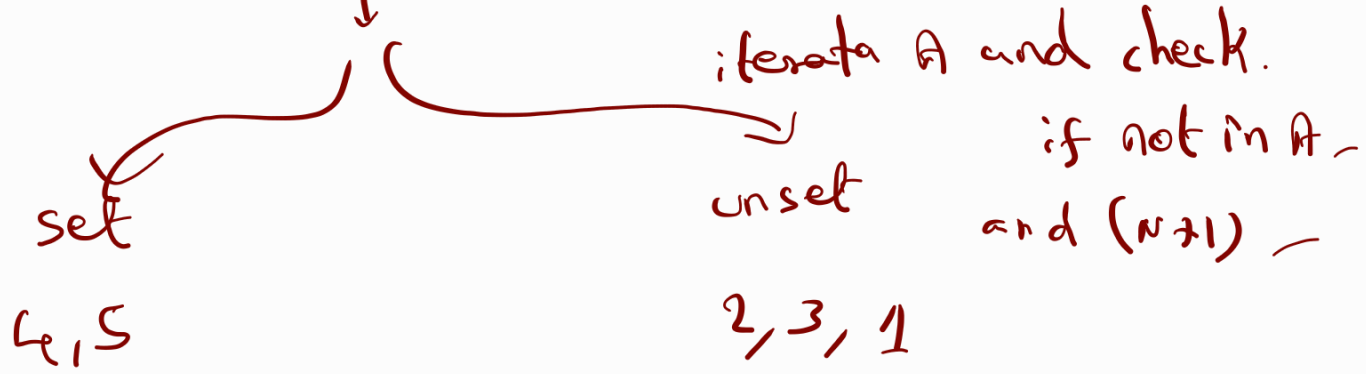
Bitwise approch: using 'XOR' operation.



N = 3

(1, N+2) = (1, 5)

Ans = 1, 5



1. get all XOR of array A and
   (1 to N+2)
   Due to the XOR property Duplicate will be - removed
2. From the result we need to extract the Two missing elements.
3. find the First setbit form resulting number
4.

```
x = 0
for i in range(N)
    x = x ^ A[i]
for i in rang(N+3)
    ele = i+1
    x = x ^ ele
# extract Two element from 'x'
# identify First set bit in x
idx = 0
for i in range(32)
    if (x & (1<<i)) > 0: # set bit
        idx = i
        break
# find two groups
setbitGroup = 0
unsetbit group = 0
for i in rang(N):
    if(A[i] & 1(<<idx)) > 0: # set bit
        setbit Group = setbitGrp ^ A[i]
    else
        unsetbitGroup = unsetbitGroup ^ A[i]
for i in range (N+3)
    ele = i+1
    if (ele & (1<<i)) > 0: # set bit
        setbitGroup = setbitGroup ^ ele
    else
        unsetbitGroup = unsetbitGroup ^ ele
ans = [setbitGroup, unsetbitGroup]
ans.sort()
return ans.
```

TC = O(N)

SC = O(1)

https://github.com/syed460/Learning/blob/main/Daily%20Notes/Bit_Manipulation_2/README.md