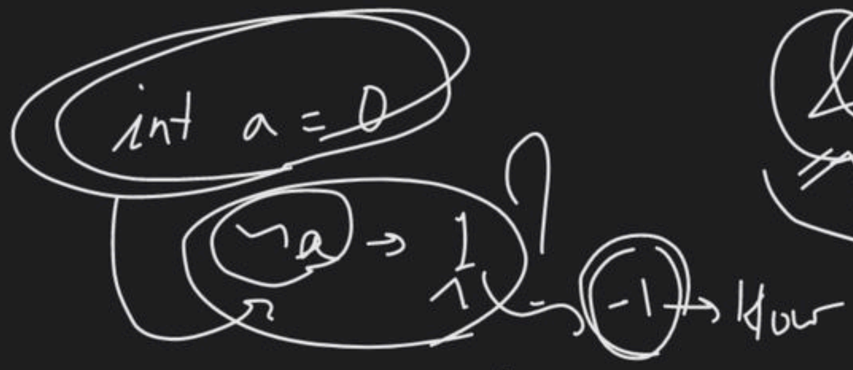
Bonus Class - Bit Manipulation

Special class

Love Babbar • July 7, 2023

Bitwise Operators:-

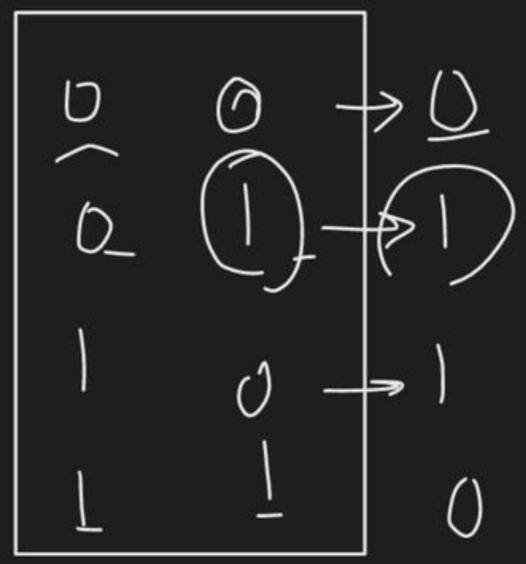


1 << 1

a	b	o/p
0	0	0
0	1	0
1	0	0
1	1	1

^ ^

a	b	OR o/p
0	0	0
0	1	1
1	0	1
1	1	1



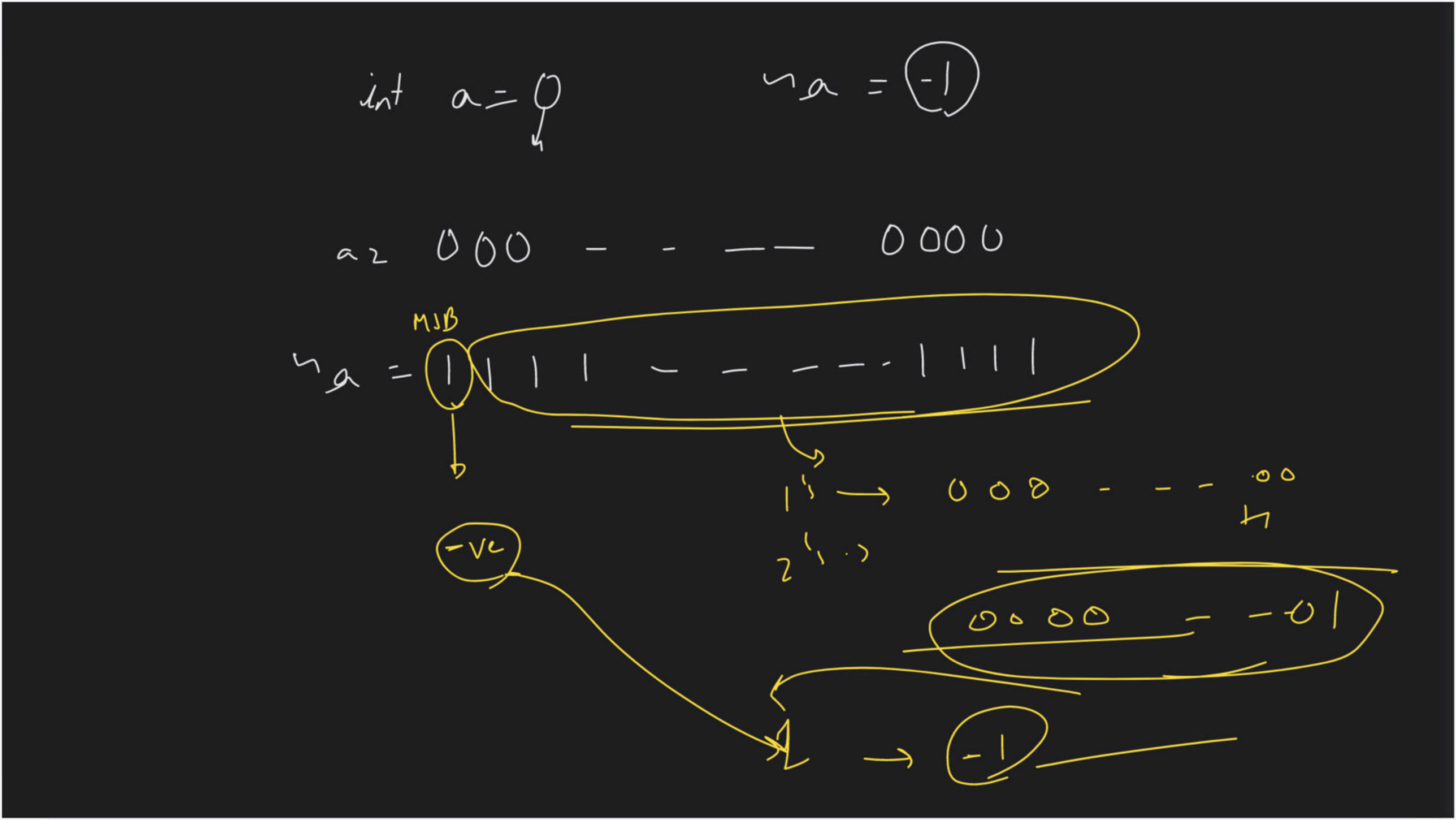
Decimal → Binary

Binary → Decimal = already

sign

1's comp

2's comp


int a = 0
~a = -1
a 2 000 - - —— 0000
MSB
~a = 1 1 1 1 - - - - - 1 1 1 1
1's → 000 - - - 00
2's
-ve
0000 - - 01
-1

→ Bitwise op → &, |, ^, ~, <<, >>

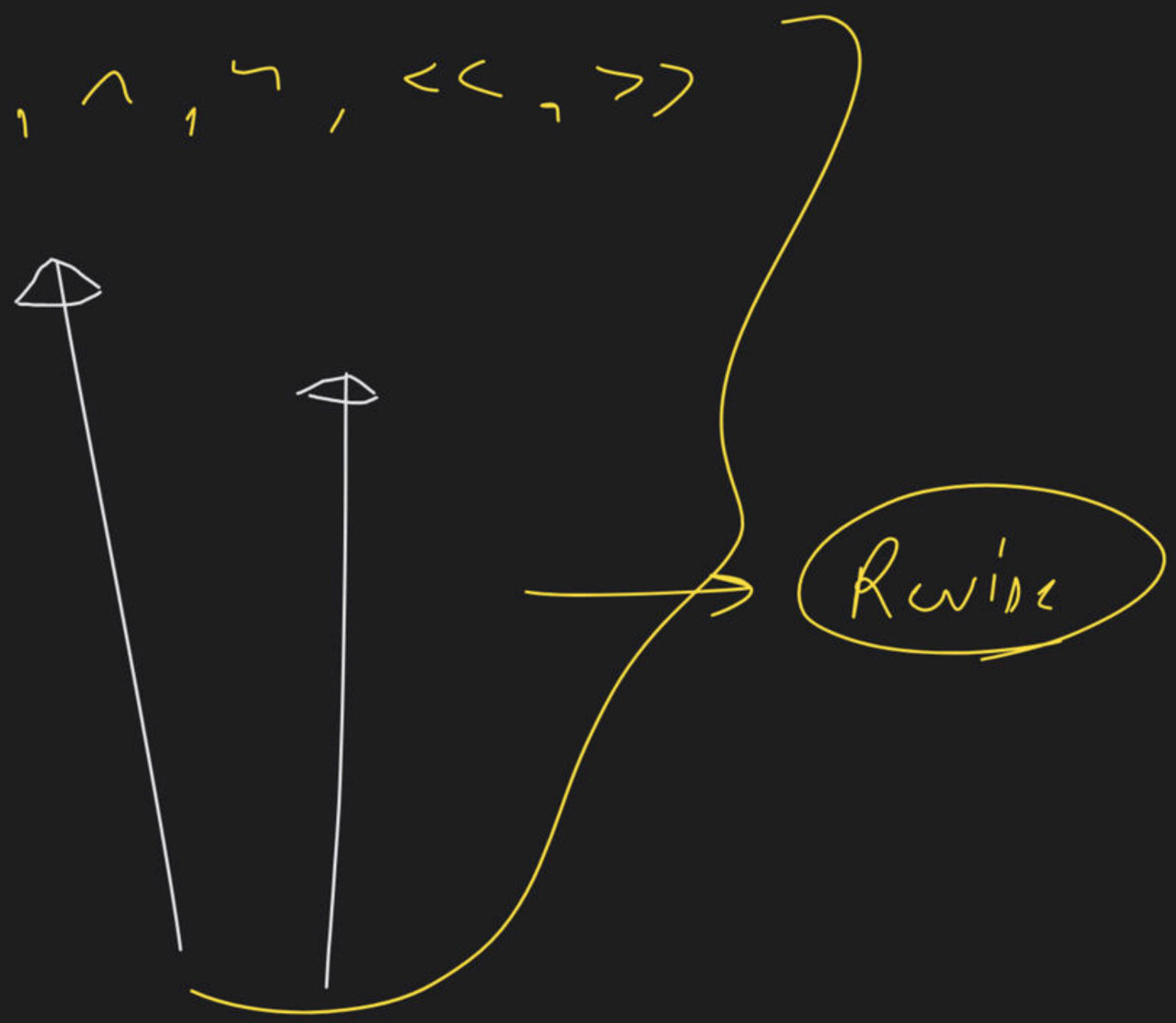
→ sign bit

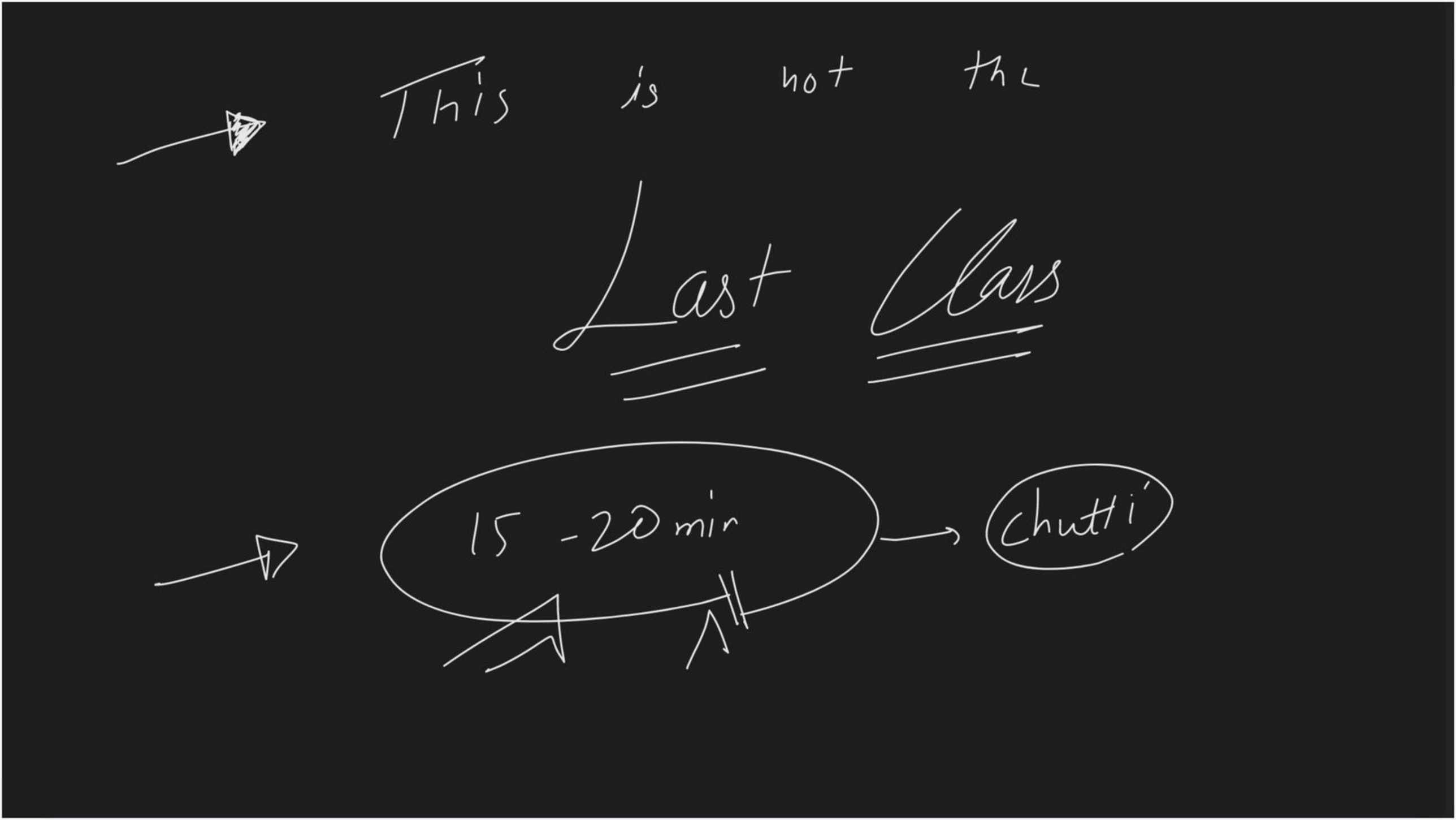
→ is comp

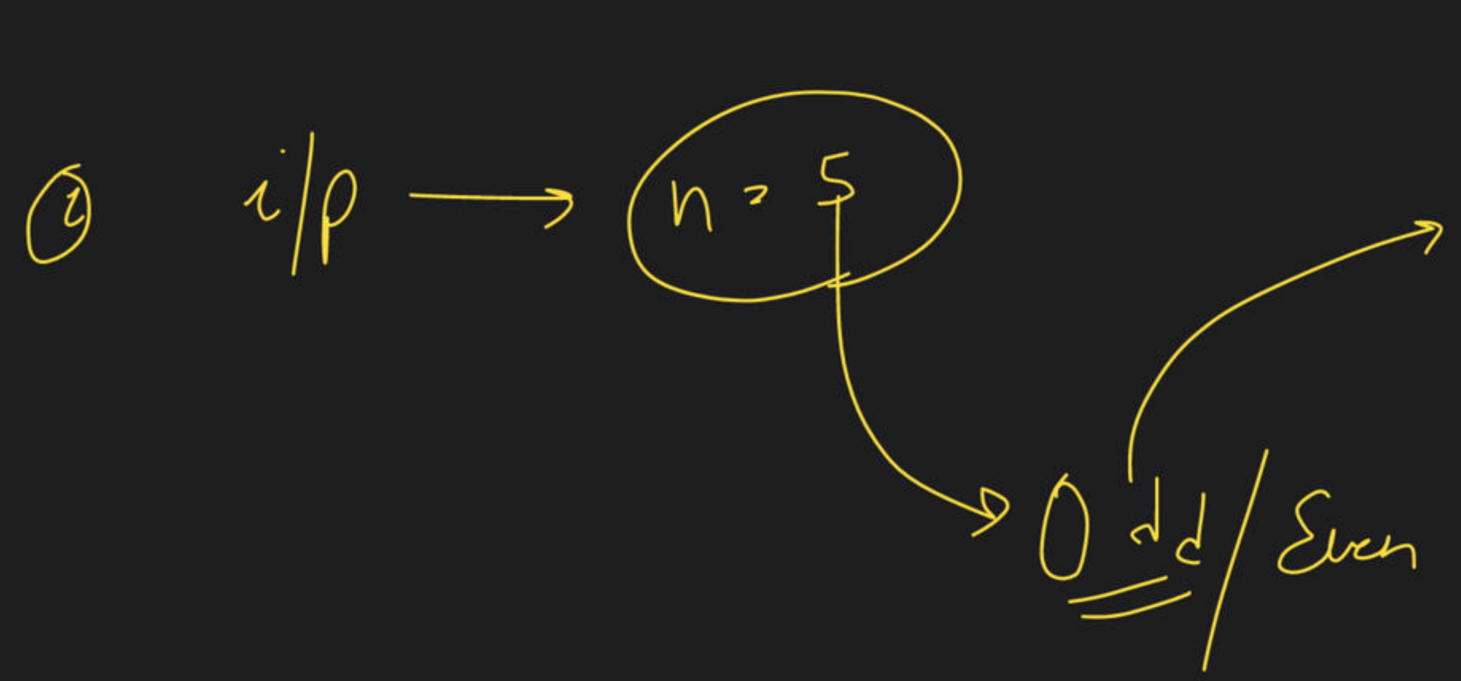
→ 2's comp

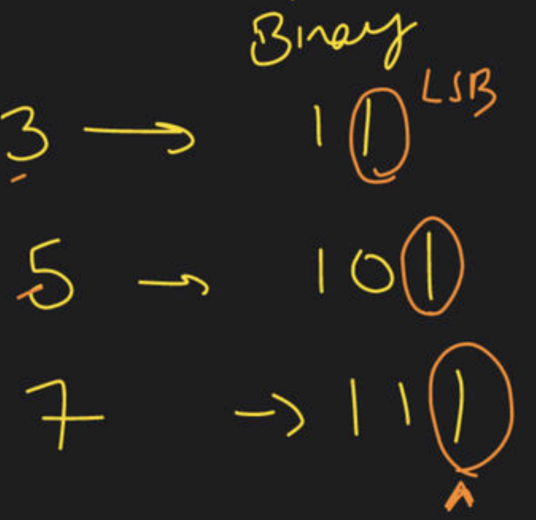
→ D → B

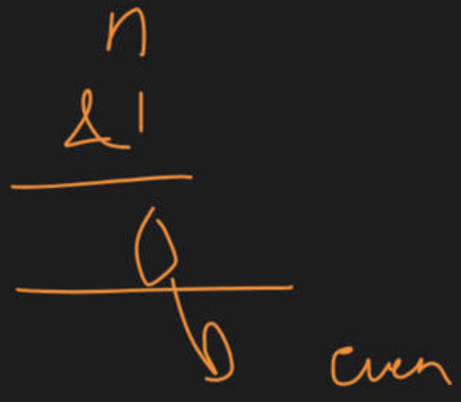
→ B → D

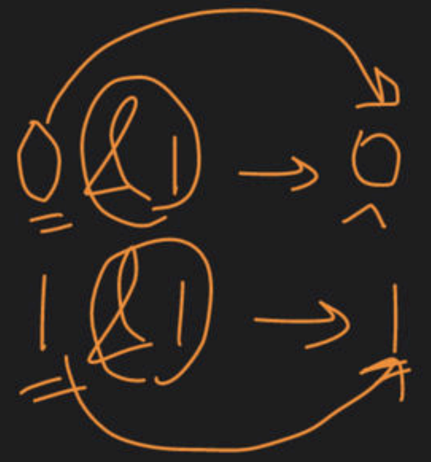



This is not the
Last Class
15 - 20 min
Chutti

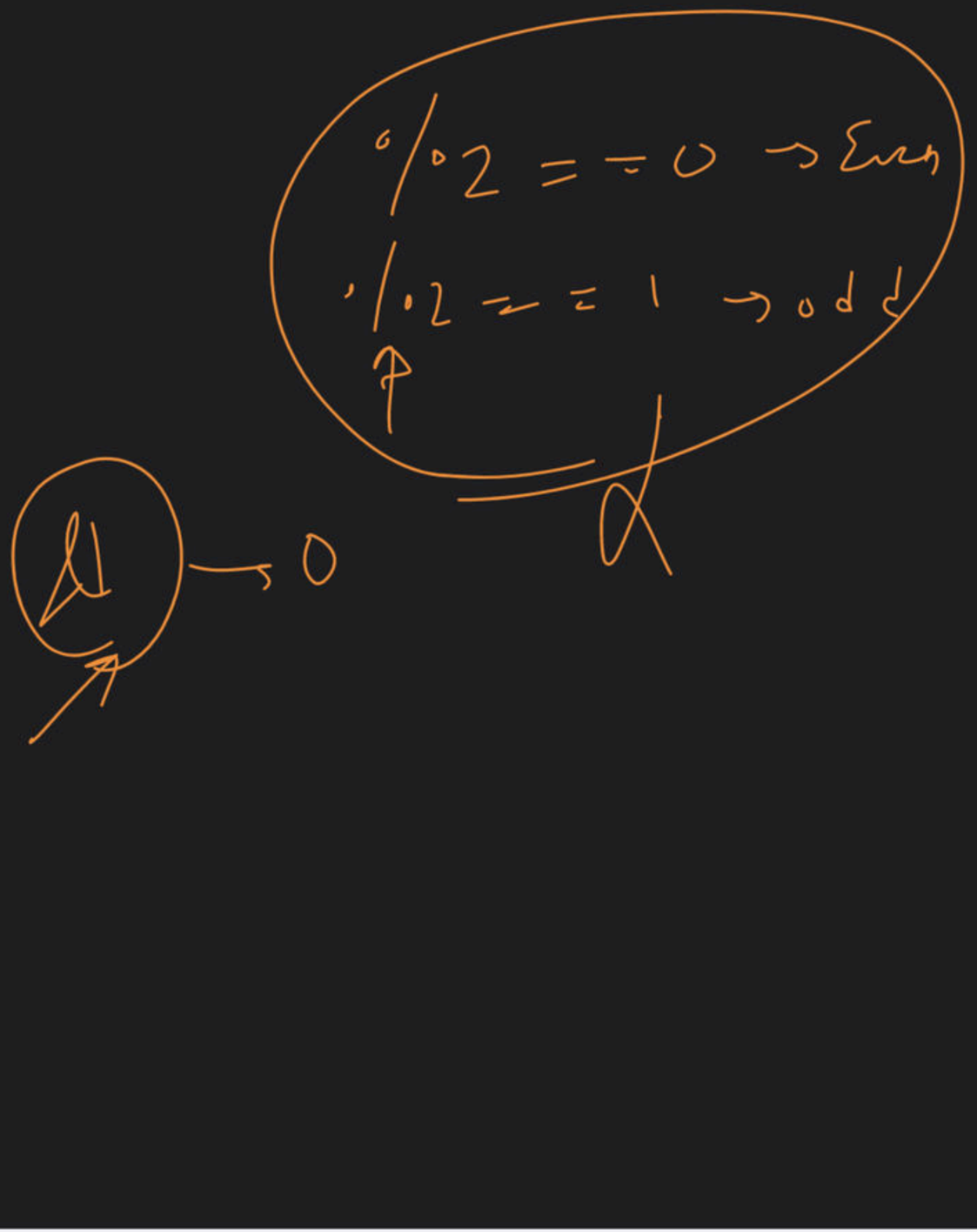

i/p
n = 5
Odd/Even

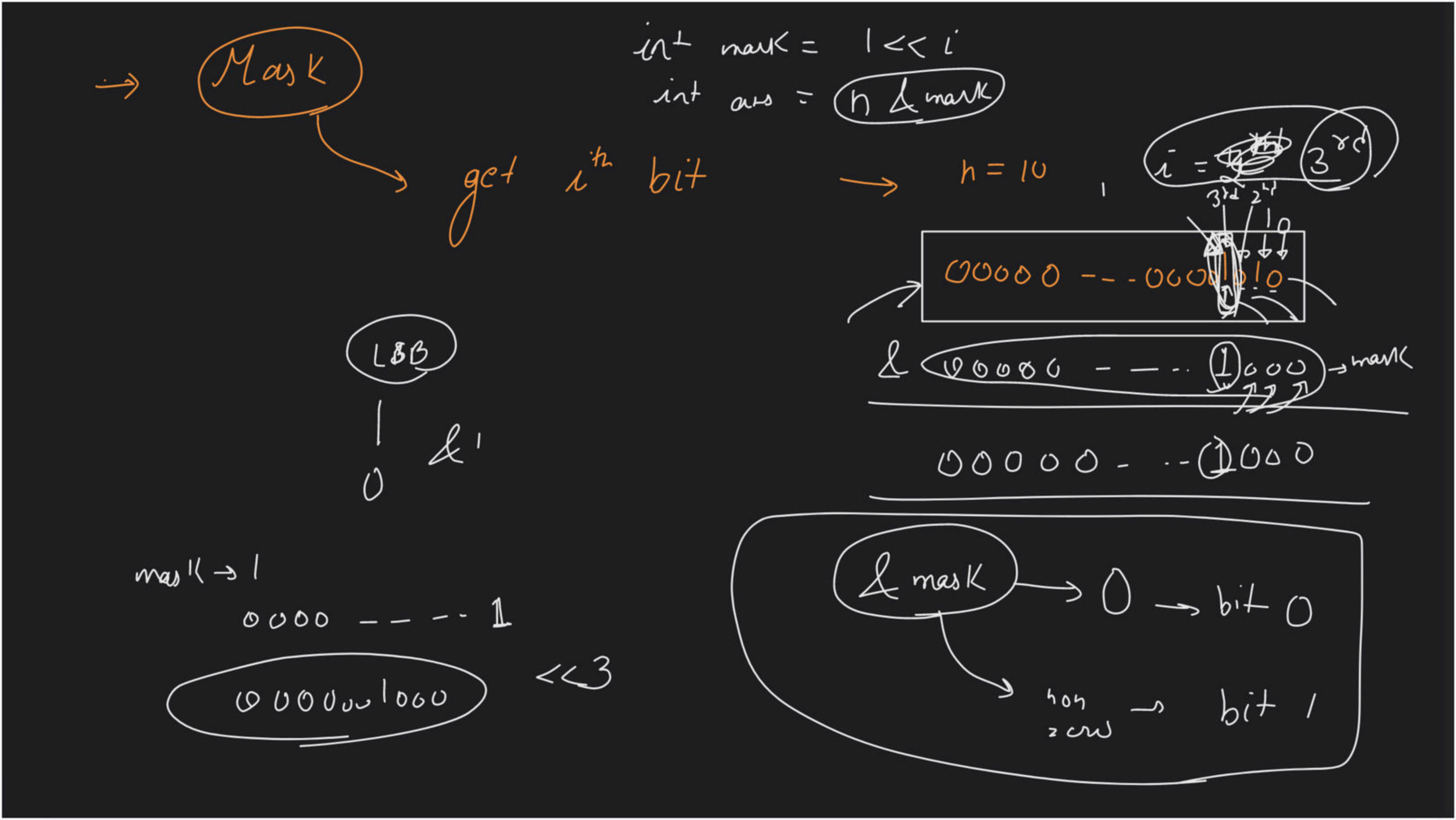

Binary
LSB
3 → 11
5 → 101
7 → 111

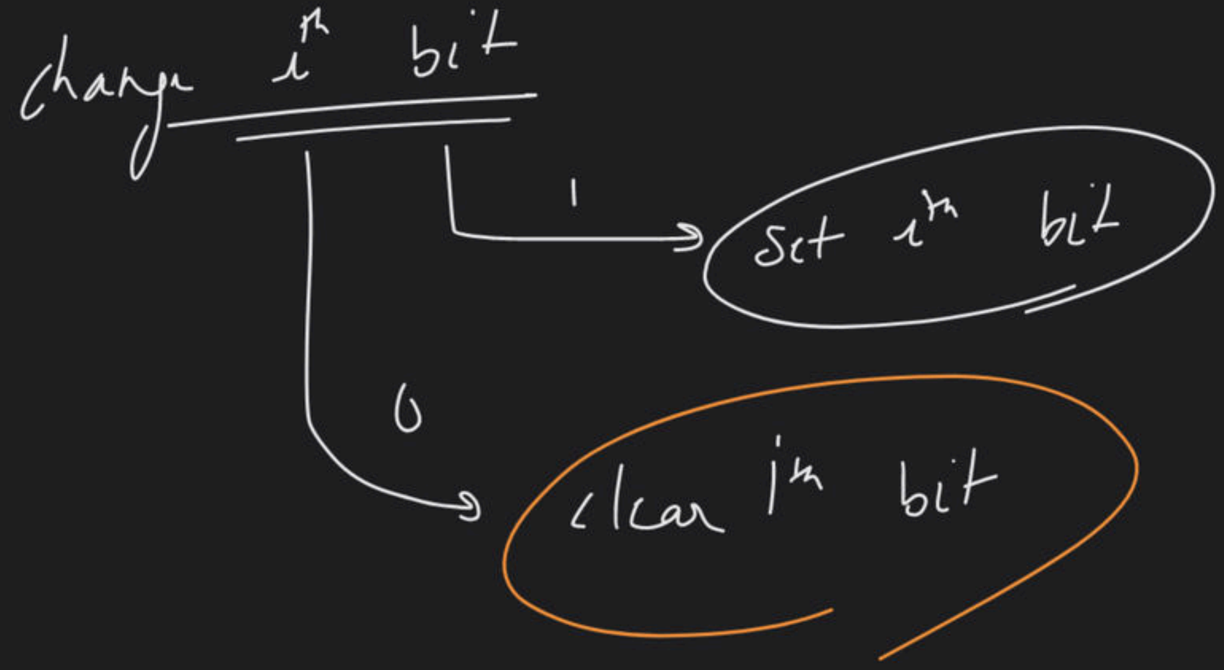

n
&1
0
even




0/0


%2 == 0 → Even
%2 == 1 → odd
A
0


Mask
get ith bit
int mask = 1<<i
int ans = n & mask
n = 10
i = 3rd
3rd 2nd 1st
0000 ---- 0000 1 0 1 0
& 0000 ---- 1000 → mask
00000 ---- 1000
LBB
1
0
& 1
mask → 1
0000 ---- 1
00000000 1000
<<3
& mask
0 → bit 0
non zero → bit 1


change i^{th} bit
1
set i^{th} bit
0
clear i^{th} bit

Set i^{th} bit

2nd bit ko set karna

n = 1010 , i = 2

2nd bit ko 1 karna

0000 ——— 0000 1 0 1 0

mask 0000 0 0 . . 0000 0 1 0 0 OR

0000 0 0 — — 0000 1 1 1 0

	0	1
0	0	1
1	1	1

↳ Create mask

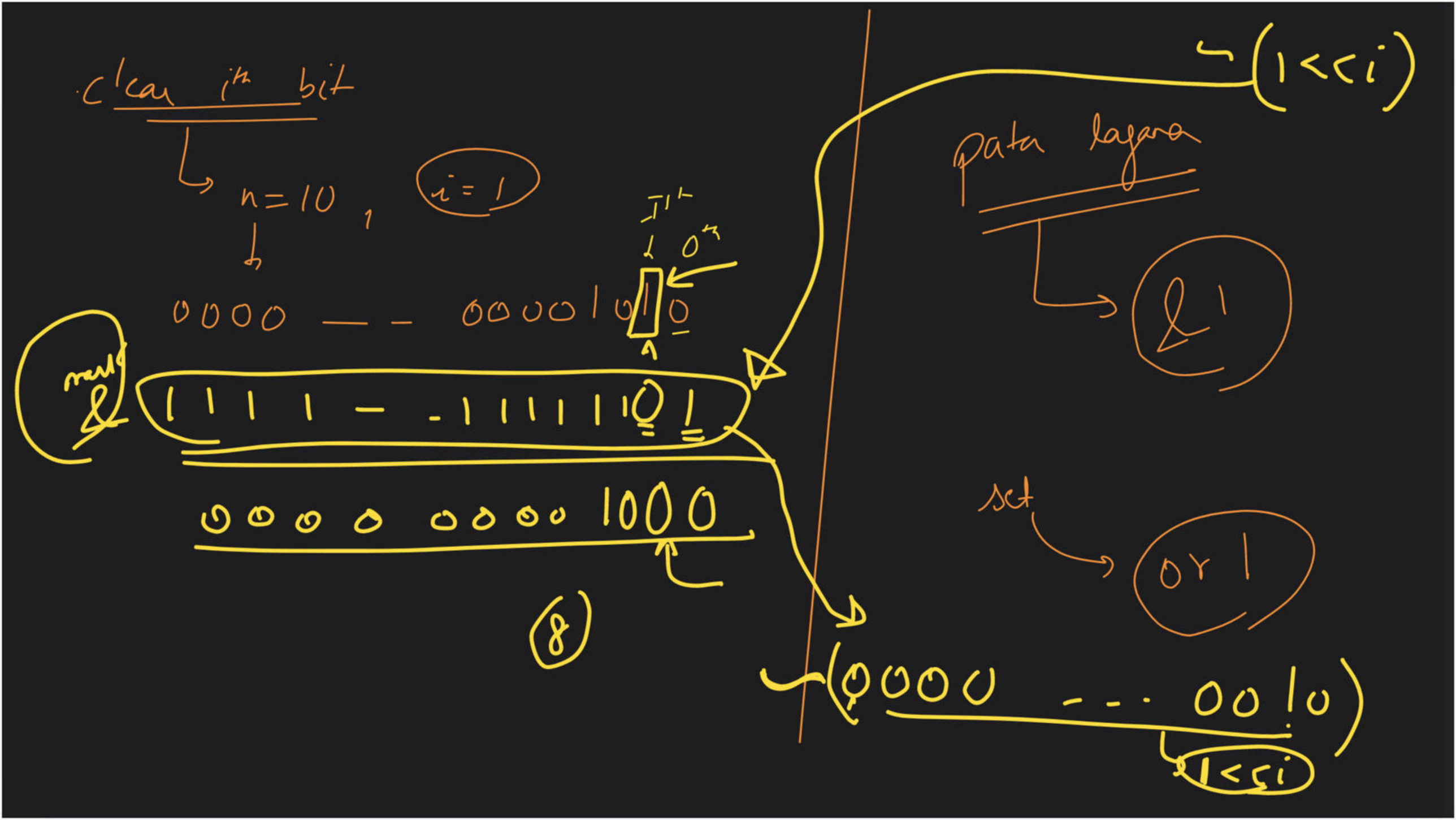
mask = 1<<2

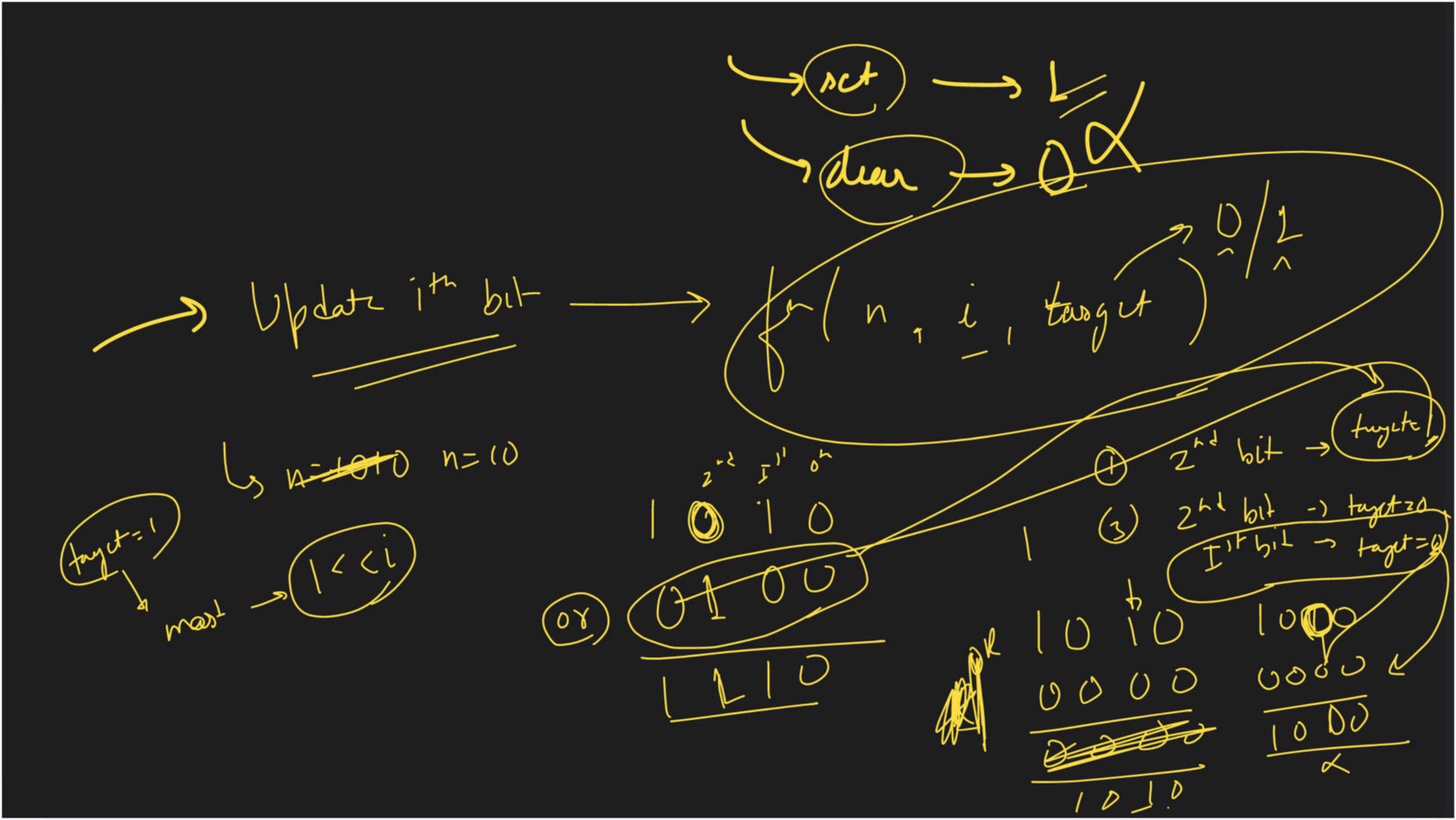
1<<i

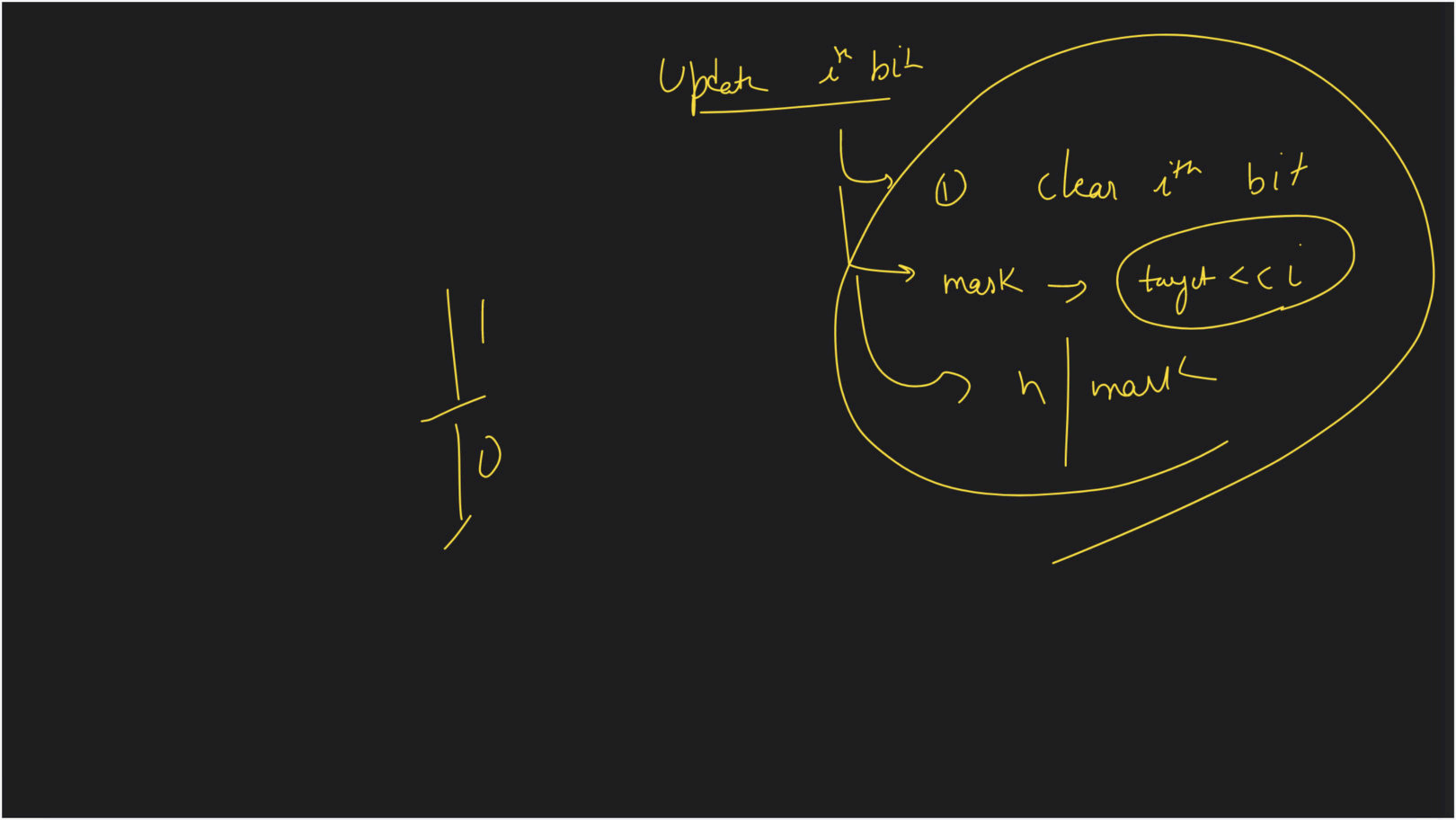
00001 <<2

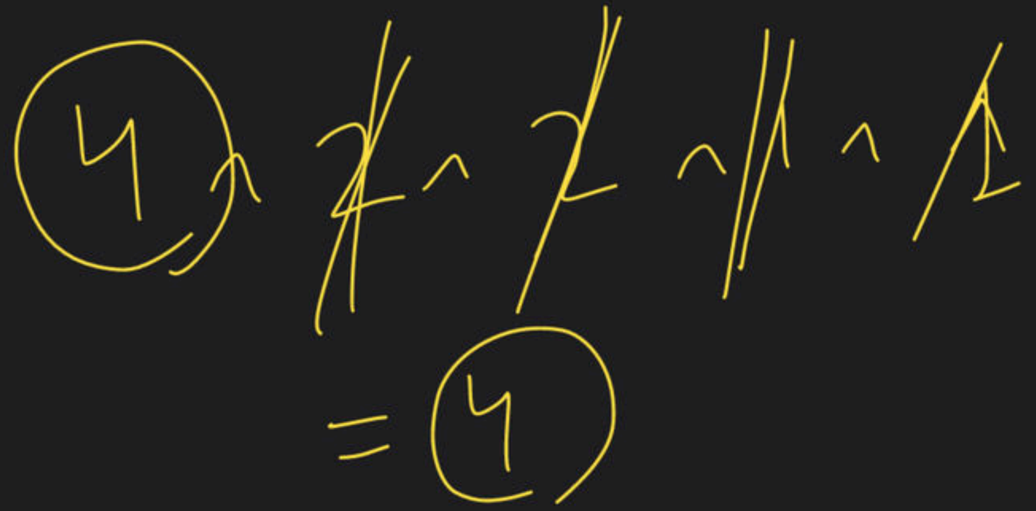
00001 00

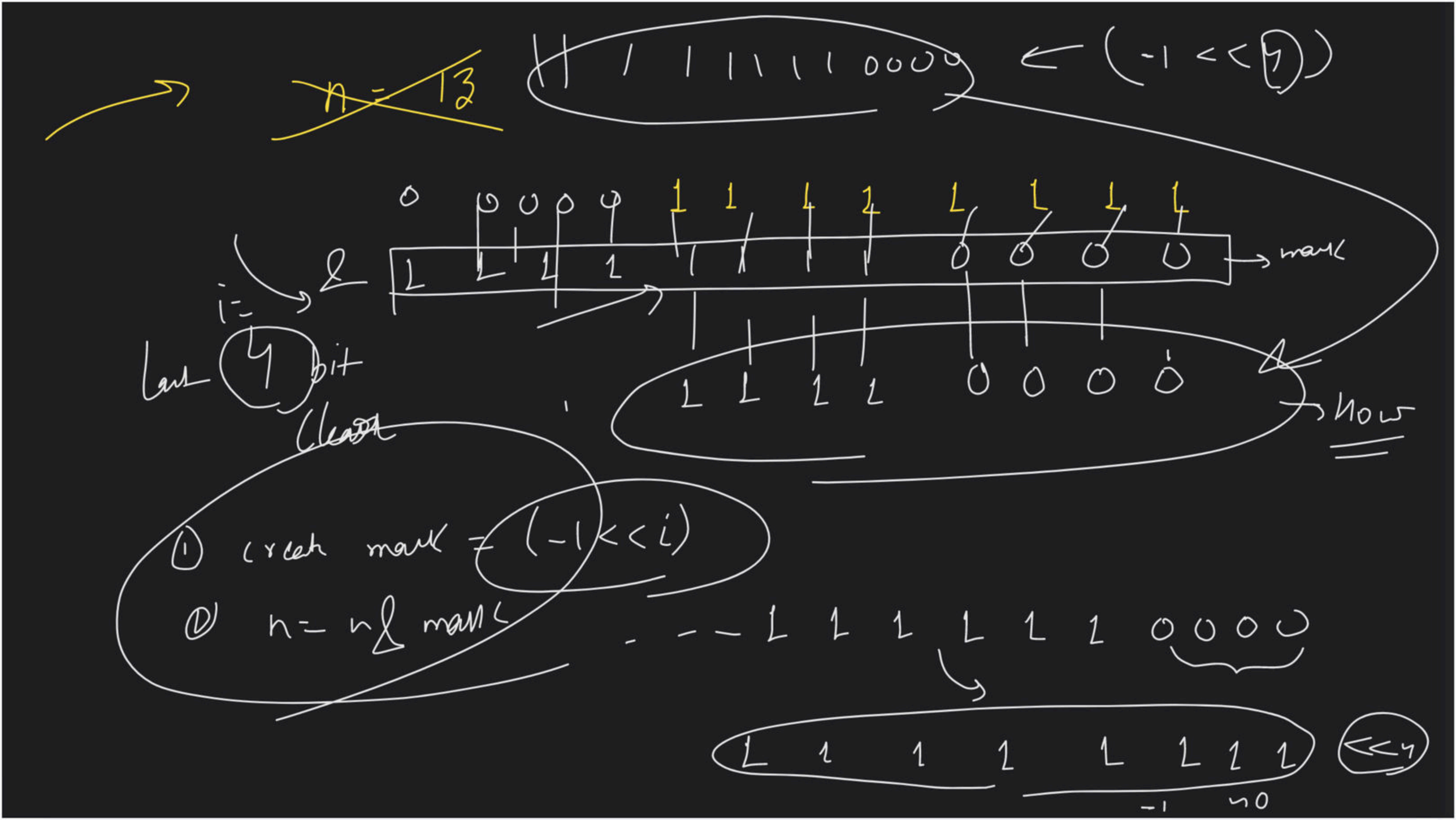
ans = n | mask

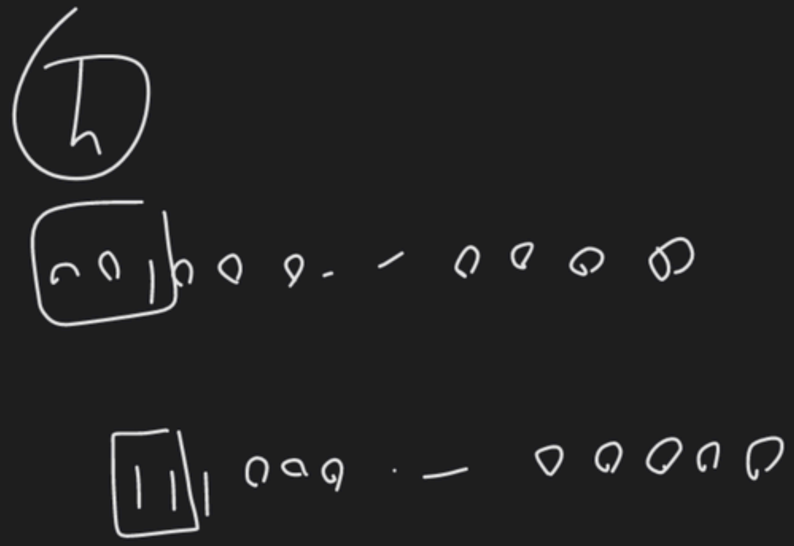

clear ith bit
n=10
i=1
0000 ... 00010
1111 ... 11101
0000 0000 1000
8
(1<<i)
Pata lagana
sct
or
(0000 ... 0010)
1<<i

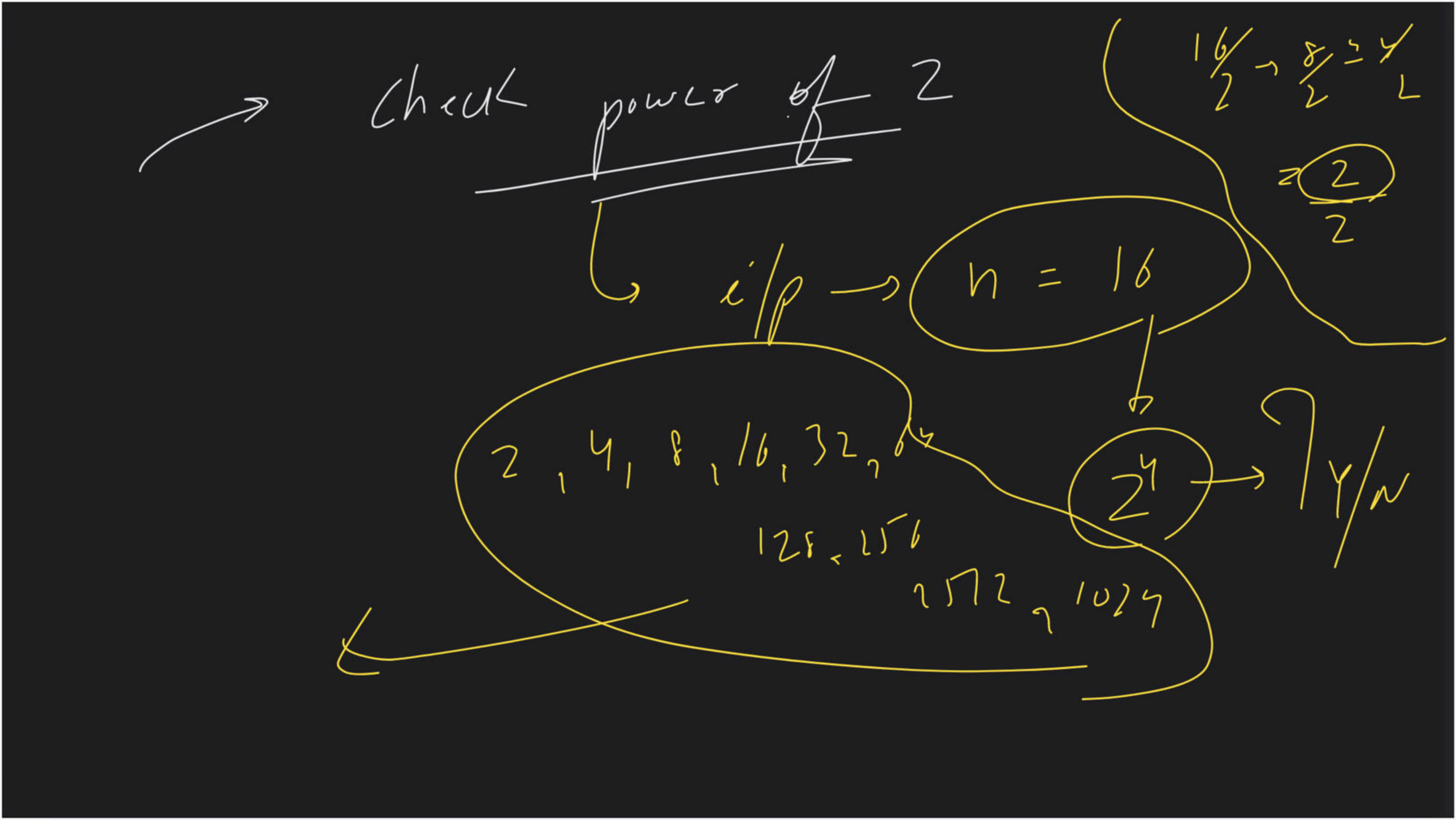

set
clear
Update ith bit
f(n, i, target)
0/1
n=10
target=1
mask
1<<i
OR
1 0 1 0
2nd bit
target
3 2nd bit -> target=0
If bit -> target=0


Update i^{th} bit
① clear i^{th} bit
mask → target << i
n | mask
1
1
0




n = 13
(-1 << 7)
mask
i =
last 4 bit
clear
1 create mask = (-1<<i)
2 n = n & mask
now
<<4
-1
-0




check power of 2
i/p
n = 16
2, 4, 8, 16, 32,
128, 256
512, 1024
2^4
Y/N
16/2 → 8/2 = 4/2
2/2

8 → 1000

7 → 0111 &

0000

$2^1 \rightarrow 2 \rightarrow 10$

$2^2 \rightarrow 4 \rightarrow 100$

$2^3 \rightarrow 8 \rightarrow 1000$

$2^4 \rightarrow 16 \rightarrow 10000$

16 → 10000

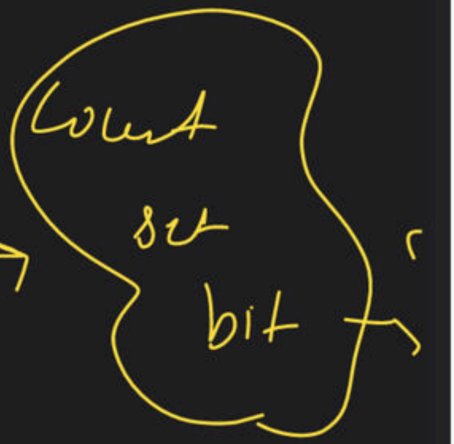
15 → 01111 &

0000

n → 4 → 100

n-1 → 3 → 011 &

000



n & (n-1) == 0

if it is a power of 2

Count Set bits

8 → 0000 ... 1000

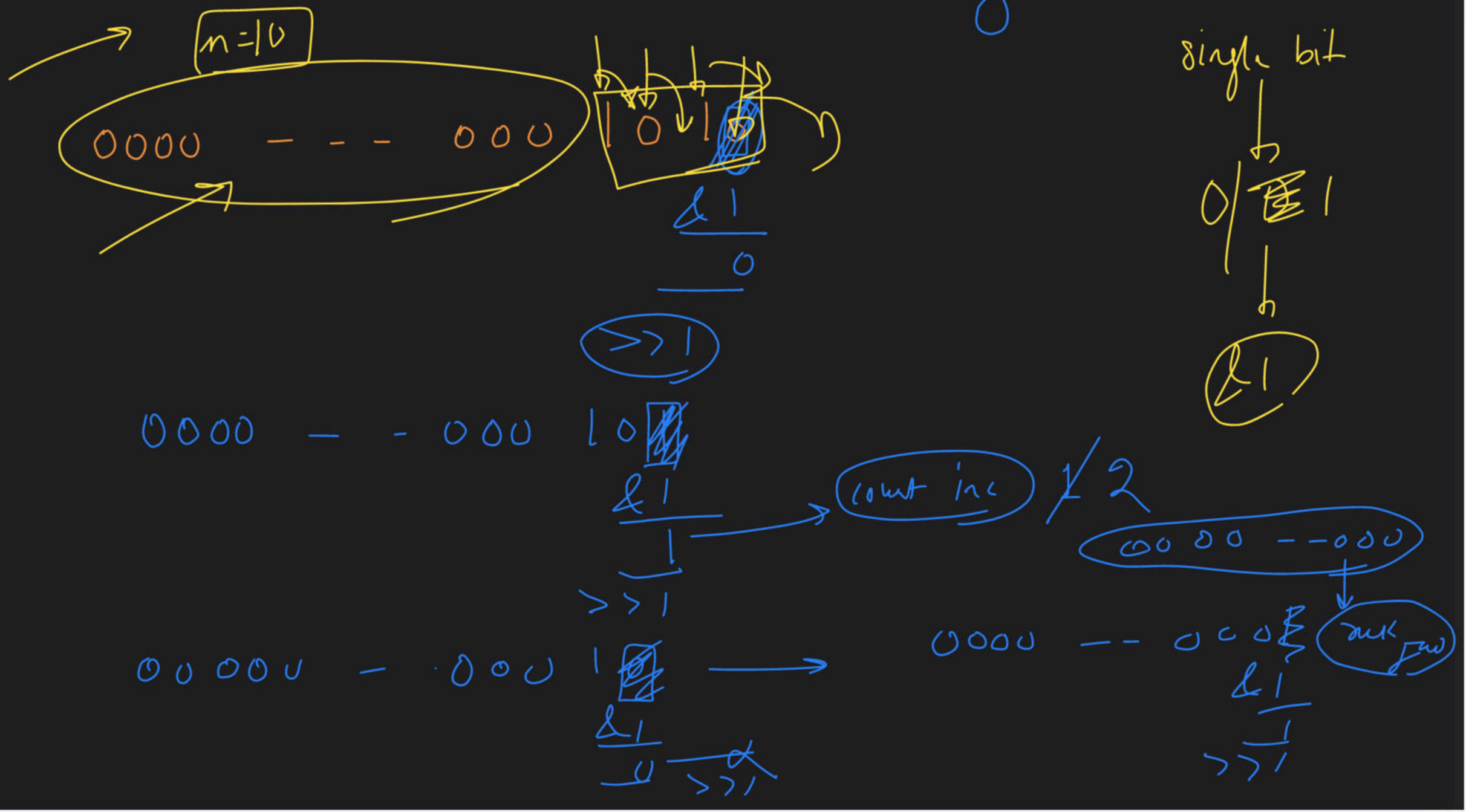
no of set bit = 1

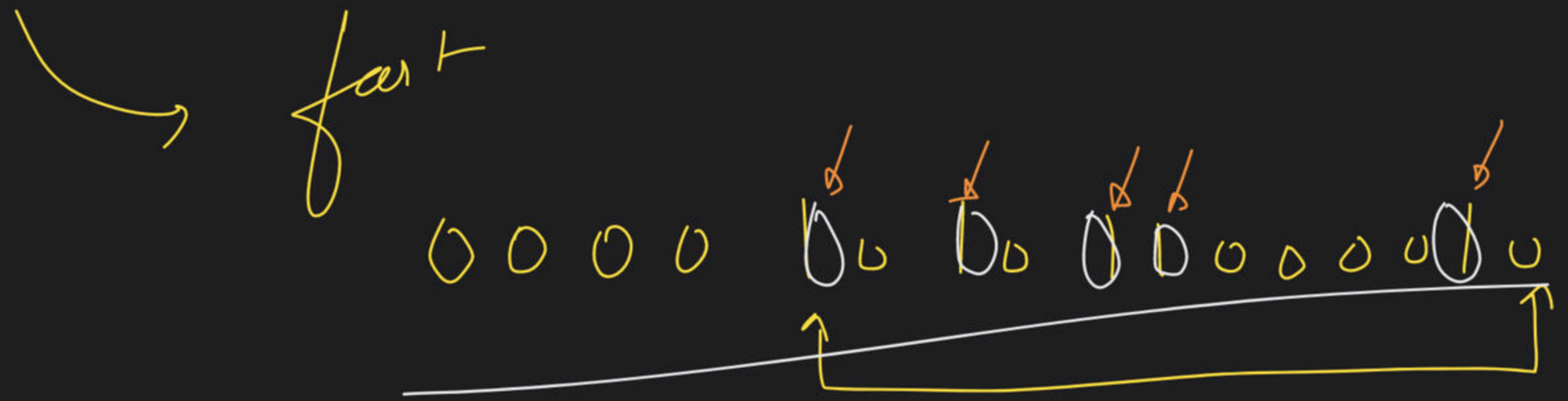
7 → 0000 ... 0111

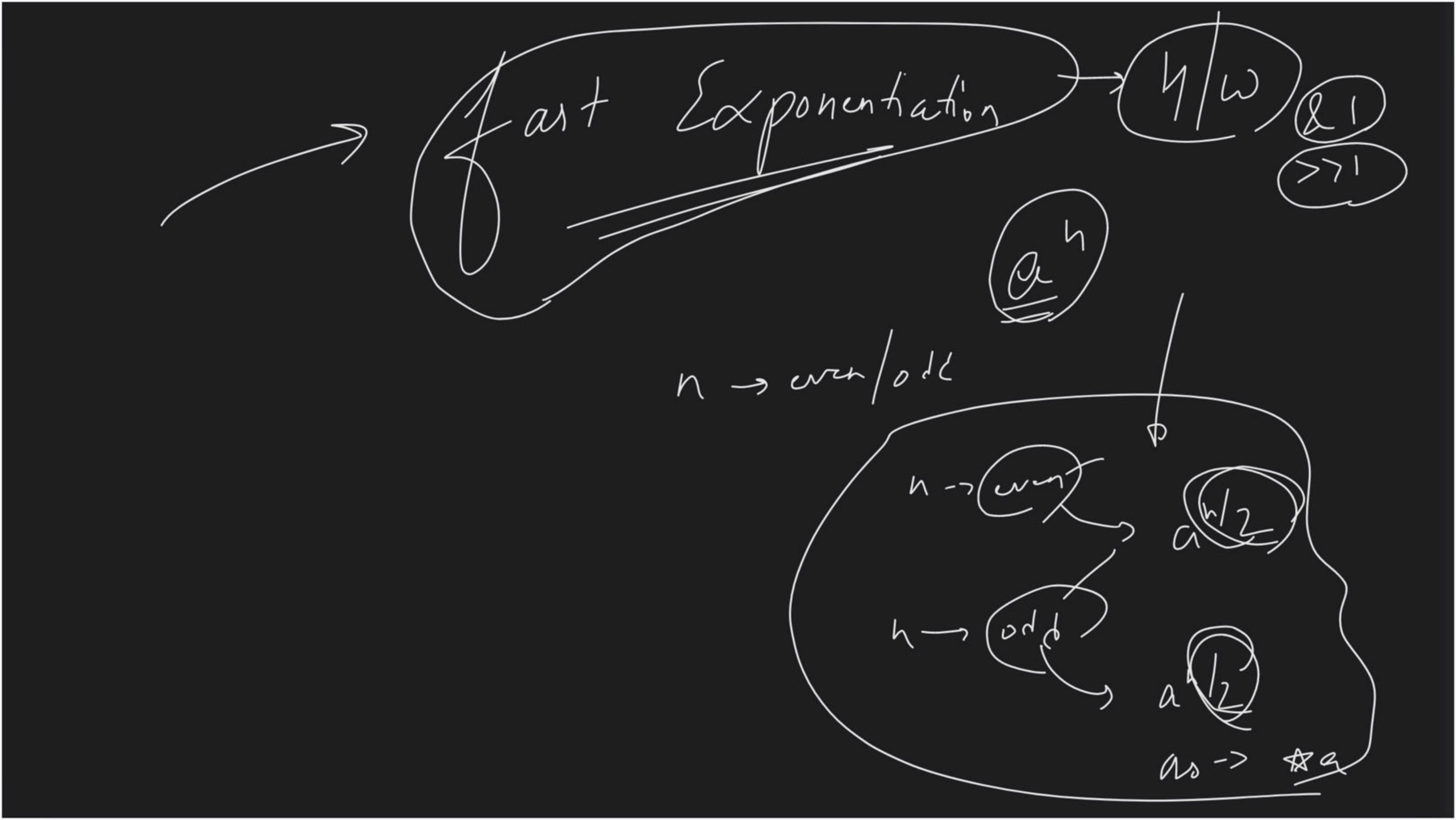
no of set bit = 3

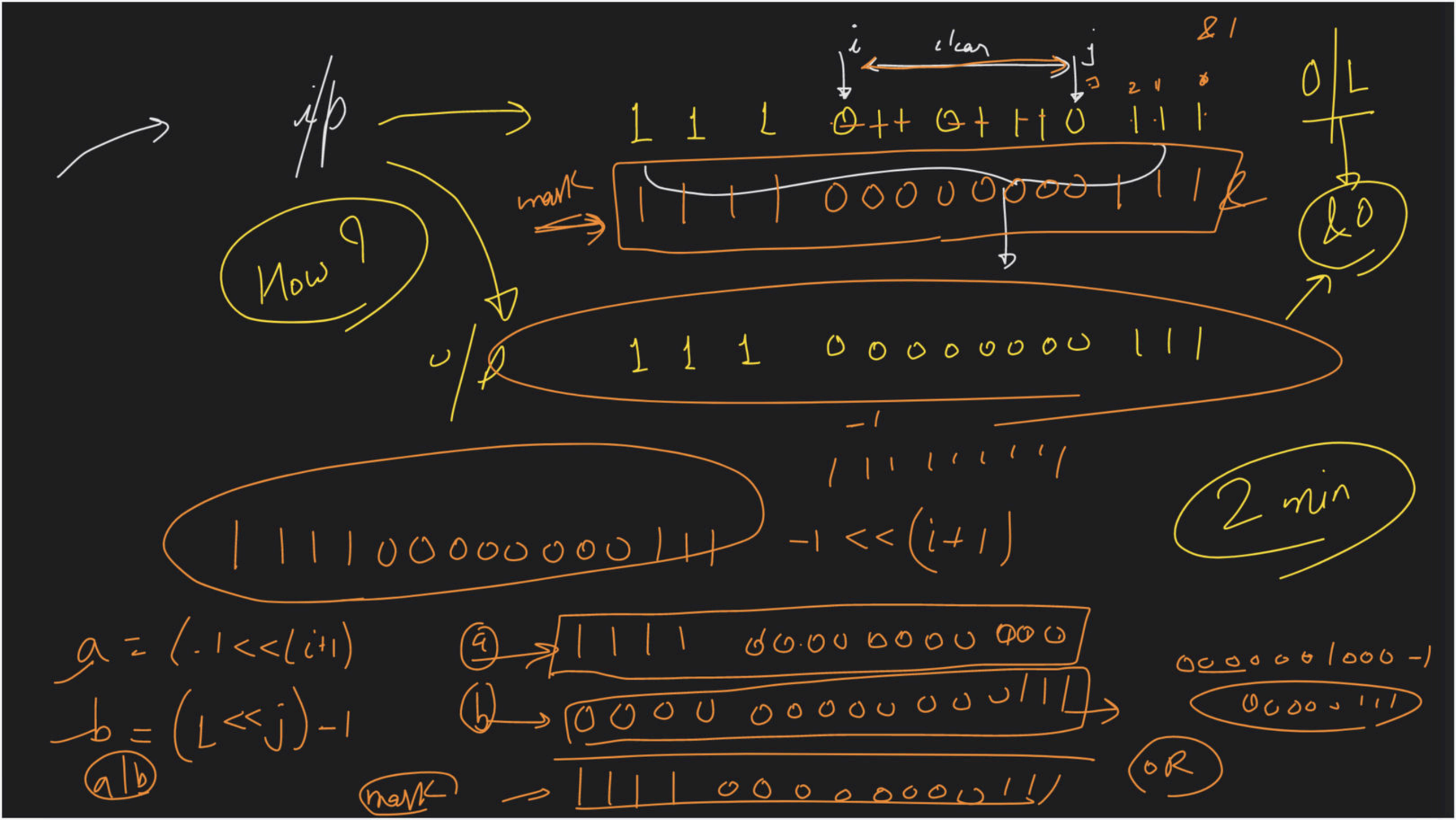
10 → 0000 ... 1010

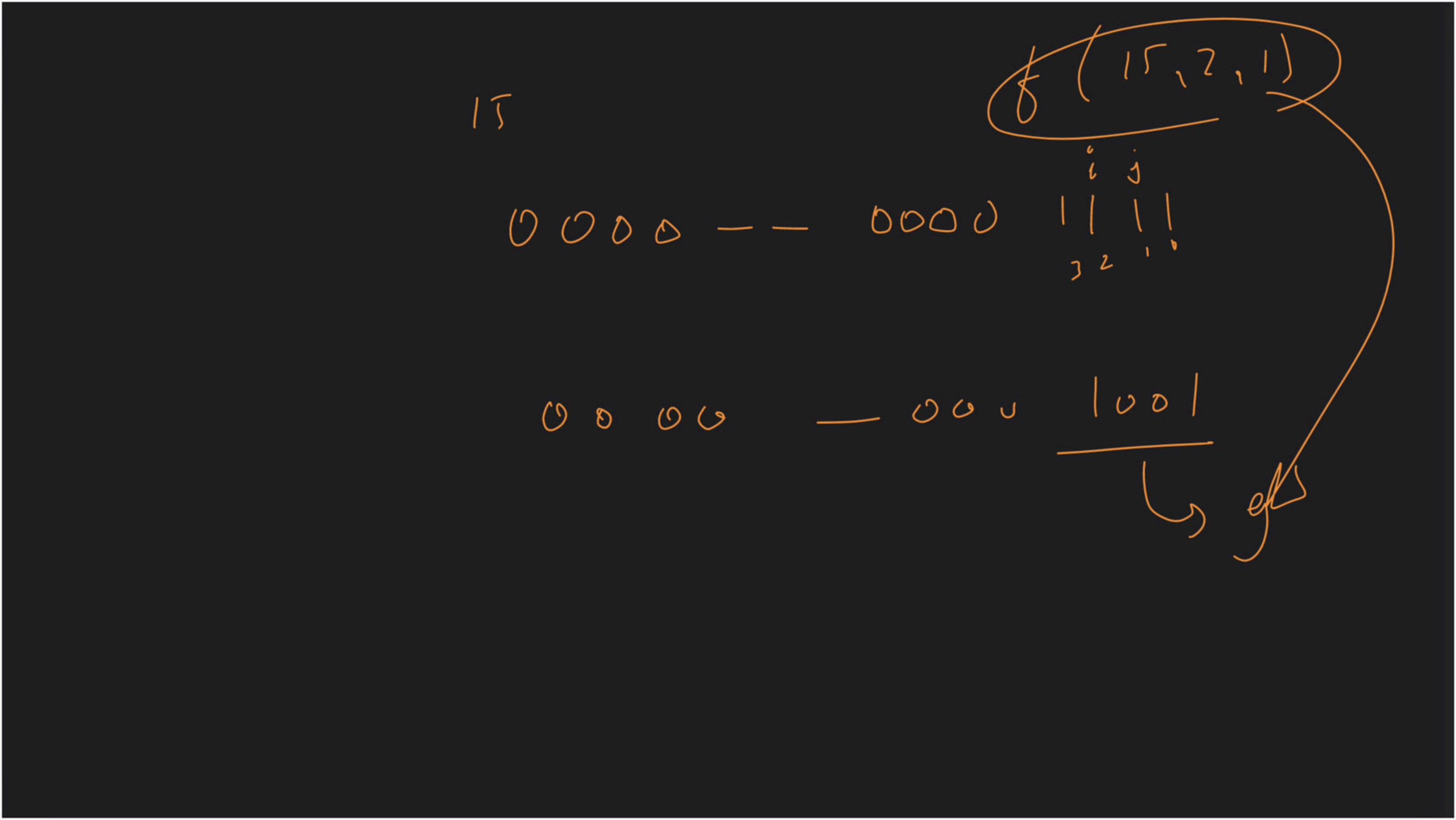
no of set bit = 2

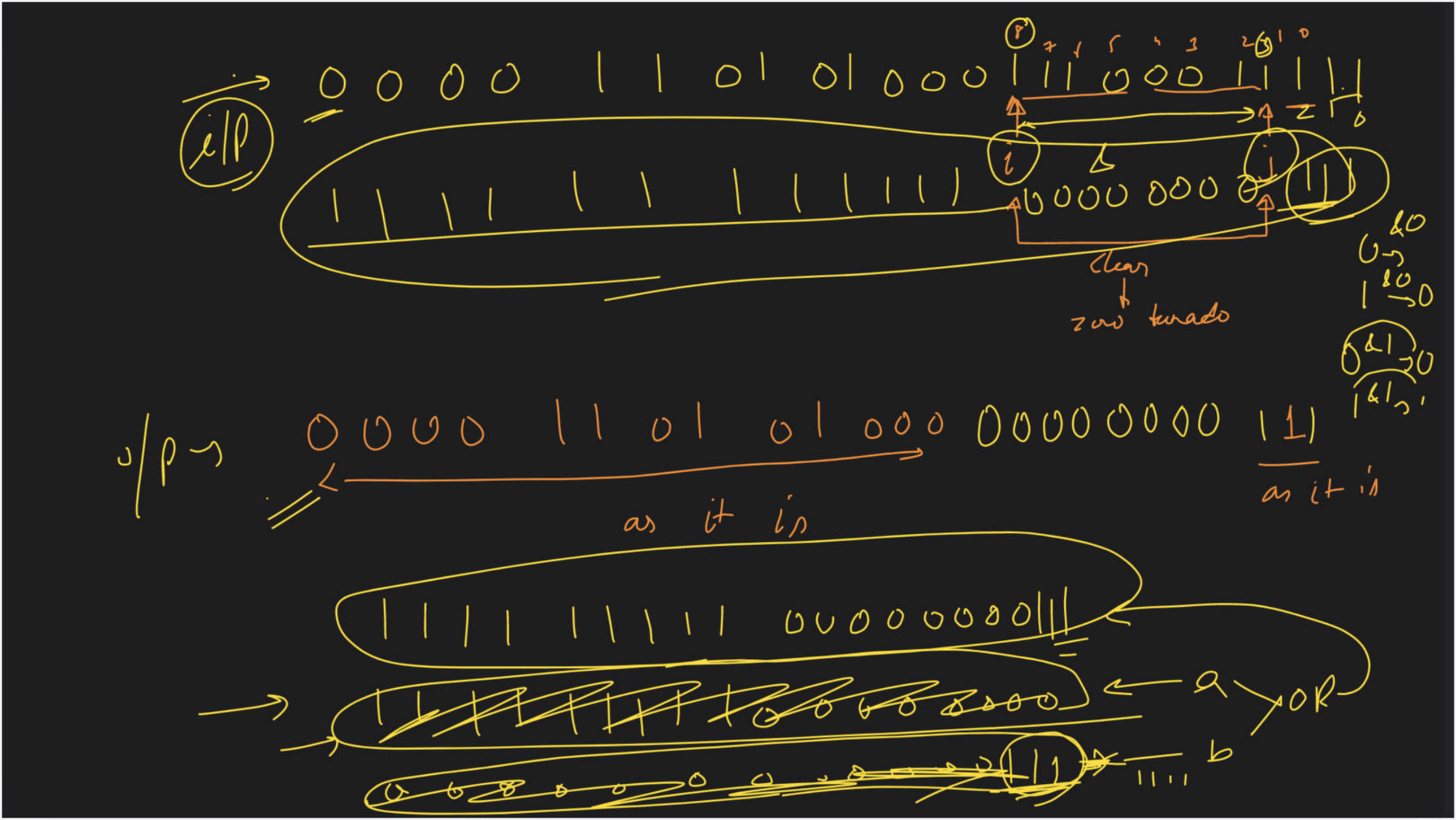

n=10
0000 - - - 000
101
&1
0
>>1
0000 - - 000 10
&1
1
count inc
/2
>>1
00000 - 000 1
&1
0
>>1
single bit
0/1
&1
0000 - - 000
mark
&1
1
>>1

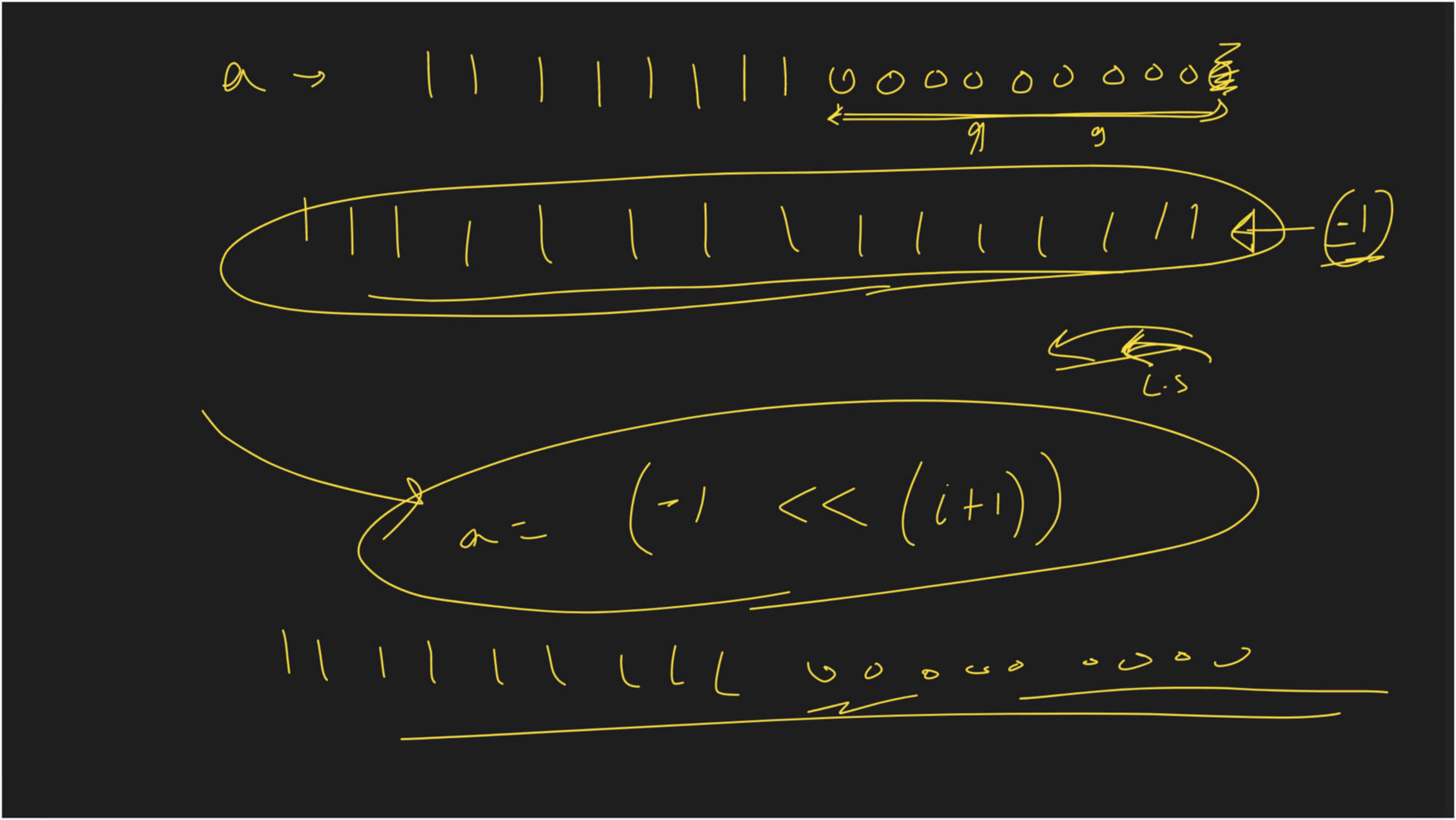

fast

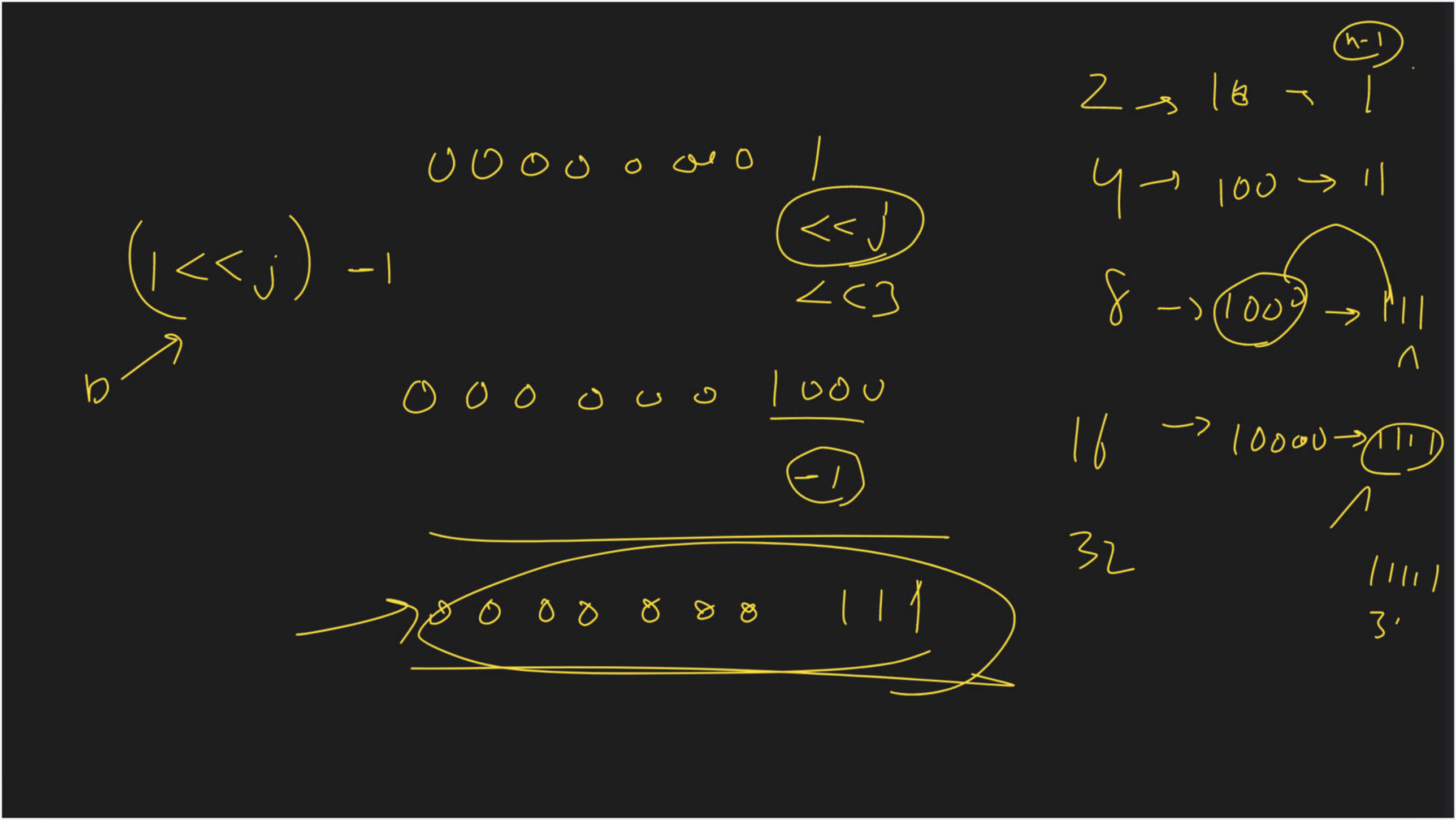

Fast Exponentiation
n/w
&1
>>1
a^n
n → even/odd
n -> even
a^n/2
n → odd
a^n/2
a^n -> a*a


i
i'can
j
&1
i/p
1 1 1 0 0 0 0 0 1 1 1
mark
0/L
&0
Now 9
0/0
1 1 1 0 0 0 0 0 0 0 0 1 1 1
-1
1 1 1 1 0 0 0 0 0 0 0 0 1 1 1
-1 << (i+1)
2 min
a = (-1<<(i+1)
b = (1<<j)-1
a|b
a
b
mark
OR

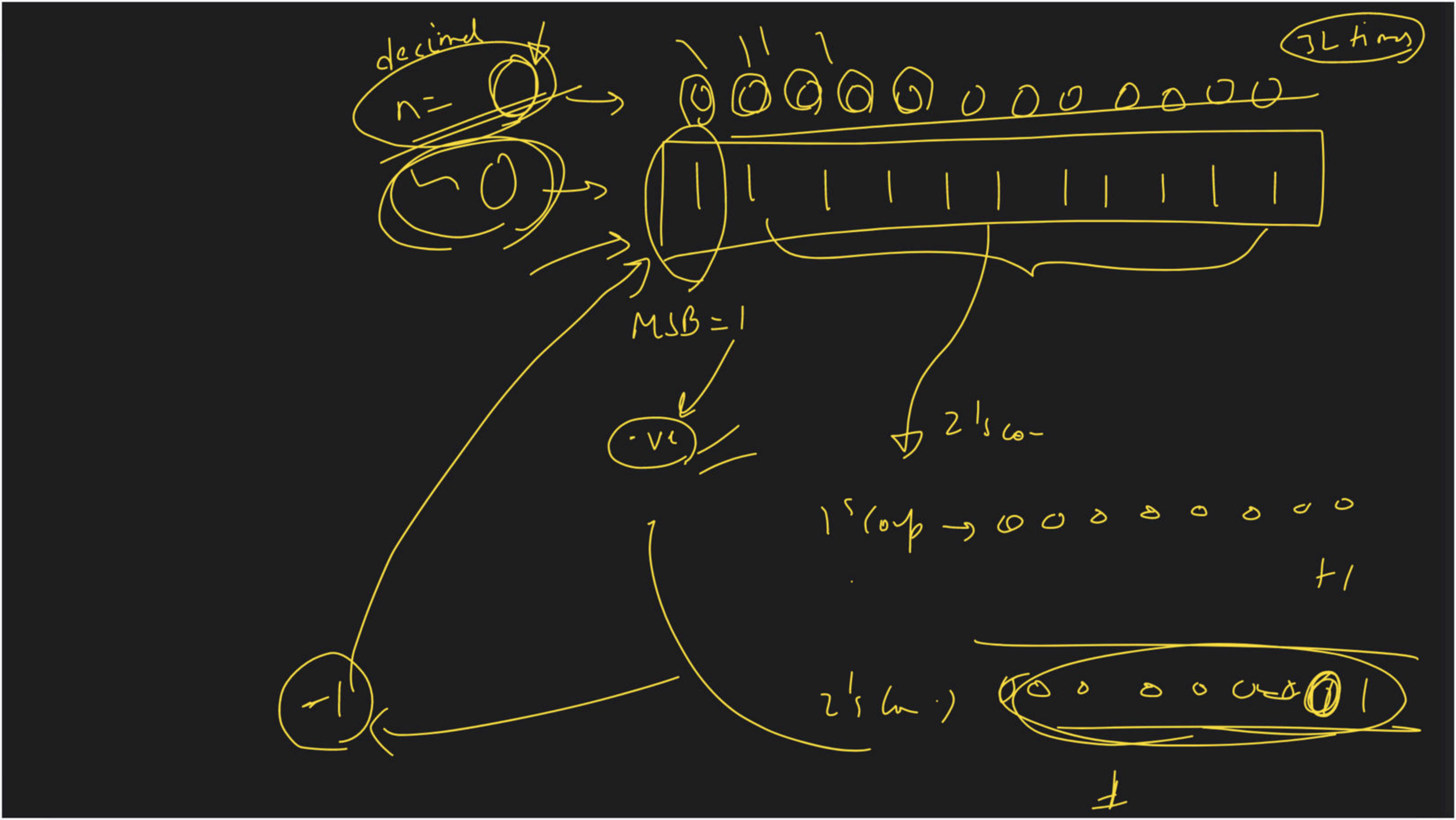


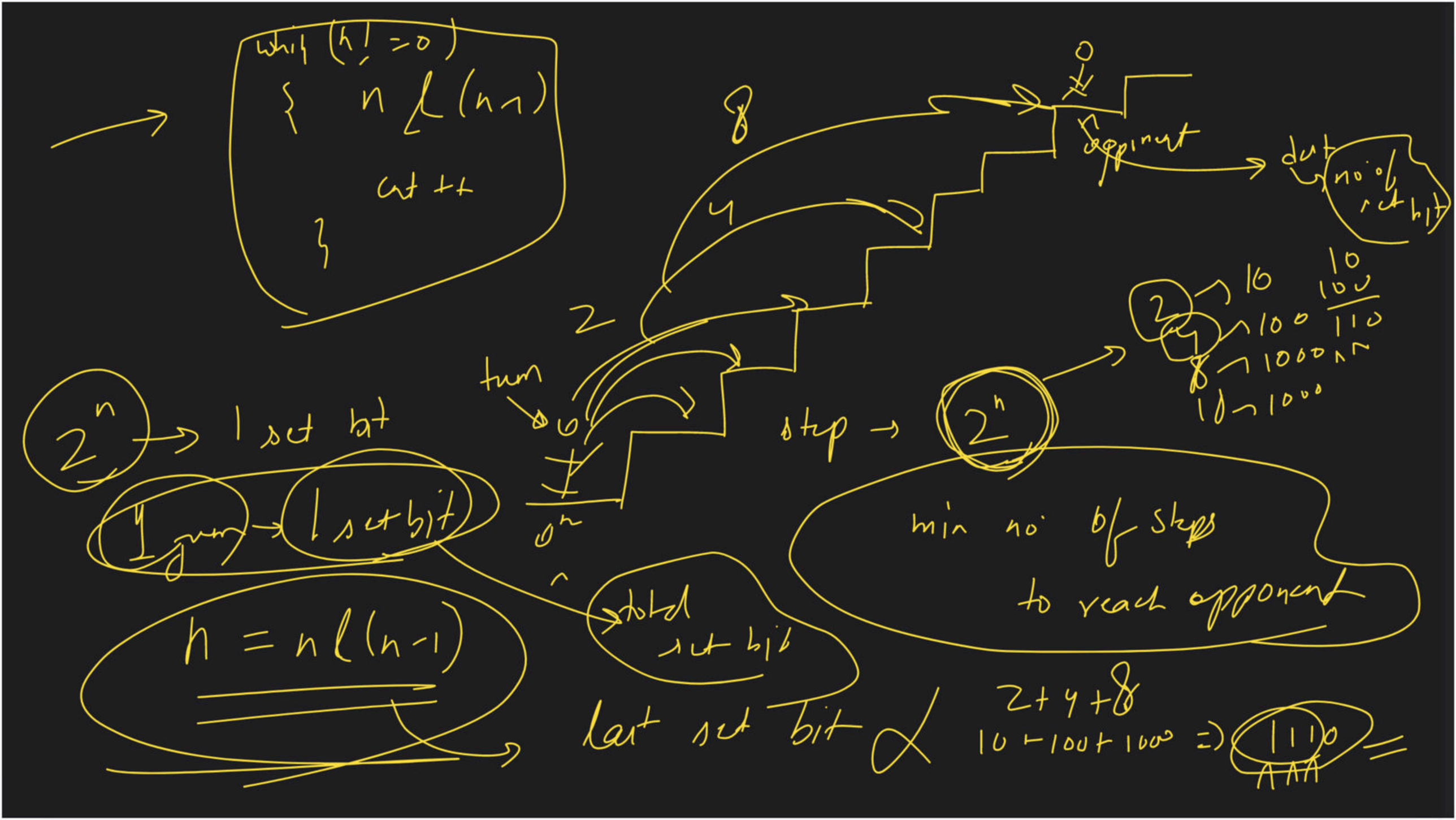

i/p
clear
zero terminated
o/p
as it is
as it is
OR


a →
-1
L.S
a = (-1 << (i+1))


(1<<j) -1
b
1<<j
<3
1000
-1
111
2 → 16 → 1
n-1
4 → 100 → 11
8 → 1000 → 111
16 → 10000 → 1111
32
11111
31


a/b
mark


decimal
n=
-0
MSB=1
-ve
2's comp
1's comp → 0 0 0 0 0 0 0 0
+1
2's comp
0 0 0 0 0 0 0 1
-1

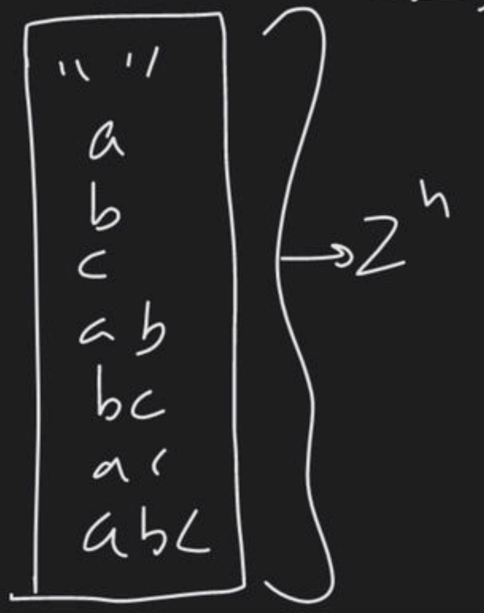

while (n != 0)
{ n = n & (n-1)
cnt ++
}
8
4
2
1
turn
0
0^n
step →
2^n
min no of steps
to reach opponent
2^n → 1 set bit
1 turn → 1 set bit
n = n & (n-1)
total set bit
last set bit
2+4+8
10+100+1000 => 1110
Sapient
dut
no. of set bit
10
100
1000
10 → 10
100 → 110
1000 → 1110

→ Subsequence Using Bitmasking

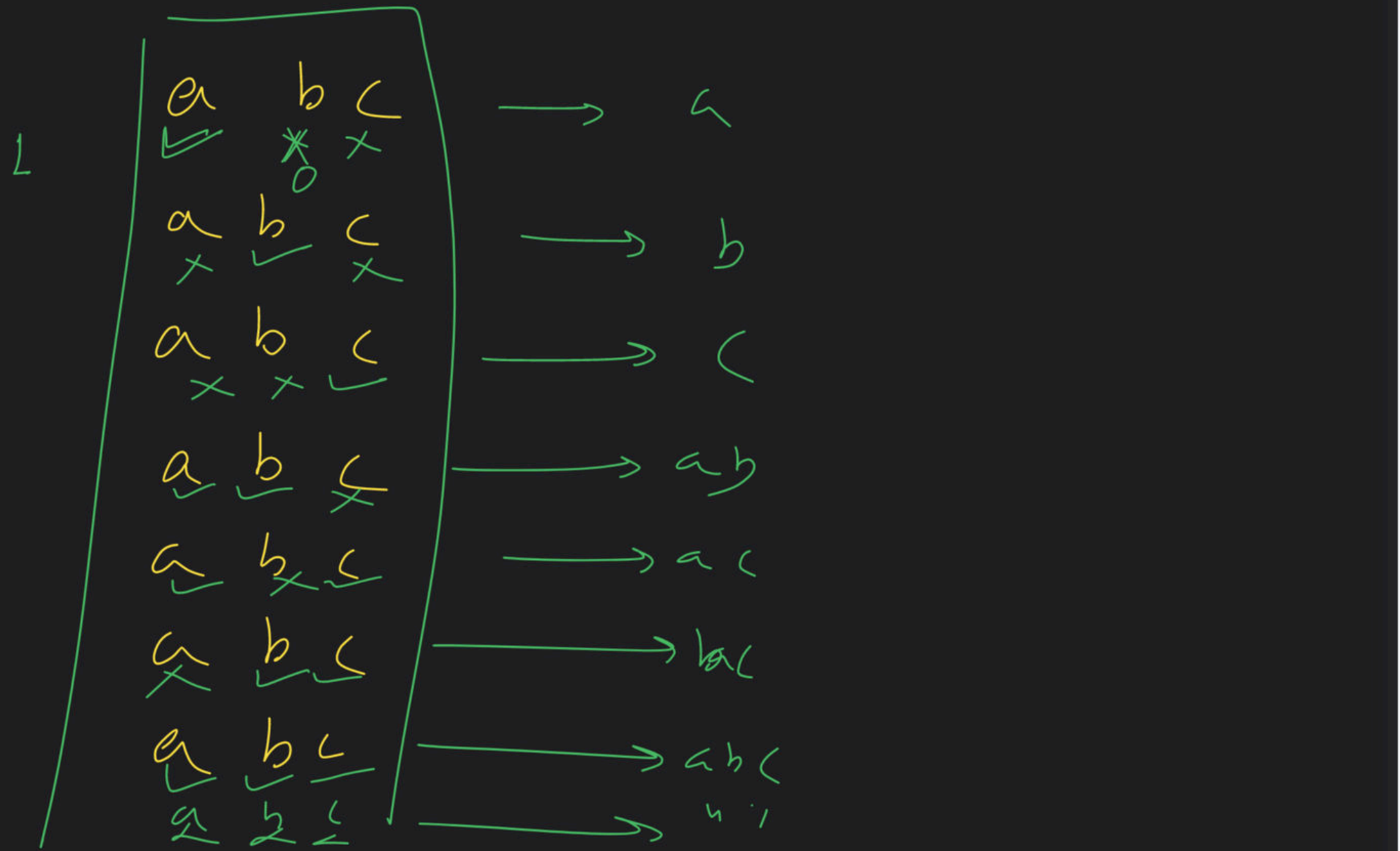
n → length of string

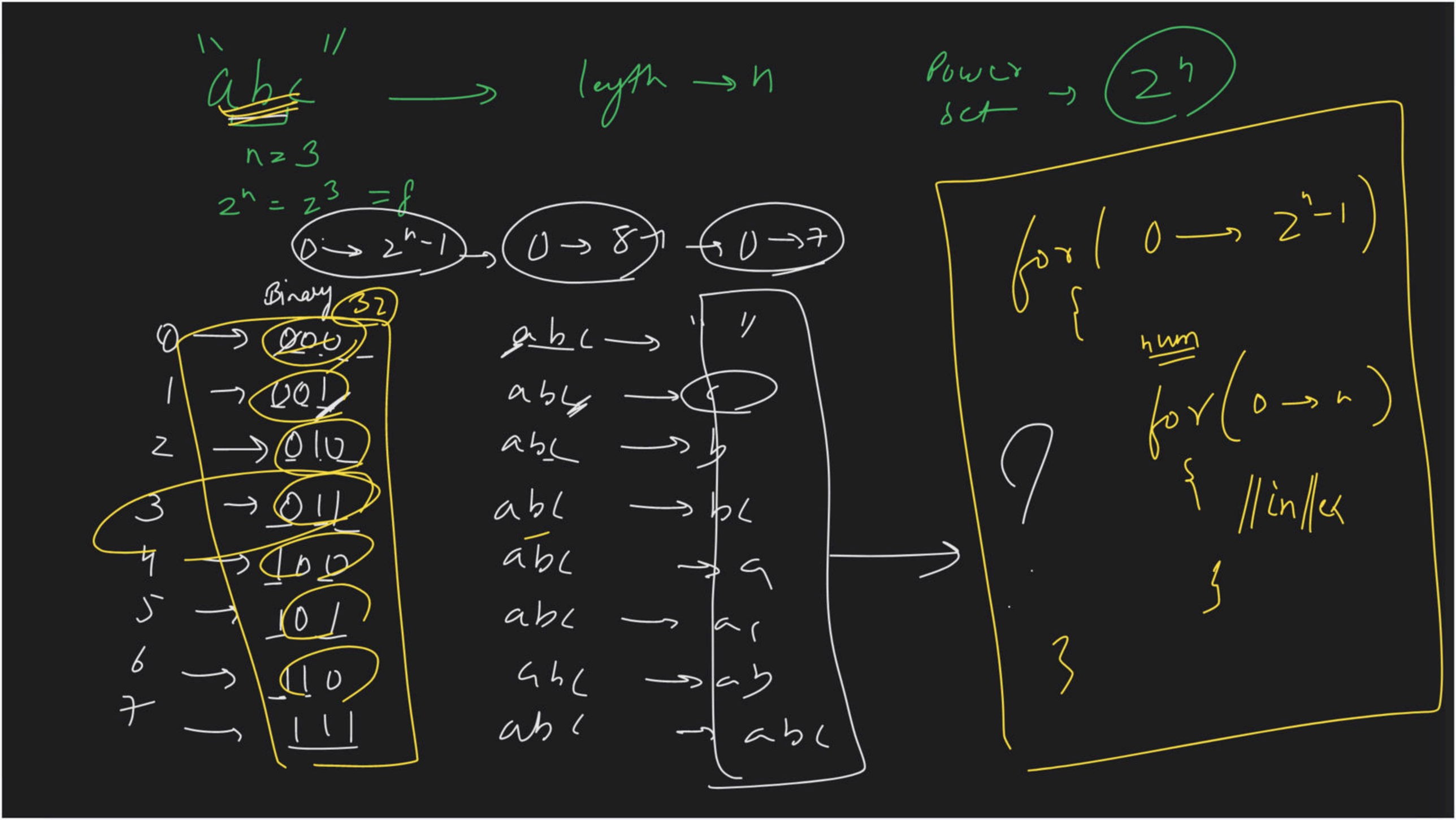
str = "abc"

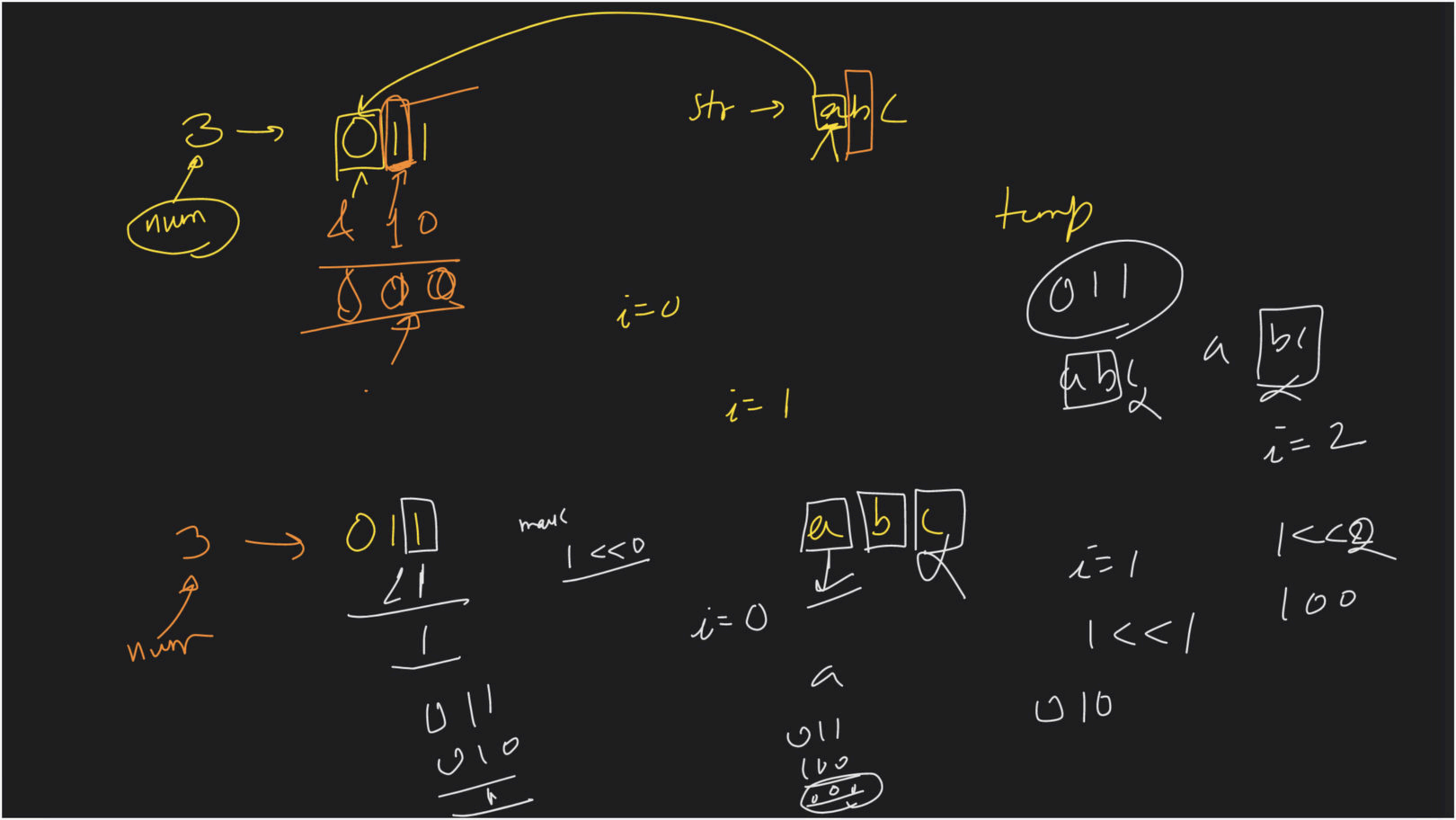
Power set or Subsequence →



already solved → Recursion → incl/excl pattern


1
a b c
a
a b c
b
a b c
c
a b c
ab
a b c
ac
a b c
bc
a b c
abc


"abc"
length → n
n = 3
Power set → 2^n
$2^n = 2^3 = 8$
0 → $2^n - 1$
0 → 8
0 → 7
Binary
32
0 → 000
1 → 001
2 → 010
3 → 011
4 → 100
5 → 101
6 → 110
7 → 111
abc → ""
abc → c
abc → b
abc → bc
abc → a
abc → ac
abc → ab
abc → abc
for(0 → $2^n - 1$)
{
num
for(0 → n)
{
//inside
}
?
}


3
num
str
a b c
i=0
i=1
temp
011
a b c
b
i=2
3
num
011
1
011
010
main
1<<0
a b c
i=0
a
011
100
001
i=1
1<<1
010
1<<2
100