1. One Odd Occurance.

↳ [1, 2, 3, 2, 3] ⇒ only one occurs once.

↳ $\boxed{0 \wedge x = x}$ $\boxed{x \wedge x = 0}$

so start with res = 0

$0 \wedge 1 \Rightarrow 1$

$1 \wedge 2 \Rightarrow 1 \wedge 2$

$1 \wedge 2 \wedge 3 \Rightarrow 1 \wedge 2 \wedge 3$

$1 \wedge 2 \wedge 3 \wedge 2 \Rightarrow 1 \wedge 3$

$1 \wedge 3 \wedge 3 \Rightarrow 1$

---

2. 2 odd Occurances.

↳ all even occurances except 2 numbers.

[5, 6, 10, 6, 6, 10, 3, 3]

1. XOR ⇒ $5 \wedge 6 \Rightarrow 0101 \wedge 0110 \Rightarrow 3$.

2.

→ Find XOR of all numbers

→ How to find the numbers from result ??

As we know the set in the XOR result means it's different in both numbers.

So consider any index which is a set and also consider which is unset.

and XOR them separately.

For Extracting the right most set bit which is last set bit

$k = res \ \& \ \sim(res - 1)$

3 ⇒ 11

↳ Any bit which is set. (For simplicity we can take the Right most / Last one which is set bit of the num.)

1: $5 \wedge 3 \wedge 3 = 5$ ✓

0: $6 \wedge 10 \wedge 6 \wedge 10 \wedge 6 = 6$ ✓

(3) Given a number, we need to Find XOR of numbers b/w 1 → N.

if we use a loop → O(n)

n = 1	1
n = 2	3
n = 3	0
n = 4	4
n = 5	1
n = 6	7
n = 7	0
n = 8	8

O(1)

⇓

⇒ if n % 4 == 0 :
  print (n)

n % 4 == 1 :
  print (1)

n % 4 == 2 :
  print (n+1)

n % 4 == 3 :
  print(0)

---

(4) Given L, R we need to compute XOR from L to R.

Consider L = 3
R = 5 ⇒ $3 \wedge 4 \wedge 5$

O(1)

~ XOR till 5 = $1 \wedge 2 \wedge 3 \wedge 4 \wedge 5$

XOR till 2 = $1 \wedge 2$

⇒ XOR till 5 $\wedge$ XOR till 2.

⇒ clear Last set Bit

1 1 0 1 1 0

⇒ 1 1 0 1 0 0

12 ⇒ 1100
11 ⇒ 1011
1000 ✓

3, 2
11
& 10
10
4, 2

4, 3
100
& 11
000
4, 0

→ Set Last unset Bit

4 → 100,
(or) 101 → 5
101

5 101
or · 110 (6)

For setting Last Bit → n | n+1

9 →
1001
1010
1011 → 11 ✓

→ Power set using bitwise . .

Str = 'abc'

n = 3

↳ no. of subsets = $2^n = 2^3 = 8$.

These sub sets can be mapped to Binary representation of numbers from 0 − 7.

	c b a				
0	000	{}	5	101	ca
1	001	a	6	110	cb
2	010	b	7	111	abc
3	011	ab			
4	100	c			

→ compute power of 2 (For running loop)

→ run a loop from $0 \to 2^n - 1$

```
for i in range (2^n):
    subset = []
    for j in range (n)
        if ((i & (1<<j))! = 0):
                add to subset
    add to powerset.
```

input set {1,2,3}

$\boxed{3}$

0		1		2	
0000 &	0001	0001 &	0001	0010 &	0001
0000 &	0010	0001 &	0010	0010 &	0010
0000 &	0100	0001 &	0100	0010 &	0100

None yields 1. ⇒ [ ]

a

b.

Longest consecutive 1's.

14 ⇒ $\underbrace{111}_{3}0$

$110\underbrace{1111}_{4}0$

Approach

1. Check if last bit is set or not

   if set :

   increase count

2. if not set ⇒ we encountered zero :

   Take max of Max Count & Current count

   assign to max

   & reset the Count

3. Right shift till the n != 0.