

AGENDA:

DSA: Bit Manipulations

Truth Table for Bitwise Operators

Basic Bitwise Operator Properties

Problem 1: Single Number

Left Shift Operator (\ll)

Power of Left Shift Operator

Setting the i -th Bit

Toggling the i -th Bit

Checking the i -th Bit

Unsetting the i -th Bit

Problem 2: Count the total number of SET bits in N

Problem 3: Single Number 3



Bit-wise Operators : $\&, |, ^, \sim, \ll, \gg$

0 \longrightarrow F unset OFF low voltage

1 \longrightarrow T set ON high voltage

a	b	a&b	a b	a^b	$\sim a$
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

if both a & b are 0
else 1

NOT

1 if a & b are 1, 0 otherwise

same same
zero game

power of 2 \longrightarrow 1, 2, 4, 8, 16, ...
 $2^0, 2^1, 2^2, 2^3, 2^4, \dots$

11 \longrightarrow

$$2^3 + 2^1 + 2^0$$

$$8 + 2 + 1$$

$$\begin{matrix} \textcircled{1} & \textcircled{0} & \textcircled{1} & \textcircled{1} \\ 2^3 & 2^2 & 2^1 & 2^0 \end{matrix}$$

2	22
2	11
2	5
2	2
2	1
	0

0
1
1
0
1

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

$$16 + 4 + 2 = \underline{\underline{22}}$$

$\text{bin}(11) \longrightarrow$



Basic Properties

1. Even / Odd Number →

$$A \& 1 == 1$$

odd

$$A \& 1 == 0$$

even

11 →	1	0	1	1
&	0	0	0	1
<hr/>				
	0	0	0	1
<hr/>				

10 →	1	0	1	0
&	0	0	0	1
<hr/>				
	0	0	0	0
<hr/>				

2. $A \& 0 \rightarrow 0$ $\forall A$



3. $A \& A \rightarrow A$

$$\begin{array}{r} A \quad 10110 \\ \text{ⓐ} \quad A \quad 10110 \\ \hline \quad \quad 10110 \\ \hline \end{array}$$

4. $A | 0 \rightarrow A$

$$\begin{array}{r} A \quad 10110 \\ | \quad 00000 \\ \hline \quad \quad 10110 \\ \hline \end{array}$$



5. $A | A \rightarrow A$

$$\begin{array}{r} A \quad 10110 \\ | \quad A \quad 10110 \\ \hline \quad \quad 10110 \\ \hline \end{array}$$

6. $A \wedge 0 \rightarrow A$

$$\begin{array}{r} A \quad 10110 \\ \wedge \quad 00000 \\ \hline \quad \quad 10110 \\ \hline \end{array}$$



7. $A \wedge A \rightarrow 0$

$$\begin{array}{r} \\ A \\ \wedge A \\ \hline \\ \hline \end{array}$$

$$2 \wedge 2 = 0$$

$$11 \wedge 11 = 0$$

Commutative Property →

Commutative

$$A \& B == B \& A$$

$$A | B == B | A$$

$$A \wedge B == B \wedge A$$

Associative Property →

order in which operations are done
doesn't matter

$$(A \& B) \& C == A \& (B \& C)$$

$$(A | B) | C == A | (B | C)$$

$$(A \wedge B) \wedge C == A \wedge (B \wedge C)$$



< Question- 1 > : Evaluate the expression: $a \wedge b \wedge a \wedge d \wedge b$

$$\begin{aligned}
 & a \wedge b \wedge a \wedge d \wedge b \\
 \Rightarrow & a \wedge a \wedge b \wedge b \wedge d \\
 \Rightarrow & 0 \wedge 0 \wedge d \\
 \Rightarrow & \underline{\underline{d}}
 \end{aligned}$$

< Question- 2 > : Evaluate the expression: $1 \wedge 3 \wedge 5 \wedge 3 \wedge 2 \wedge 1 \wedge 5$

$$\begin{aligned}
 & 1 \wedge 3 \wedge 5 \wedge 3 \wedge 2 \wedge 1 \wedge 5 \\
 \Rightarrow & 1 \wedge 1 \wedge 3 \wedge 3 \wedge 5 \wedge 5 \wedge 2 \\
 \Rightarrow & 0 \wedge 0 \wedge 0 \wedge 2 \\
 \Rightarrow & \underline{\underline{2}}
 \end{aligned}$$

Value of $120 \wedge 5 \wedge 6 \wedge 6 \wedge 120 \wedge 5$ is - 0



< **Question** > : Given arr[N] where every element is present twice except one unique element.
Find that unique element.

$A = [4, 5, 5, 4, 1, 6, 6]$

Output = 1

Bruteforce \rightarrow Create a counter of A and key with
frequency == 1 is my ans



Bruteforce

Create a counter out of the given A itself and return the key with freq == 1

```
import collections
```

```
A = [7, 5, 5, 1, 7, 6, 1, 6, 4]
```

```
def single_number_1(A):
```

```
    counter = collections.Counter(A)
```

```
    for k, freq in counter.items():
```

```
        if freq == 1:
```

```
            return k
```

```
    return -1
```

```
print(single_number_1(A))
```

```
# TC: O(N)
```

```
# SC: O(N)
```



Optimised XOR ALL

A = [7, 5, 5, 1, 7, 6, 1, 6, 4]

```
def single_number_1(A):
```

```
    xor = 0
```

```
    for val in A:
```

```
        xor ^= val
```

```
    return xor
```

```
print(single_number_1(A))
```

```
# TC: O(N)
```

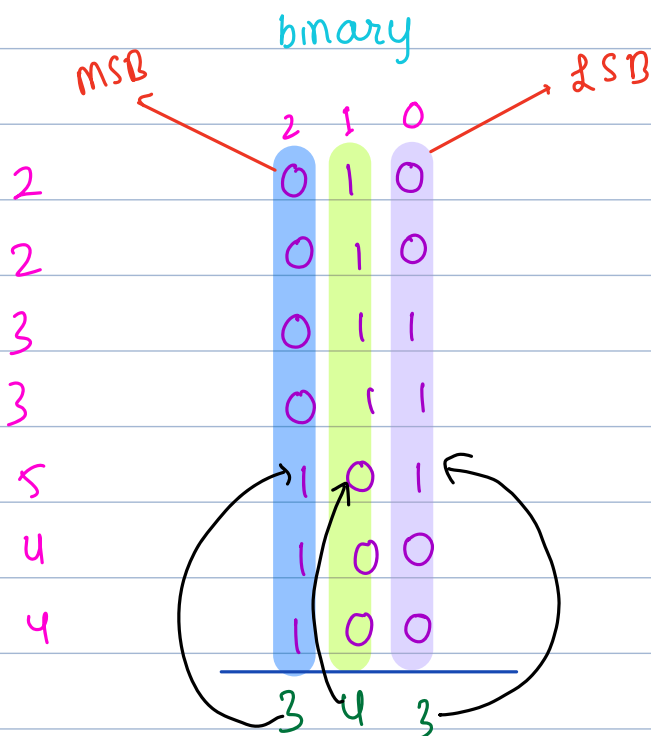
```
# SC: O(1)
```

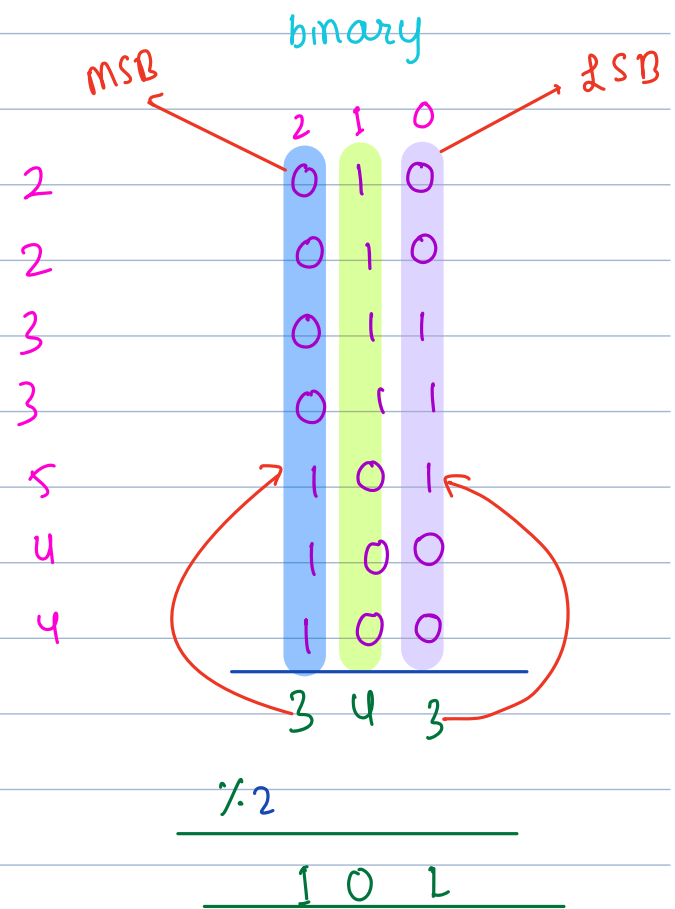
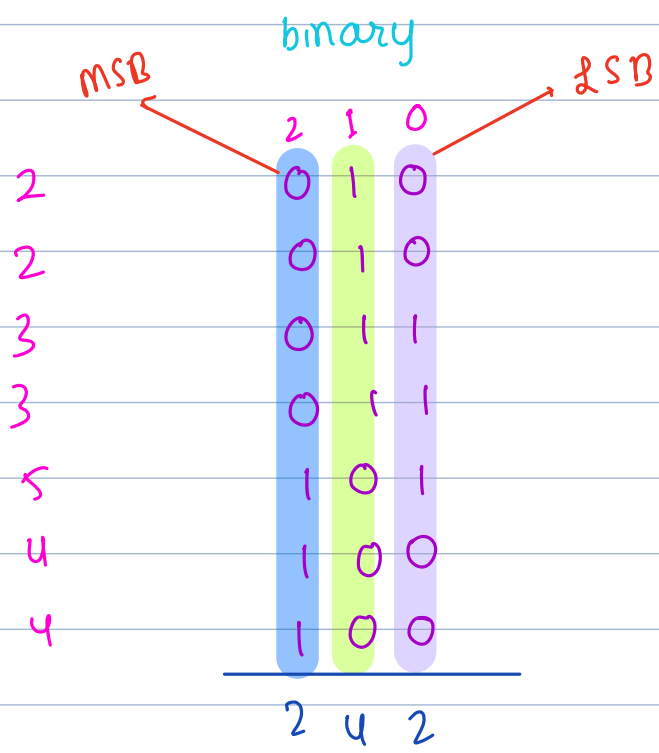


4

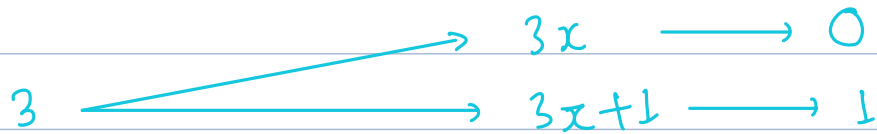
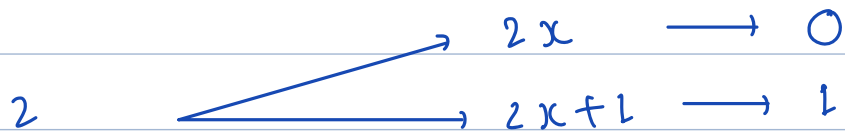
ans = 5

A = 2 2 3 3 5 4 4





every element appears



A = 2 2 2 3 3 3 5 4 4 4

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

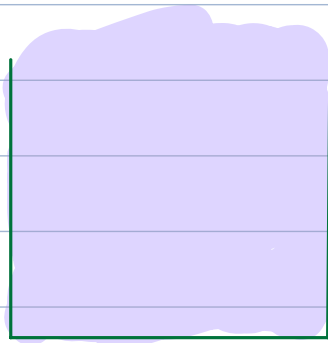
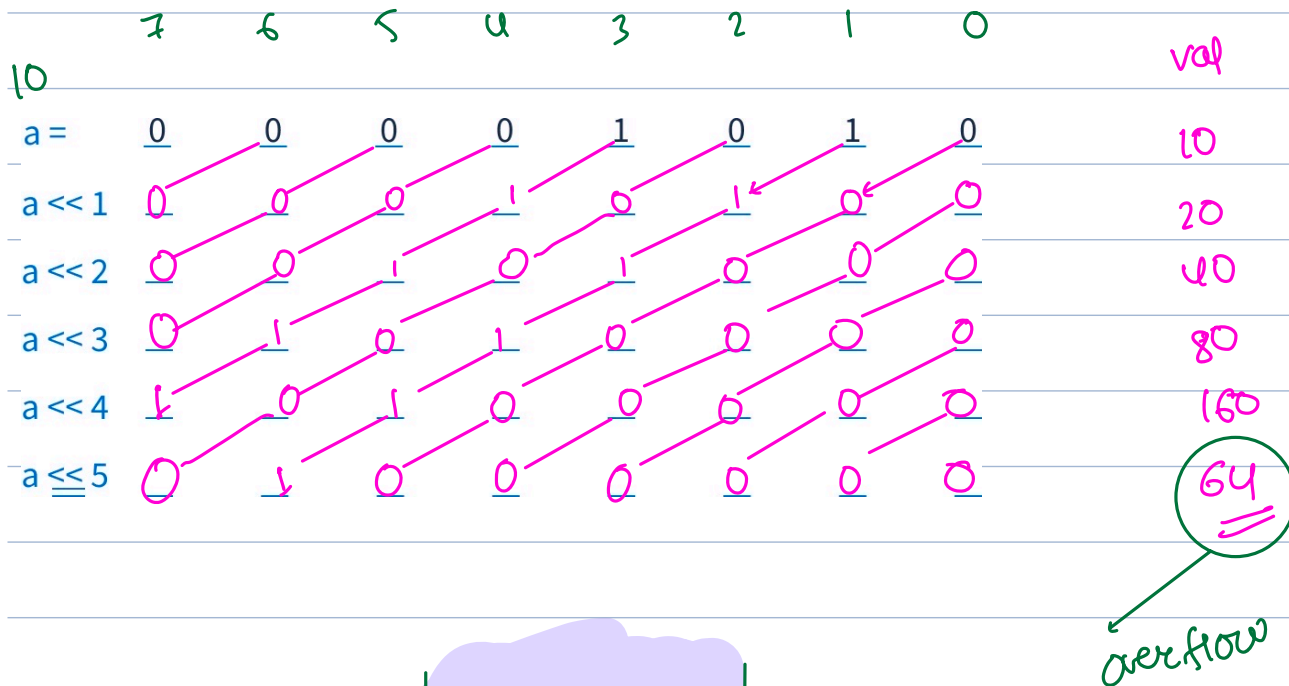
multiples of 3

7.3 1 0 1



Left Shift Operator (<<)

8 bit rep



Note → NOT applicable for python

⇒ left shift by 1 ⇒ multiply by 2 = 2^1
 left shift by 2 ⇒ multiply by 4 = 2^2
 left shift by 3 ⇒ multiply by 8 = 2^3
 ⋮

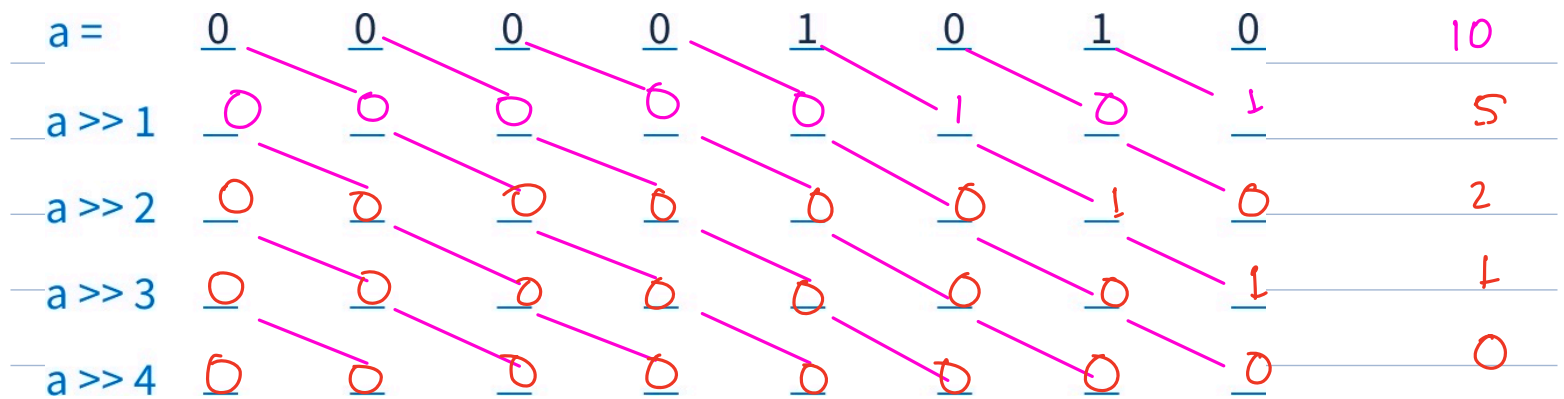
left shift by n ⇒ multiply by 2^n

$$1 \ll 3 = 1 * 2^3 = 8$$

⇒ 6 5 4 3 2 1 0
 0 0 0 0 0 0 1 ⇒ 0 0 1 0 0 0 8



Right Shift Operator (>>)



int division

Right shift by 1 \Rightarrow divide by 2

Right shift by 2 \Rightarrow divide by 2^2

Right shift by 3 \Rightarrow divide by 2^3

Right shift by 4 \Rightarrow divide by 2^4

Right shift by 5 \Rightarrow divide by 2^5

⋮

⋮

Right shift by $n \Rightarrow$ divide by 2^n

$$10 \gg 2 = \frac{10}{2^2} = \frac{10}{4} = 2$$

$$4 \ll 3 = 4 * 2^3 = 4 * 8 = 32$$

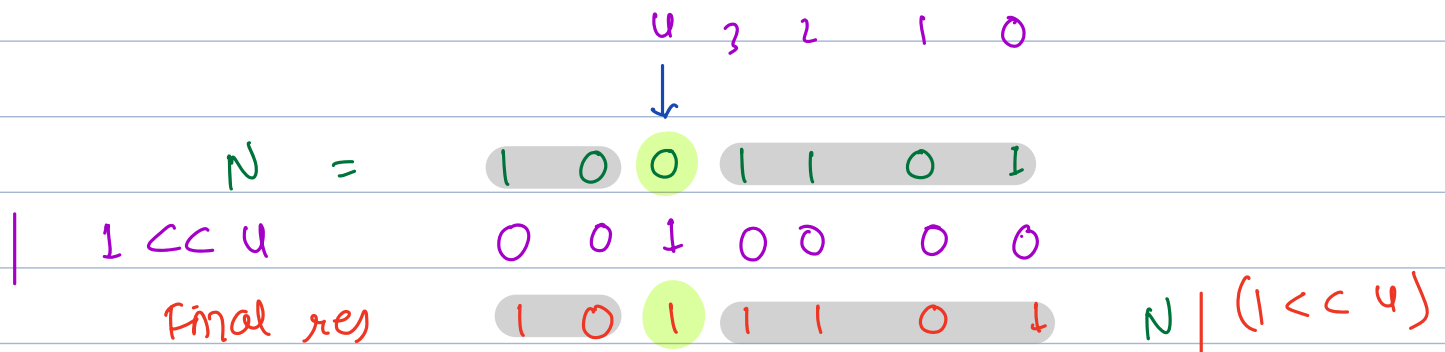


Power of Left Shift Operator

Break: 9:35

If you want to set i^{th} bit

1. OR Operator $\rightarrow N | (1 \ll i)$



Setting a bit \Rightarrow making it 1

Set i^{th} bit $\Rightarrow N | (1 \ll i)$

I want to toggle the i^{th} bit

	6	5	4	3	2	1	0
N	0	1	0	1	1	0	1
$\wedge 1 < i < 3$	0	0	0	1	0	0	0
	0	1	0	0	1	0	1

toggle 3rd bit

$0 \rightleftharpoons 1$

	6	5	4	3	2	1	0
N	0	1	0	0	1	0	1
$\wedge 1 < i < 3$	0	0	0	1	0	0	0
	0	1	0	1	1	0	1

Toggle i^{th} bit $\Rightarrow N \wedge (1 < i < 3)$



How to check if the i^{th} bit is 0 or 1?

	7	6	5	4	3	2	1	0
N =	1	1	0	1	0	1	1	0
$1 \ll 4$	0	0	0	1	0	0	0	0
	0	0	0	1	0	0	0	0

	7	6	5	4	3	2	1	0
N =	1	1	0	0	0	1	1	0
$1 \ll 4$	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	0	0

Tell me if i^{th} bit is 0 or 1?

if $N \& (1 \ll i) == 0$:

i^{th} bit is 0

else

i^{th} bit is 1

```
def check_bit ( N , i ) :
    return N & ( 1 << i ) > 0
```



Unset ith bit

[47]



Code to unset the i-th bit

```
N = 0b11001101
```

Use whatever we have learnt above

Unset 2nd bit in N

"""

if the ith bit is already 0/unset don't touch it

if the ith bit is set or 1 -> toggle it to make it 0

"""

```
def unset(N, i):
```

```
    if check_bit(N, i):
```

```
        return N ^ (1 << i) # toggle ith bit
```

```
    return N
```

```
N = 0b100110110
```

```
i = 12
```

```
pb(N)
```

```
pb(unset(N, i))
```



```
100110110
```

```
100110110
```

```
100110110
```

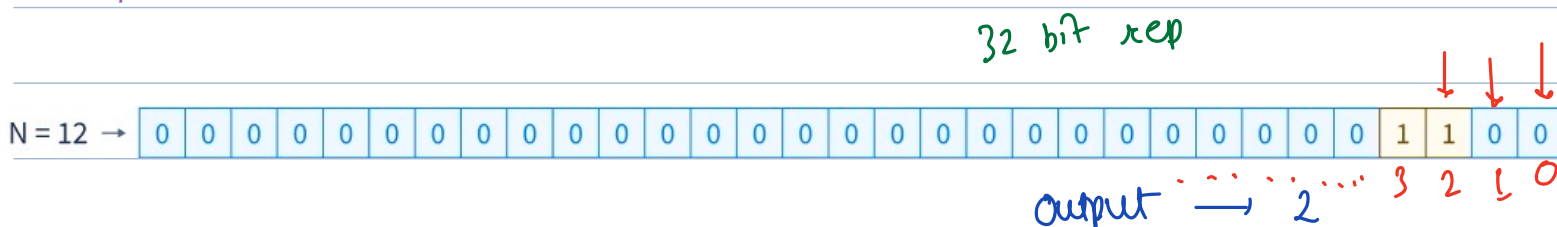


< Question > : Given an integer N. Count the set-bits in N.

$$N \leq 10^9$$

10^9

Example :



for all the bit position call check-bit fn

Code to count the total number of SET bits in N

N = 0b1001101100110

ans = 0

for i in range(32): → log(N)

if check_bit(N, i):

ans += 1

ans += check_bit(N, i) → O(1)

print(ans)

Tc: $O(\log N)$

int $\approx 2^{32}$ log
long $\approx 2^{64}$ log

• Single Element III

Given $arr[N]$, every element repeats twice except for 2 elements. Find the two unique elements.

$$\begin{aligned} \text{Eg: } A[6] &= \{ 3, 6, 4, 4, 3, 8 \} = 6, 8 \\ A[4] &= \{ 4, 9, 9, 8 \} = 4, 8 \end{aligned}$$

Bruteforce Hashmap

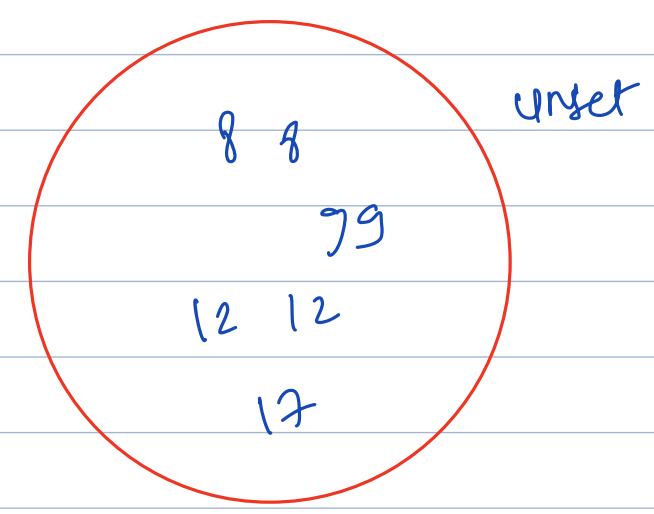
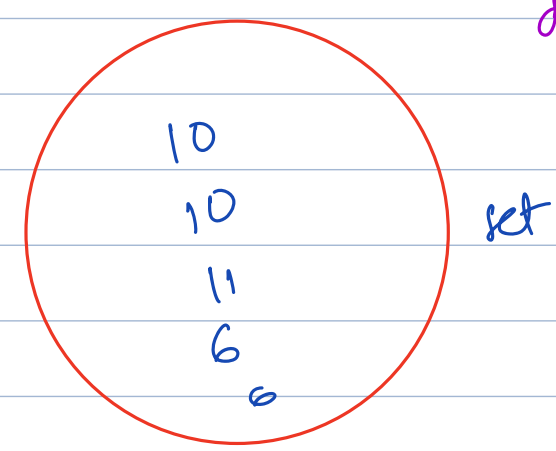
$A[12] = 10 \overset{01010 \uparrow}{10} \underset{01000 \downarrow}{8} \underset{01001 \downarrow}{8} \underset{01100 \uparrow}{9} \underset{01011 \downarrow}{9} 11 \overset{01100 \uparrow}{12} \underset{00110 \downarrow}{12} \underset{10001 \uparrow}{6} \underset{00110 \downarrow}{6} 17$

xor all $\longrightarrow 11 \wedge 17 \longrightarrow$

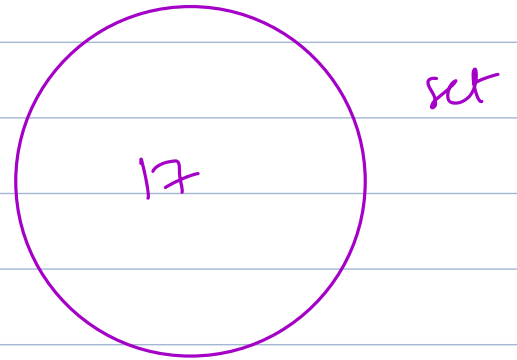
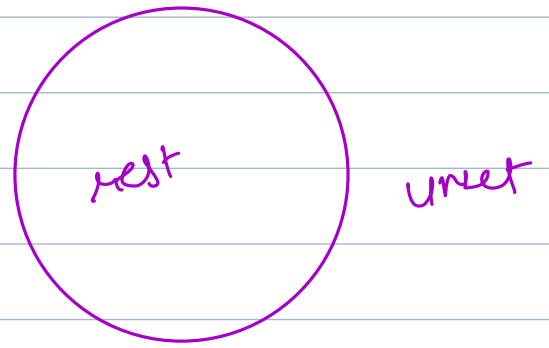
	4	3	2	1	0
	0	1	0	1	1
\wedge	1	0	0	0	1
	1	1	0	1	0

1st bit
 11 & 17 were diff

	4	3	2	1	0
10	0	1	0	1 [✓]	0
10	0	1	0	1 [✓]	0
8	0	1	0	0 [✓]	0
8	0	1	0	0 [✓]	0
9	0	1	0	0 [✓]	1
9	0	1	0	0 [✓]	1
11	0	1	0	1 [✓]	1
12	0	1	1	0 [✓]	0
12	0	1	1	0 [✓]	0
6	0	0	1	1 [✓]	0
6	0	0	1	1 [✓]	0
17	1	0	0	0 [✓]	1



	4	3	2	1	0
10	0	1	0	1	0
10	0	1	0	1	0
8	0	1	0	0	0
8	0	1	0	0	0
9	0	1	0	0	1
9	0	1	0	0	1
11	0	1	0	1	1
12	0	1	1	0	0
12	0	1	1	0	0
6	0	0	1	1	0
6	0	0	1	1	0
17	1	0	0	0	1



```
# Code to find the two unique numbers
```

```
A = [10, 10, 8, 8, 9, 9, 11, 12, 12, 6, 6, 17, -1, -1, 0, 0]
```

```
# xor all
```

```
xor = 0
```

```
for val in A:
```

```
    xor ^= val
```

} $O(N)$

```
# xor will contain the xor of 11 ^ 17
```

```
diff = -1
```

```
for i in range(32):
```

```
    if check_bit(xor, i):
```

```
        diff = i
```

```
        break
```

} $\log N$

```
pb(xor)
```

```
print(diff)
```

```
a = 0 # first unique
```

```
b = 0 # second unique
```

```
for val in A:
```

```
    if check_bit(val, diff):
```

```
        a ^= val # group set
```

```
    else:
```

```
        b ^= val # group unset
```

} $O(N)$

```
print(a, b)
```

TC: $O(N)$

SC: $O(1)$