Numbers in bit

0 = 0]2
1 = 1]2
2 = 10]2
3 = 11
4 = 100]
5 = 101 | 4
6 = 110 |
7 = 111]
8 = 1000]
9 = 1001 |
10 = 1010 | 8
11 = 1011 |
12 = 1100 |
13 = 1101 |
14 = 1110 |
15 = 1111]
16 = 10000

28 =

11100

$\log 2^4 = 16$

28 − 16 = 12

$\log 12 = 3$

$2^3 = 8$

12 − 8 = 4

2 | 28
2 | 14
2 | 7

29 − 16 = 13

13 − 8 = 5

log

1) Bitwise &

- And on each bit
- 1 if both bit are 1
- else output is 0

* is used to quickly check if a number is odd or even. (x&1) = 1 odd
= 0 even

- Used to unset a bit at nth position.
  i) we use ~ (not) and << (left shift) to create the num that will reset the bit
  eg num = 1101
  we do 1<<4 = 1000
  ii) now ~ it so only nth bit is reset

∴ ~(1<<4) = 0111

iii now & with num

→ x & ~(1<<4)

```
 1101
 0111
------
 0101
------
 ↑
```

nth bit reset.

- check nth bit set or not
  → just & with 1 (with <<)
  → 1 = set
  0 - unset

2 Bitwise OR |

- OR operator gives 1 if any bit is 1 and
- and 0 if any both 0

☆ Set a bit in a number

→ OR with 1 to set the n^{th} bit. first like in and operator we will apply n|(1<<pos) first.

3 Bitwise XOR (^)

- XOR gives 1 if the two bit are different
- and gives 0 if the bit are same

i) num ^ 0 = num
ii) num ^ num = 0

Swap without Variable

a
b
a = a^b
b = b^a (b^a^b) → (a)
a = a^b (a^b^a) → b

Find duplicate element in array

Find missing element in array

Find two non repeating

remember that XOR can carry the value forward. That is if we take XOR of 10 value it will kind of remember them.

XOR Hint

→ Continuous
→ Consecutive
duplicate (Single)
→ pair (Single)