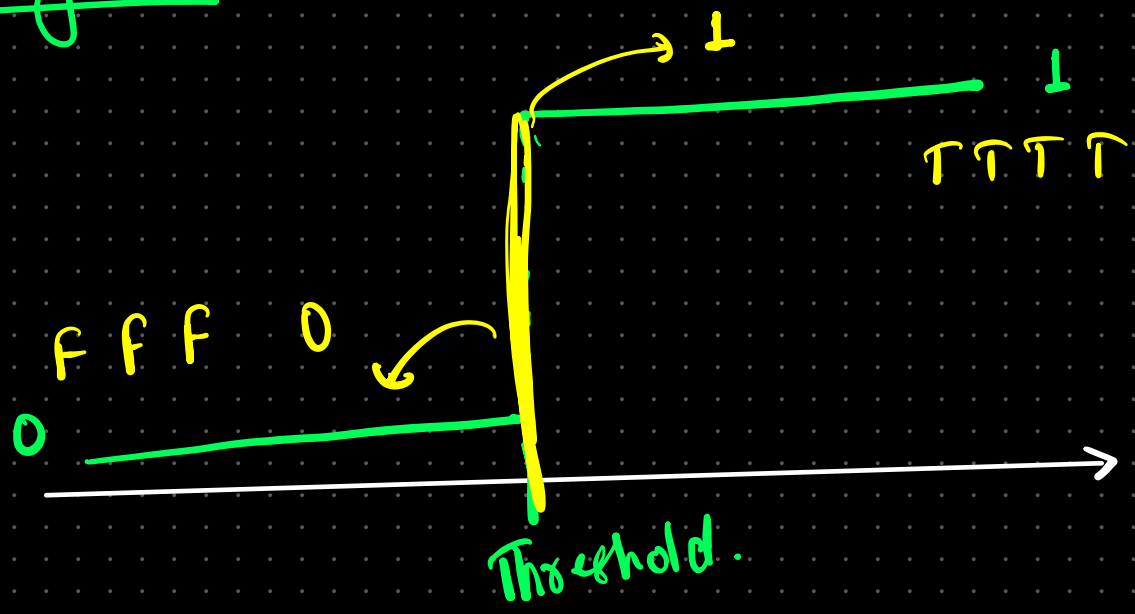
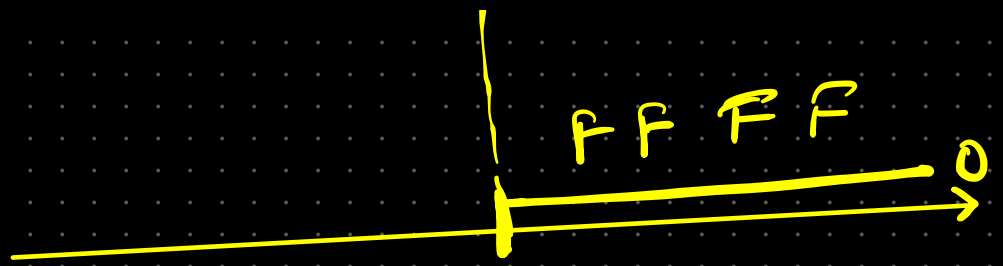
Day 2: Binary Search:

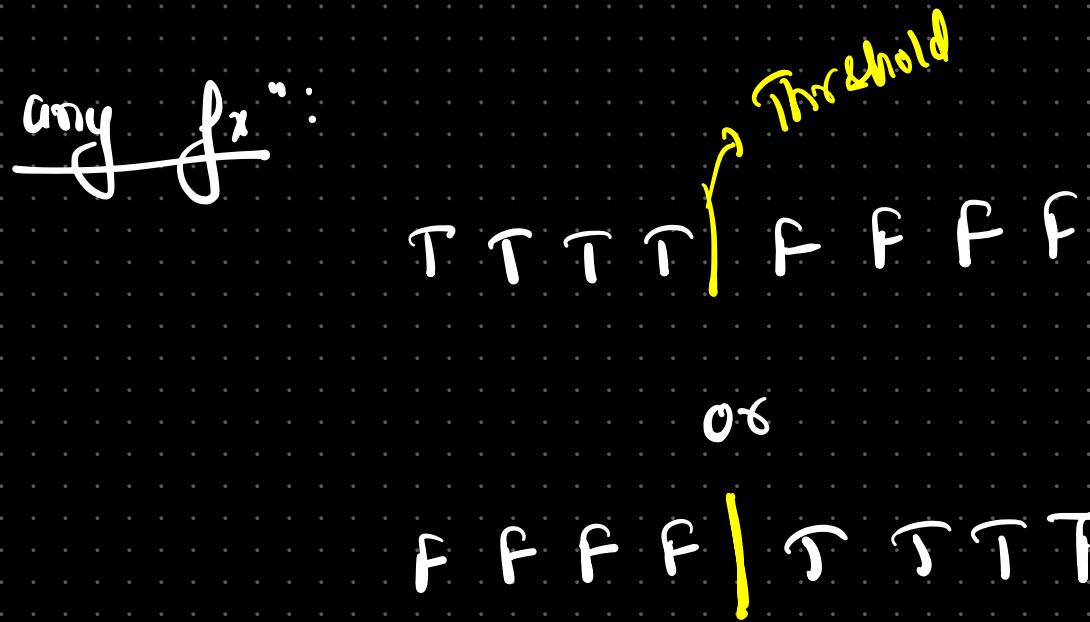
Predicate fx":



1. T T T T



any f_x":



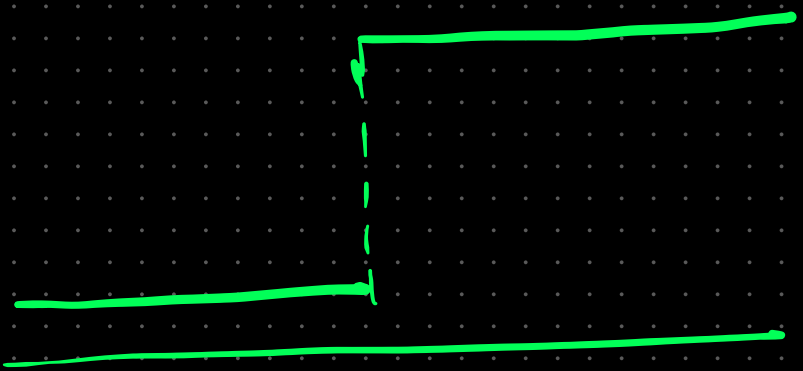
1 2 3 5 7 9 11

target = 3

start = 0

end = 6.

mid = 6/2 = 3.



```
if (arr[mid] >= target) {
    ans = mid.
    end = mid - 1
}
```

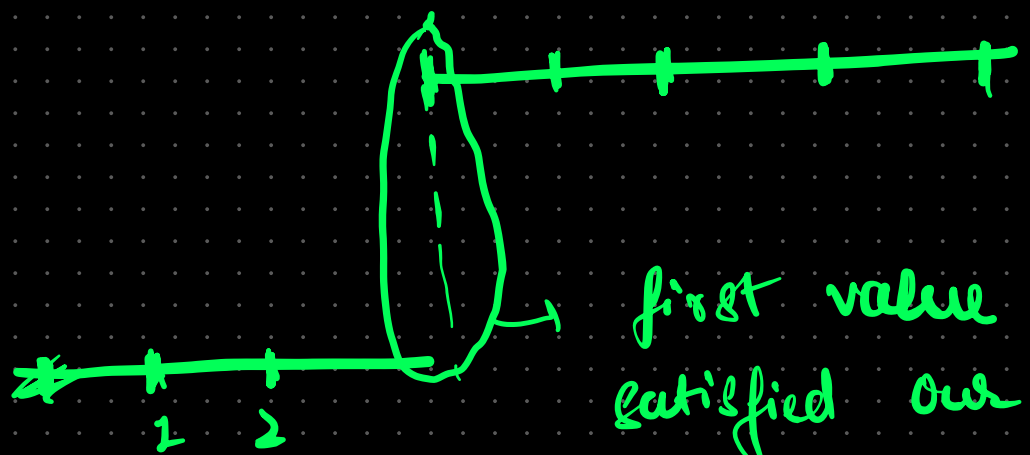
}

else start = mid + 1.

What is predicate here ?

is the element in array

~~arr~~ indexed at mid >= target !



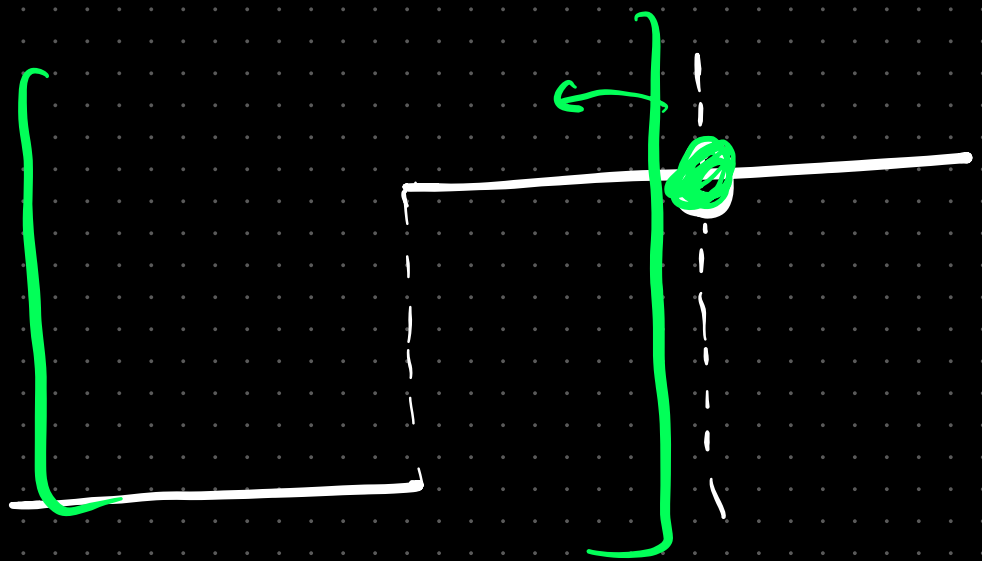
first value which satisfied our predicate

:/ our

1 2 (3) 5 7 9 11 ... answer.

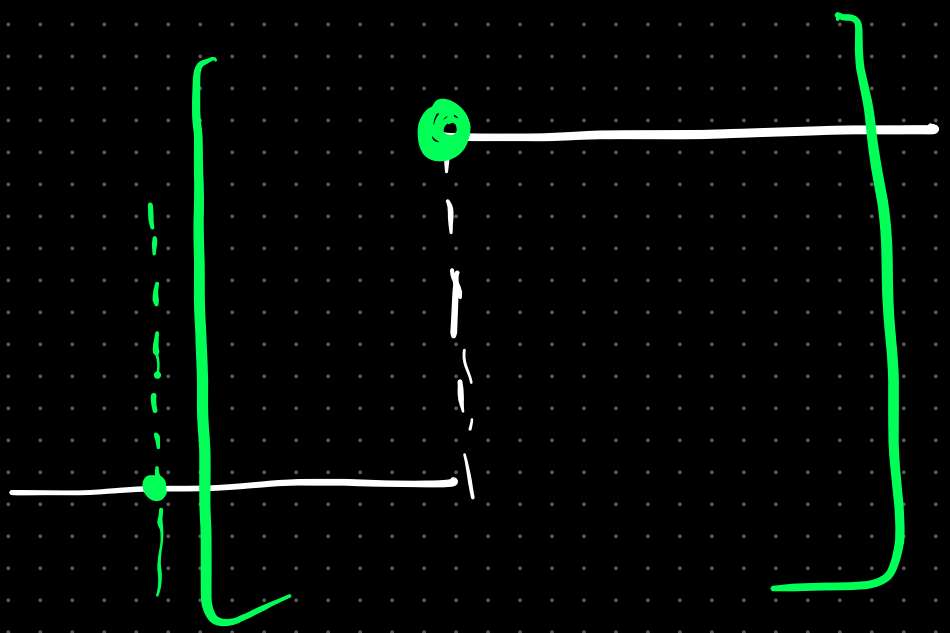
boolean: True that our $f(x)$ is correct.

(assumption about mid is correct)



if correct.

end = mid - 1



Start = mid + 1.

Start = 0

end = 1e6 = 10^6.

while (start <= end) {

mid = (start + end) / 2.

if (predicate(mid) == True) {

ans = mid.

start = mid + 1 / end = mid - 1

}.

else {

start = mid + 1 or end = mid - 1;

}

$n^2 \rightarrow n \log n$.

Question:

odd sized array (unsorted).

integer - K.

you can apply k operation.

arr $[a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5]$.

one op: $[a_1 \quad a_2+1 \quad a_3 \quad a_4 \quad a_5]$.

Aim: We have to maximize the median of this array.

$n = 5$

$k = 5$

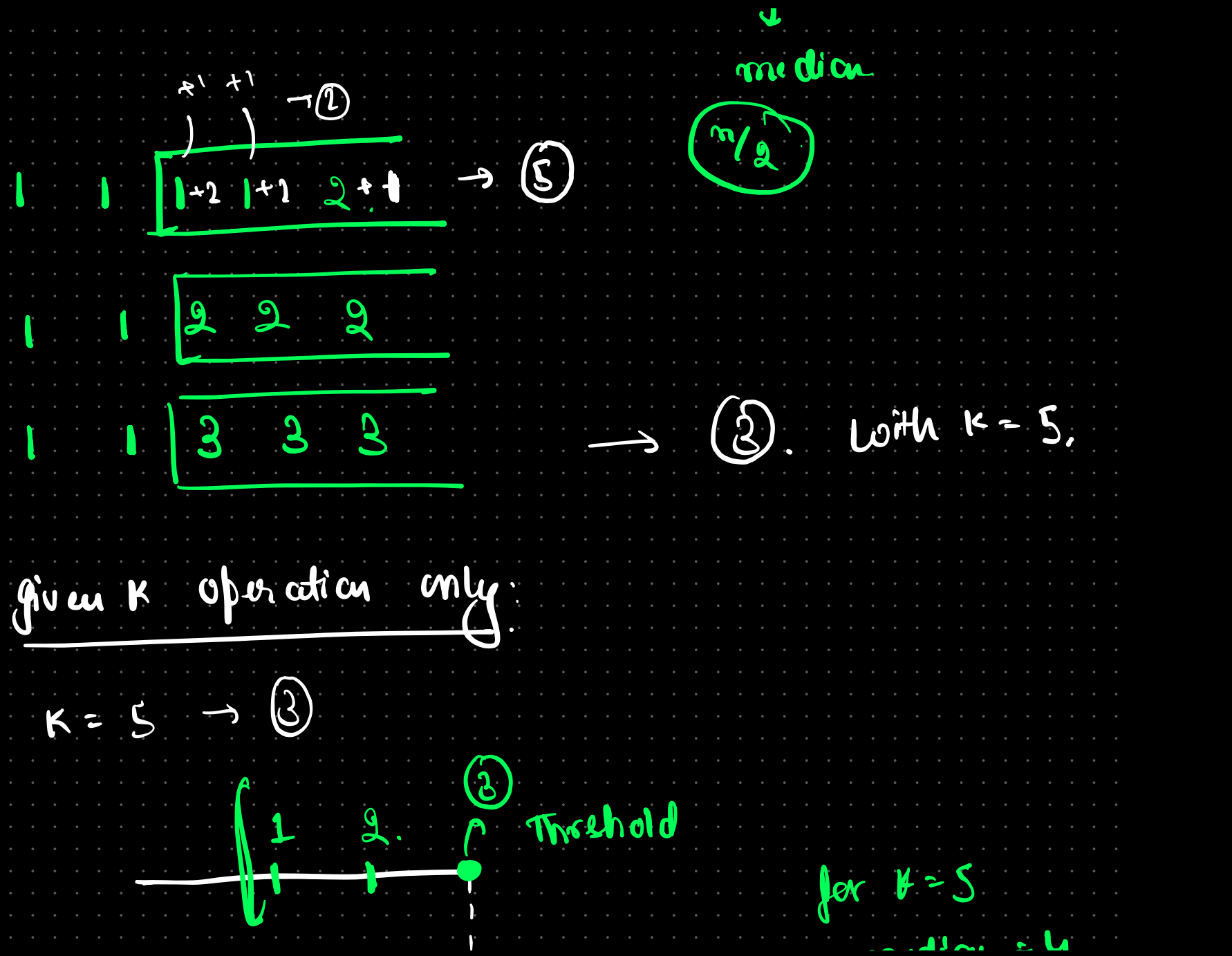
1 2 1 1 1

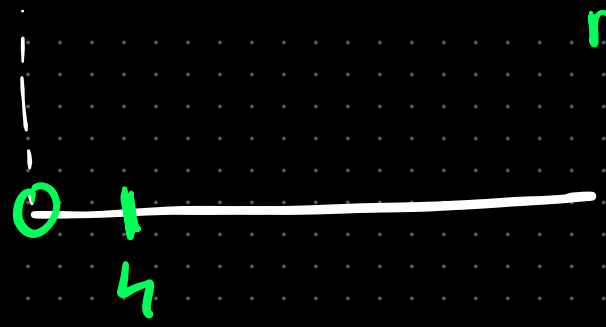
median :

0 1 2 3 4

1 1 1 1 2

1


median
m/2
+1 +1 → ②
1 1 1+2 1+1 2+1 → ⑤
1 1 2 2 2
1 1 3 3 3 → ③. with K = 5.
given K operation only:
K = 5 → ③
1 2. ③ Threshold
for K = 5



median = 4
not possible

Predicate: is median of 'x' possible (x maximum) with given 'k'.

1 1 1 1 2.

+2 +2 +1 → (5)

1 1 (3) 3 3

(3)

K = 5.

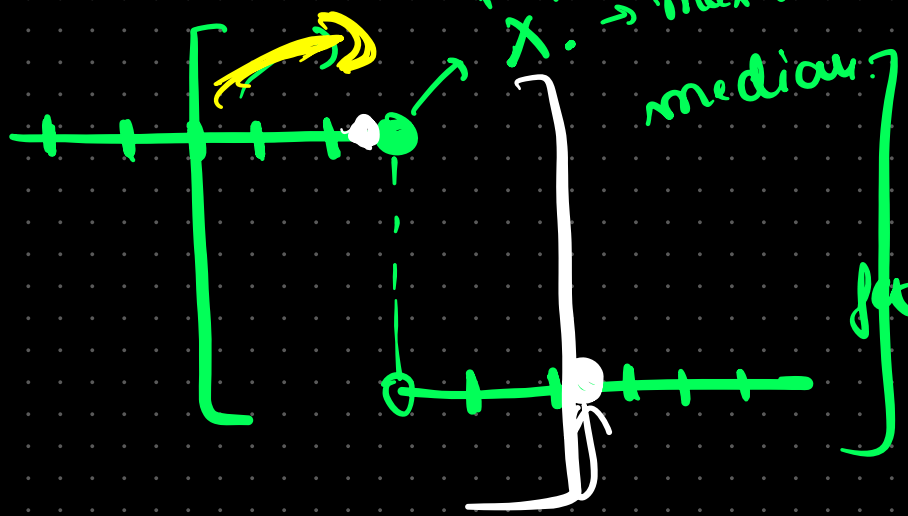
(5) every op.

(+1)

1 1 1 1 2.

+3 +3 +3 → & only $k = 5$ allowed.

α.



for $k = 5$

we can't make max med = 5.

1 1 1 1 2 $\quad\quad$ K = 5.

start = 1
end = 9
→ range b/co which our max median lies.

main

'x' ≤ x ≤ x.

1 1 | 1 1 2

Sum = 0

if (nums[i] < x)

Sum += (5 - 1)

= 4

Sum = 4

mid = (9+1)/2

= '5'.

assumed median 'x'

Sum += (5-1) = 4+4 → (8)

Sum += (5-2) = 4+4+3

= 11

Sum = 11 ✗

K = 5

end = mid - 1

Sum <= K

ans = mid

We know:

(5) × (6)

log n × n.

[n log n]

ans → max median

[mid]

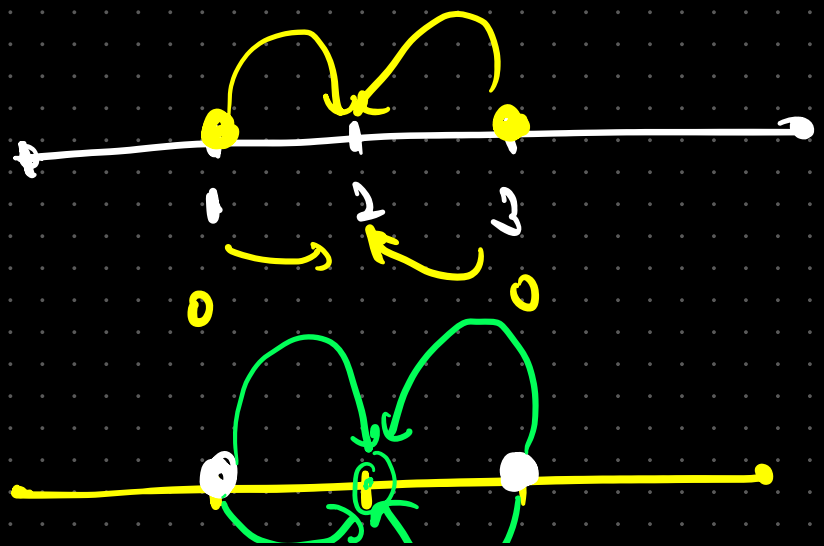
Question :

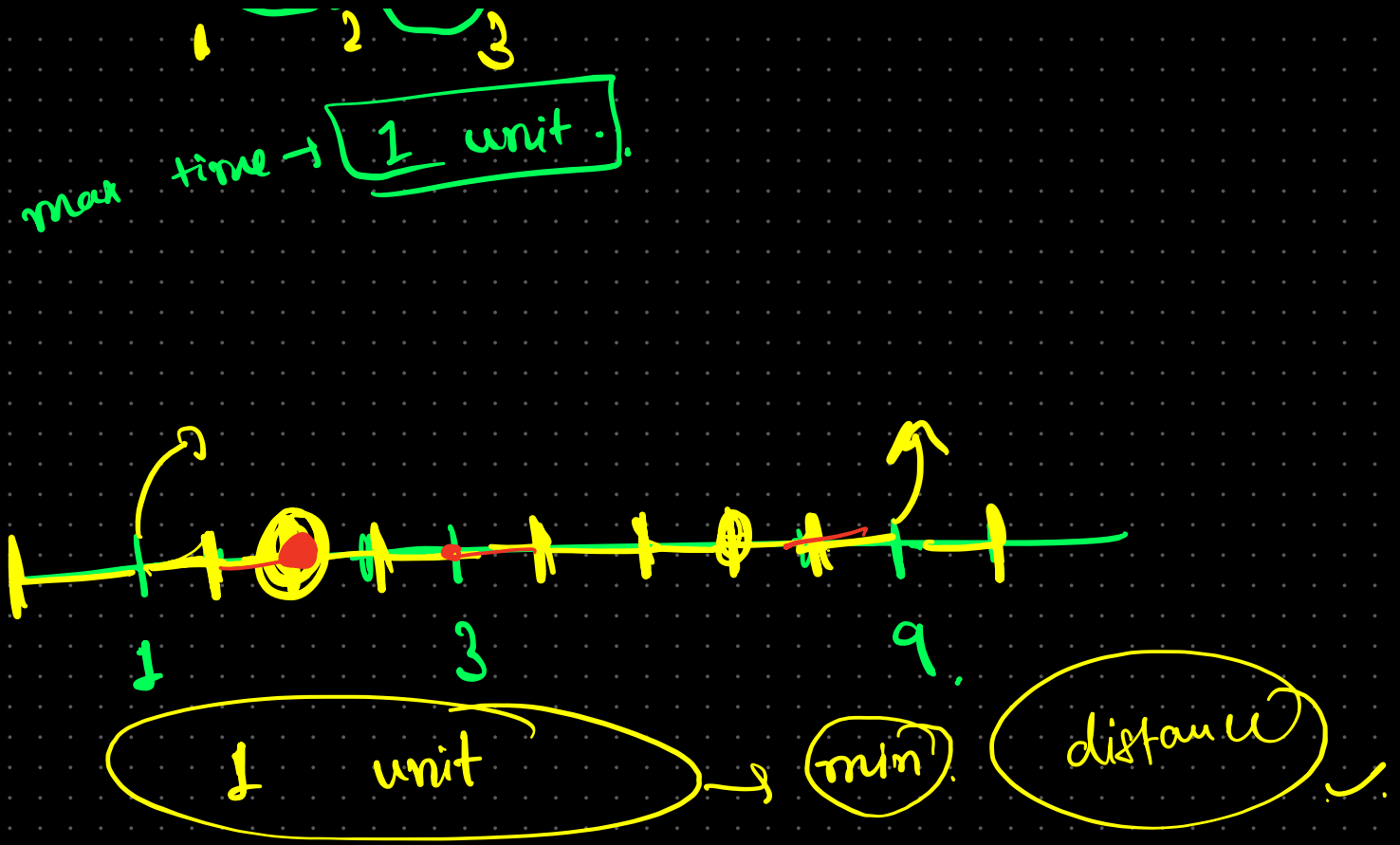
all must come to a single point for meting.

ti

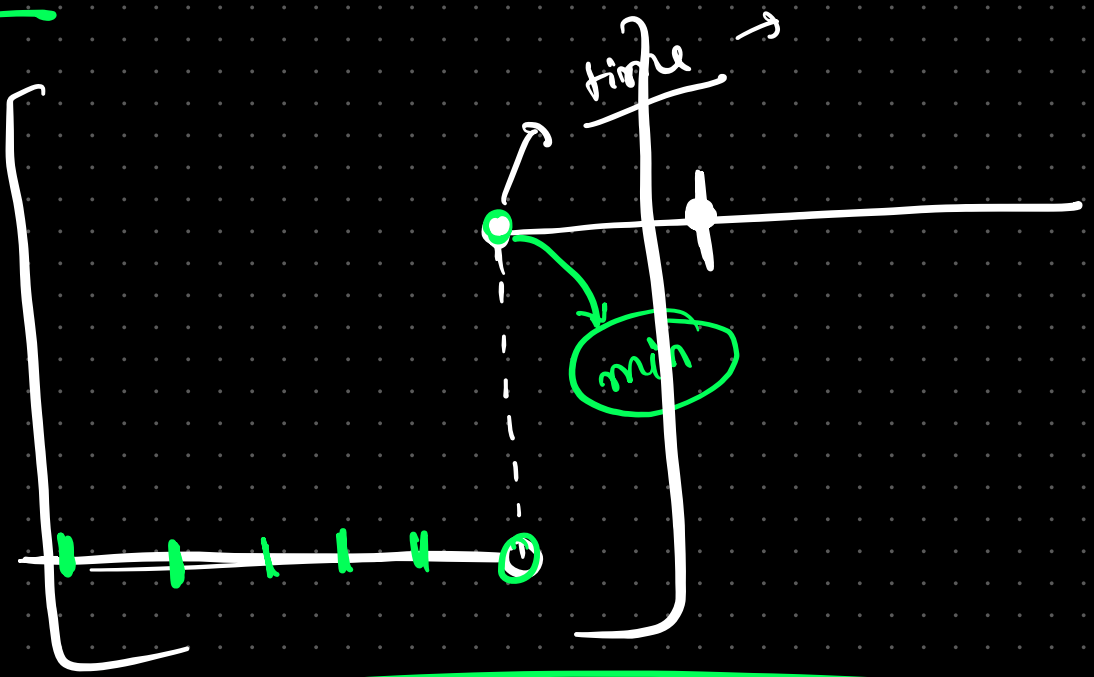
3 1 → Position.

0 0 → time to get ready




1
2
3
max time → 1 unit.
1
3
9.
1 unit
min
distance

Predicate :



$$[1, 5] \cap [4, 9] \cap [11, 15].$$

intersection

$l = -1e9 = -10^9$

$r = +1e9 = 10^9$

val

$[l, r]$

$A \cap B = \emptyset$

(1, 5]

"$\emptyset$"

$[l, r] \rightarrow [-10^9, 10^9]$

range

{U} → intersection

[-10, 15]

$[-10^9, 10^9]$

(1, 5)

[4, 5)

(11, 15]

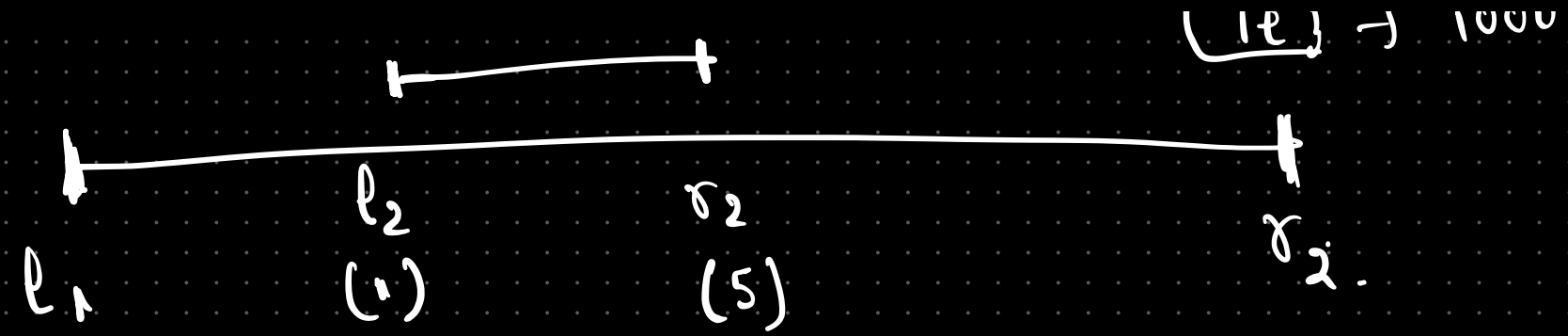
$l = -1e9$

$r = 1e9$

$1e9 = 10^9$

$1e1 \rightarrow 10$

$1e1 \rightarrow 100$



$$new\_lft = \max(lft, l_2)$$

$$= \max(-1e9, 1) = 1.$$

$$new\ lft = 1$$

$$new\_right = \min(right, r_2)$$

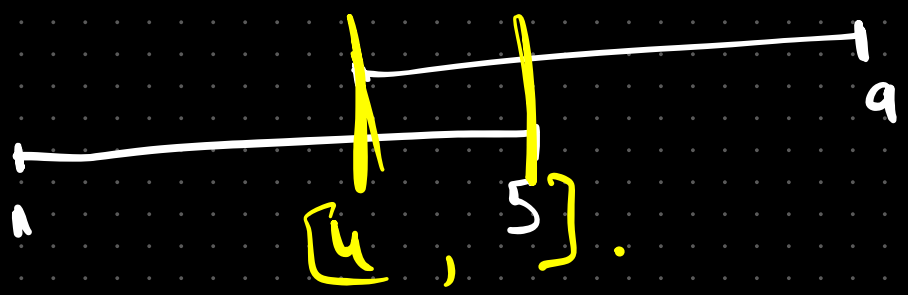
$$= \min(1e9, 5) = 5.$$

new (l, r) = [1, 5] → [4, 9]

new_left = max (1, 4) = 4.

new_right = min (5, 9) = 5

new interval = [4, 5].



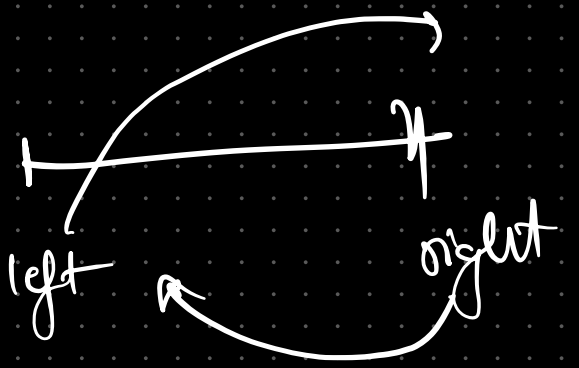
range ⇒ [4, 5] ∩ [11, 15].

new_left = max(4, 11) = 11

new_right = min(5, 15] = 5

new left > new right.

left > right.



Whenever:

left > right. ⟶ null 'ϕ'

{ϕ}.

predicate: that is the time 'T' possible for everyone to meet at a point.

x_1 x_2 x_3. ['T'].

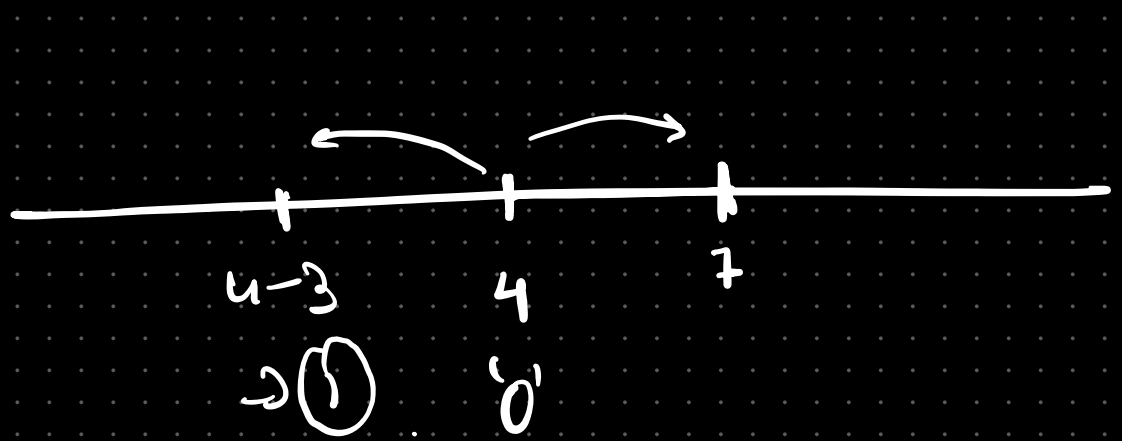
$(x_1 - T,\ x_1 + T)$

$(x_2 - T,\ x_2 + T)$

$(x_3 - T,\ x_3 + T)$

} intersection of all 3 to be done.

if intersection $\{\phi\}$



[1, 7] → anywhere b/w [1,7].

True (l, r) → [l <= r] →

if [l > r] → ϕ → False.

start = 0

end = 1e9

while (start <= ~~mid~~ end)

mid = (start + end) / 2.

if (pred (mid) == True) {

end = mid - 1 ;

}

else start = mid + 1 ;

```
Predicate ( mid ) {.          ti + (ni - n0)
    l = -1e9
    r = 1e9
    for ( i = 0 ⟶ n ) {.
        l = max ( l, Pos[i] - (mid - t[i])
        r = min ( r, Pos[i] + (mid - t[i])
        if (l > r) return false
    }
    return true;
```

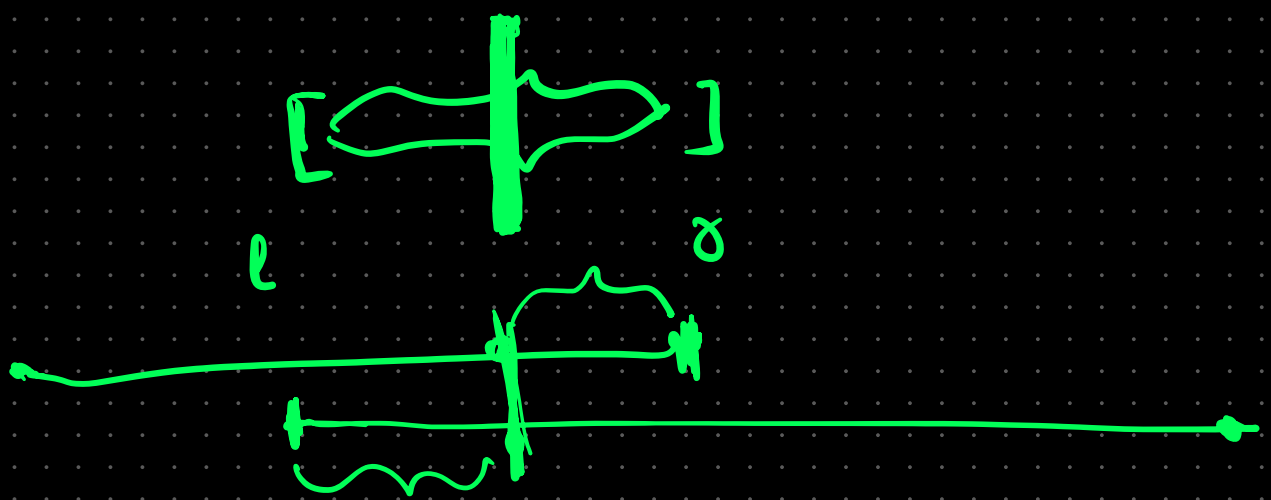
min time:

$l = -1e9$
$r = 1e9$] →

positions

$(x[i]+T,\ x[i]+T)$

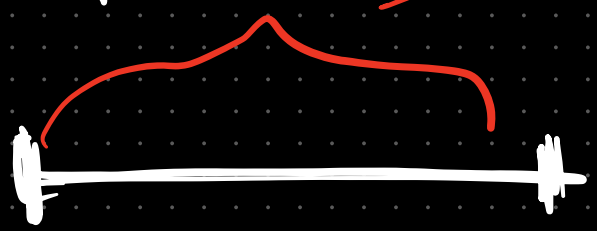
(l, r) $\left(\frac{l+r}{2}\right)$

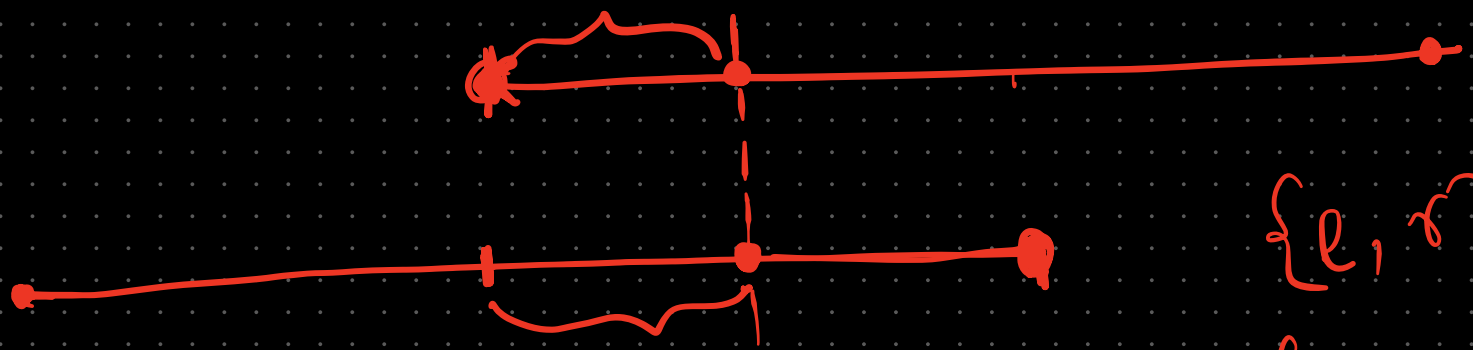


min time

[l, r]

meeting





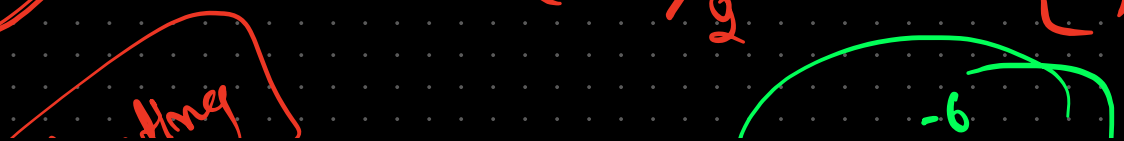
{l, r}

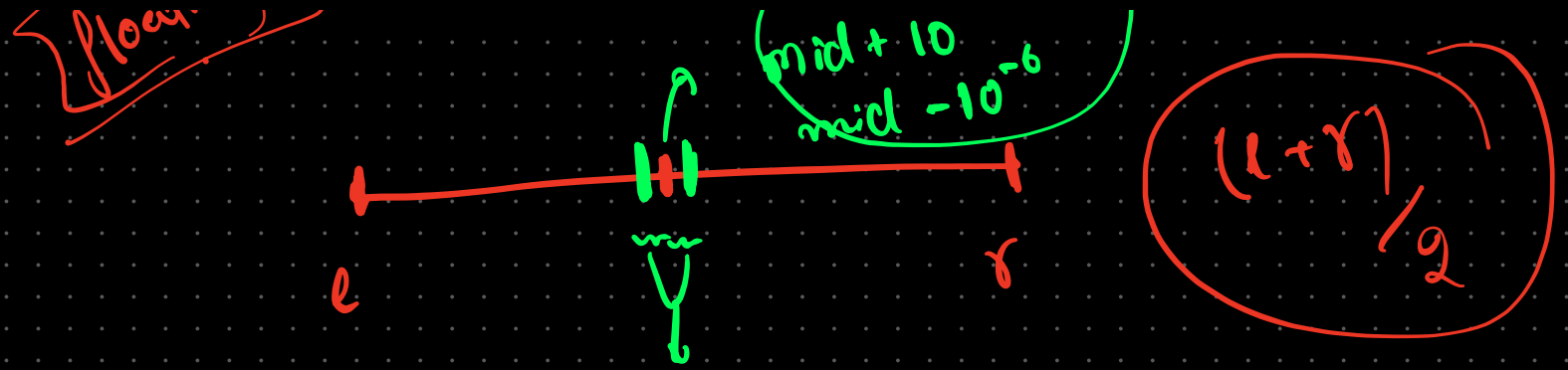
{5, 5}

mid

(1+3)/2 → (4)

{1, 3}





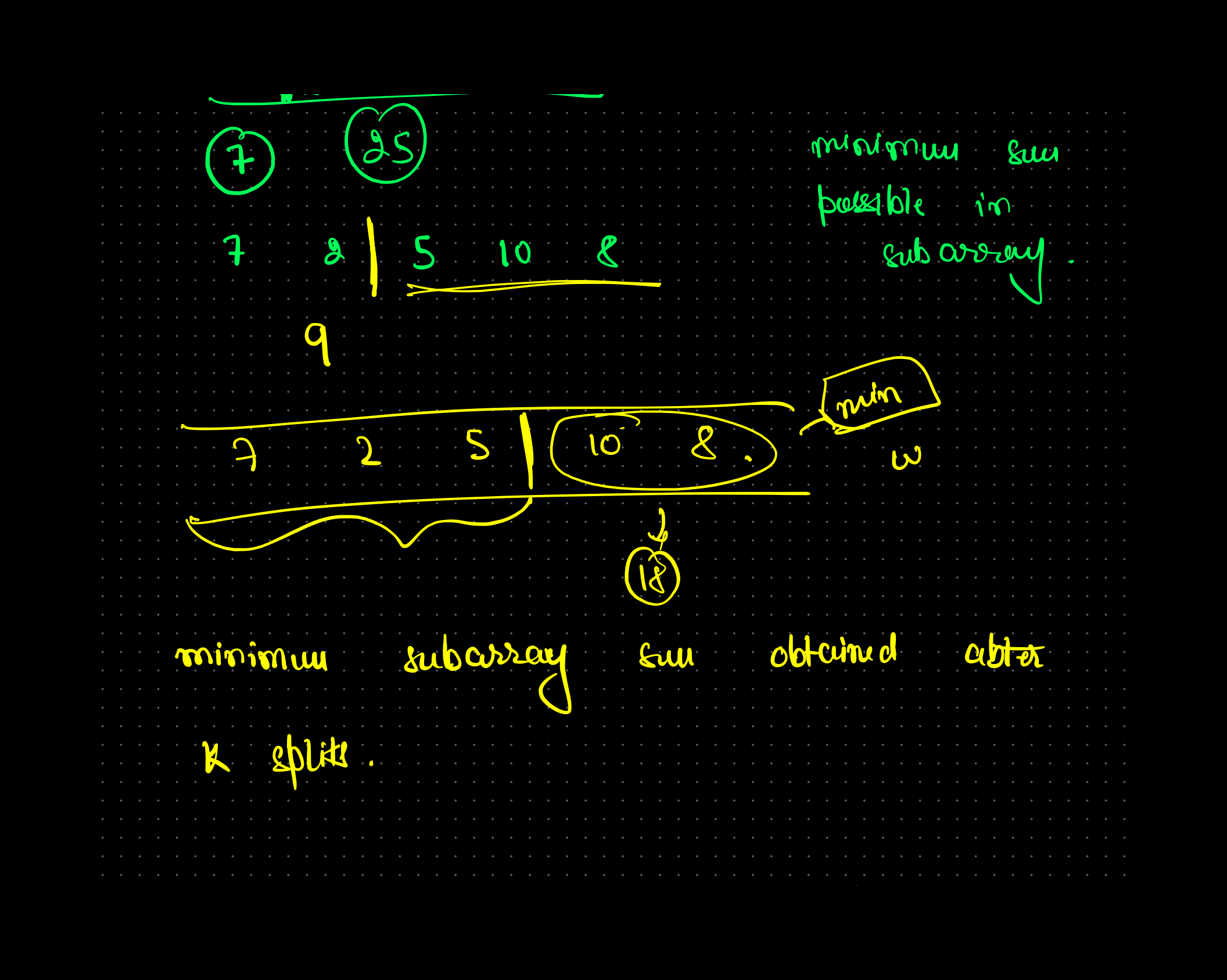
K → no. of splits.

array of size 'n'.

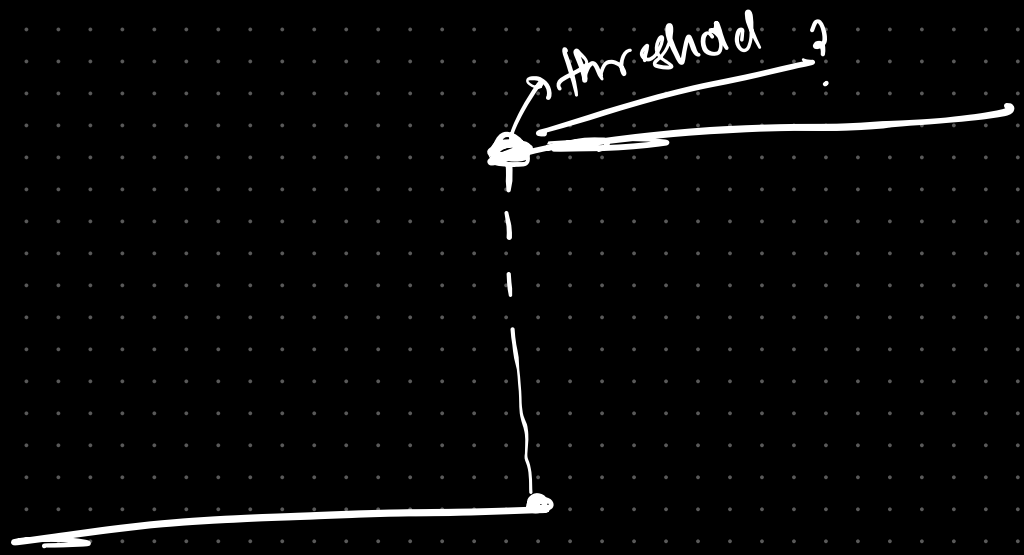
[maximum sum that is possible in one of the subarray]

| 7 | 2 | 5 | 10 | 8 |

K = 2


7
25
minimum sum possible in subarray.
7 2 | 5 10 8
9
min
7 2 5 | 10 8
18
minimum subarray sum obtained after
k splits.

mid → sum (potential min)



x - split

1 2 3 4 5

Bit manipulation:

Remember: any no. can be represented as a sum of power of 2.

001
010
011
100
:

→ in 0's & 1's.

:)

integer : signed : -2^{30} to $2^{31}-1$

total 32 bits are used for int.

unsigned int : 0 to $2^{32}-1$

and : '&' → if all one's then '1' otherwise '0'

'|' → if any bit is one

then '1'

'^' if odd no. of 1's

then 1

if even no. of 1's then 0.

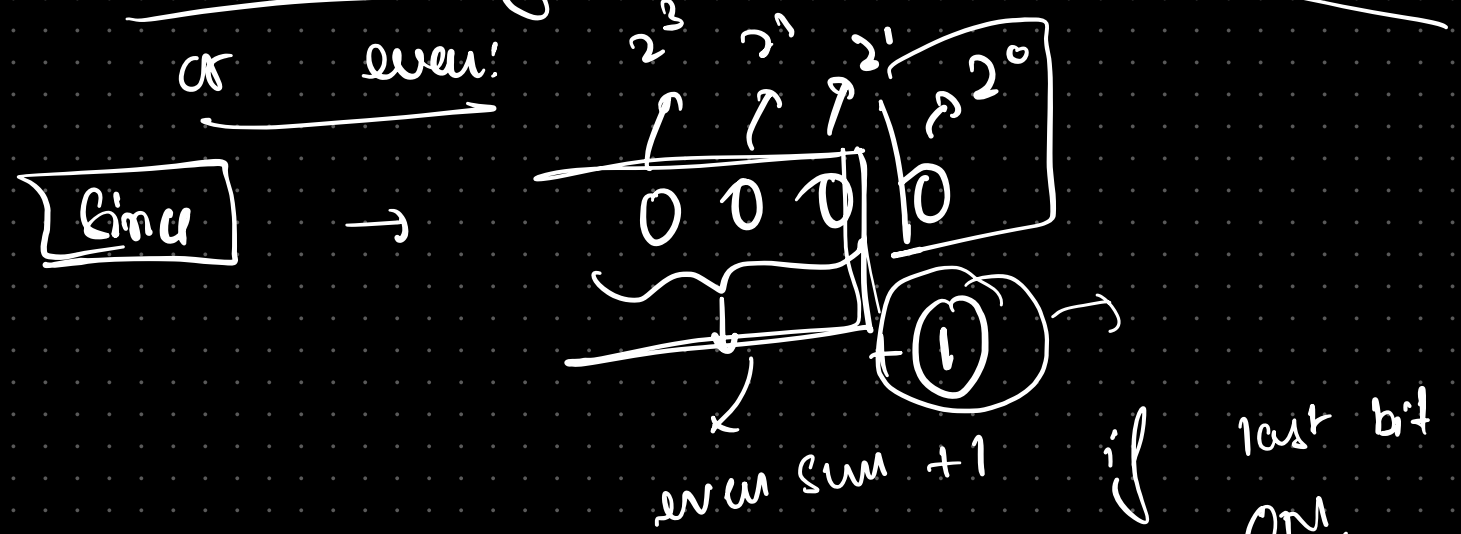
<< left shift.

001 << 1

0010

0001 ⇒ when 001 >> 1

to find if an integer is odd or even

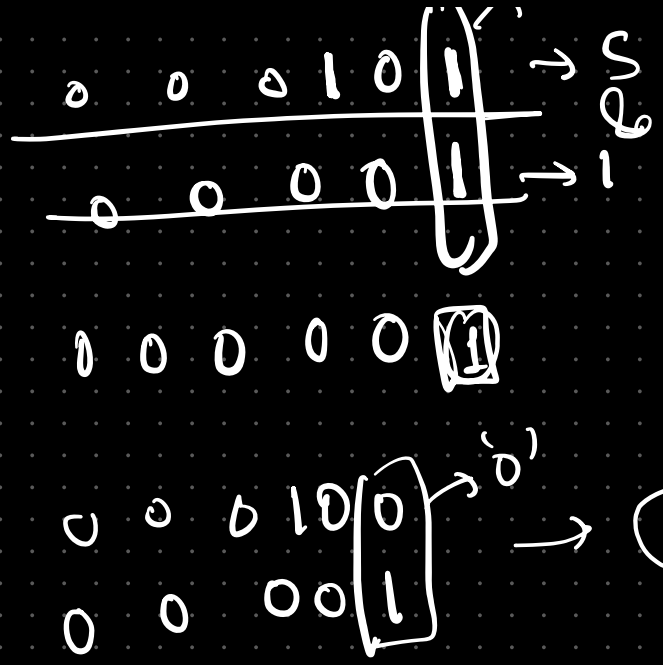


mathematic → a%b → time consuming.

if a is odd or even

a & 1

a → (5) → odd



right shift ⇒ divides then by 2^k.

left shift ⇒ multiply by 2^n.

100 << 1

1000 → '8'

1000 >> 1

10 → '2'

mid = (start + end) / 2.

mul & div.

+ -

(start + end) >> 1 →

how many bits are set.

1 31

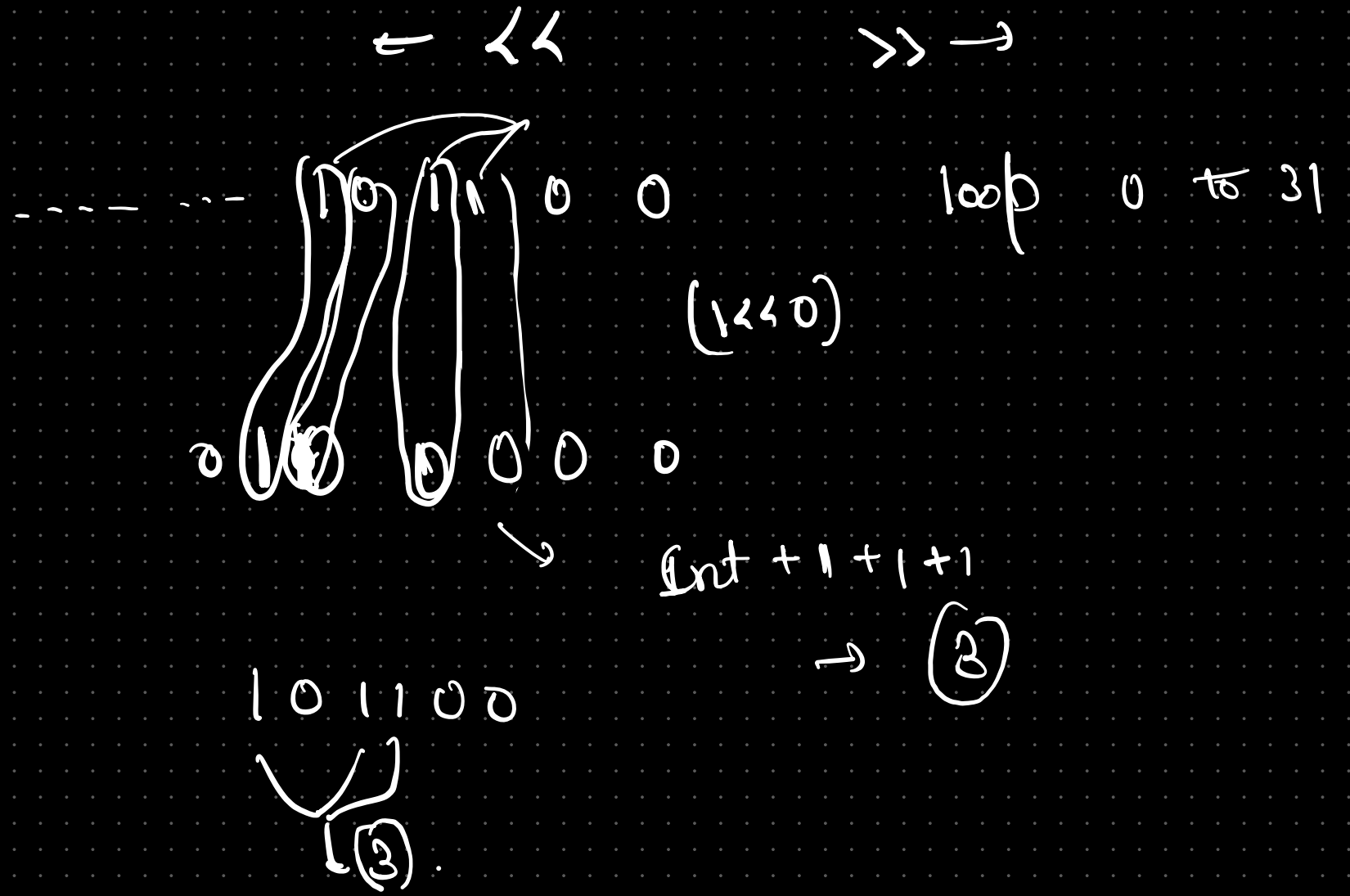
Count = 0.

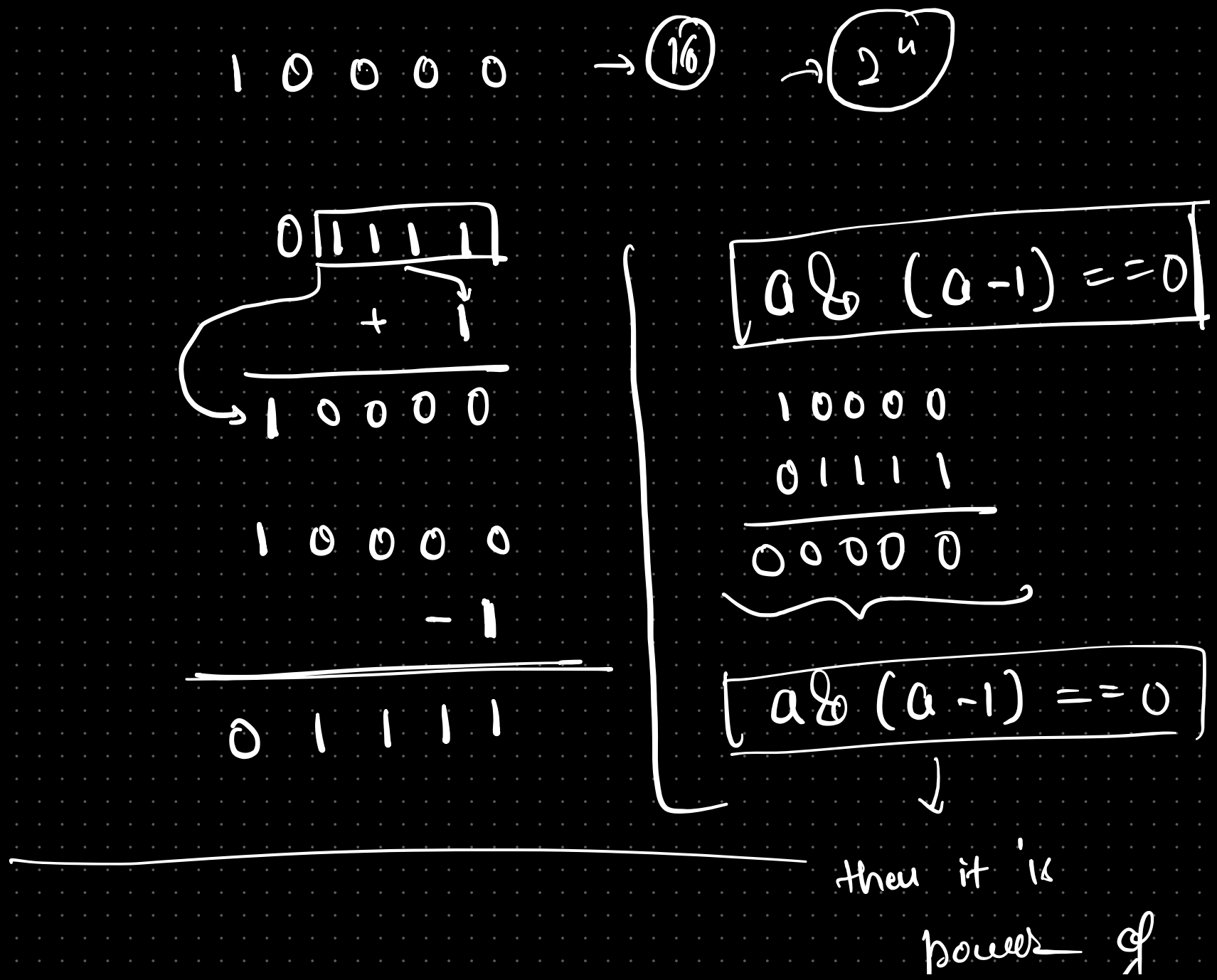
```
for (i = 0 ⟶ 31)
    if(a & (1 << i)) {
        count++;
    }.
```

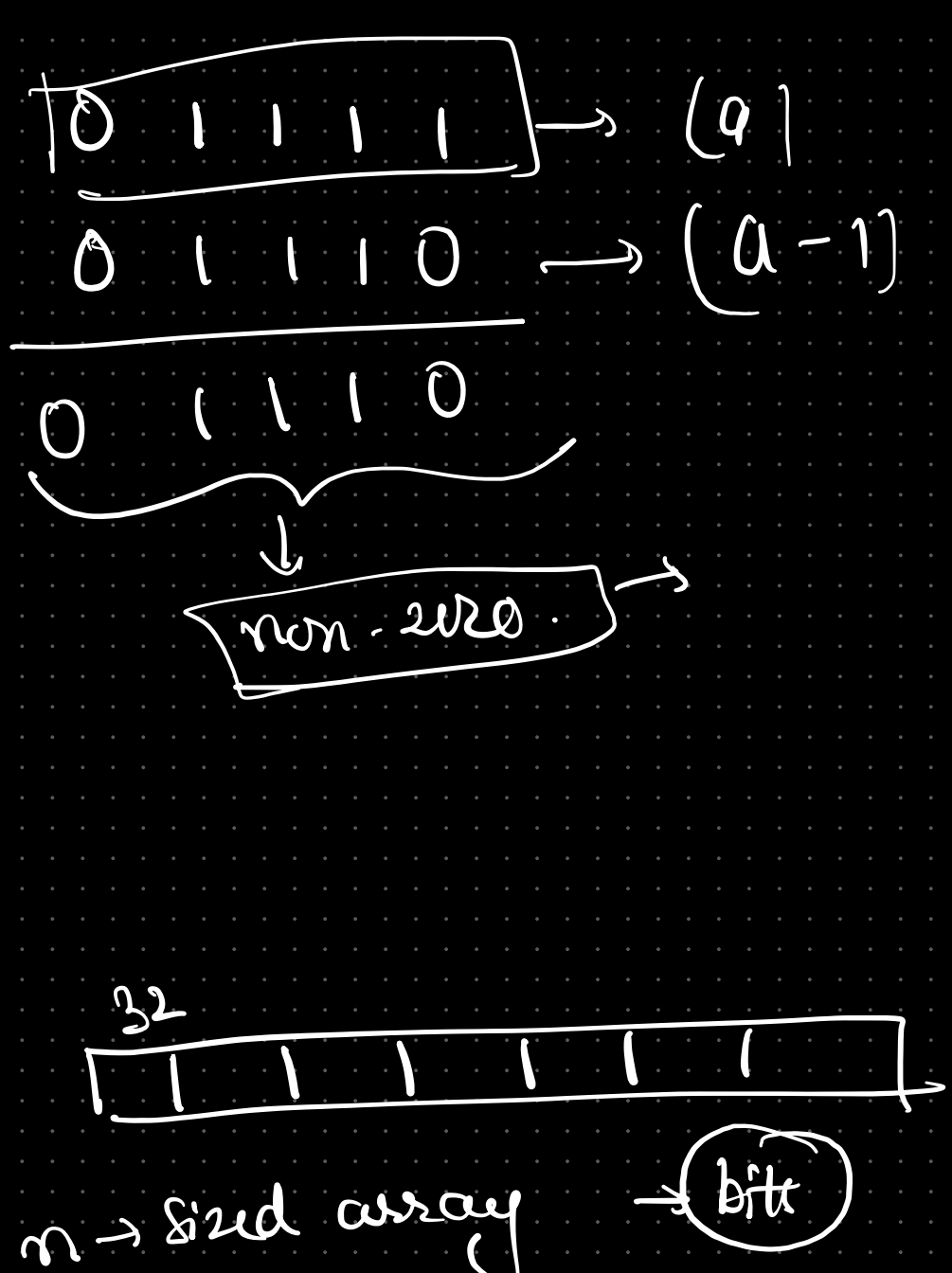
← << >> →

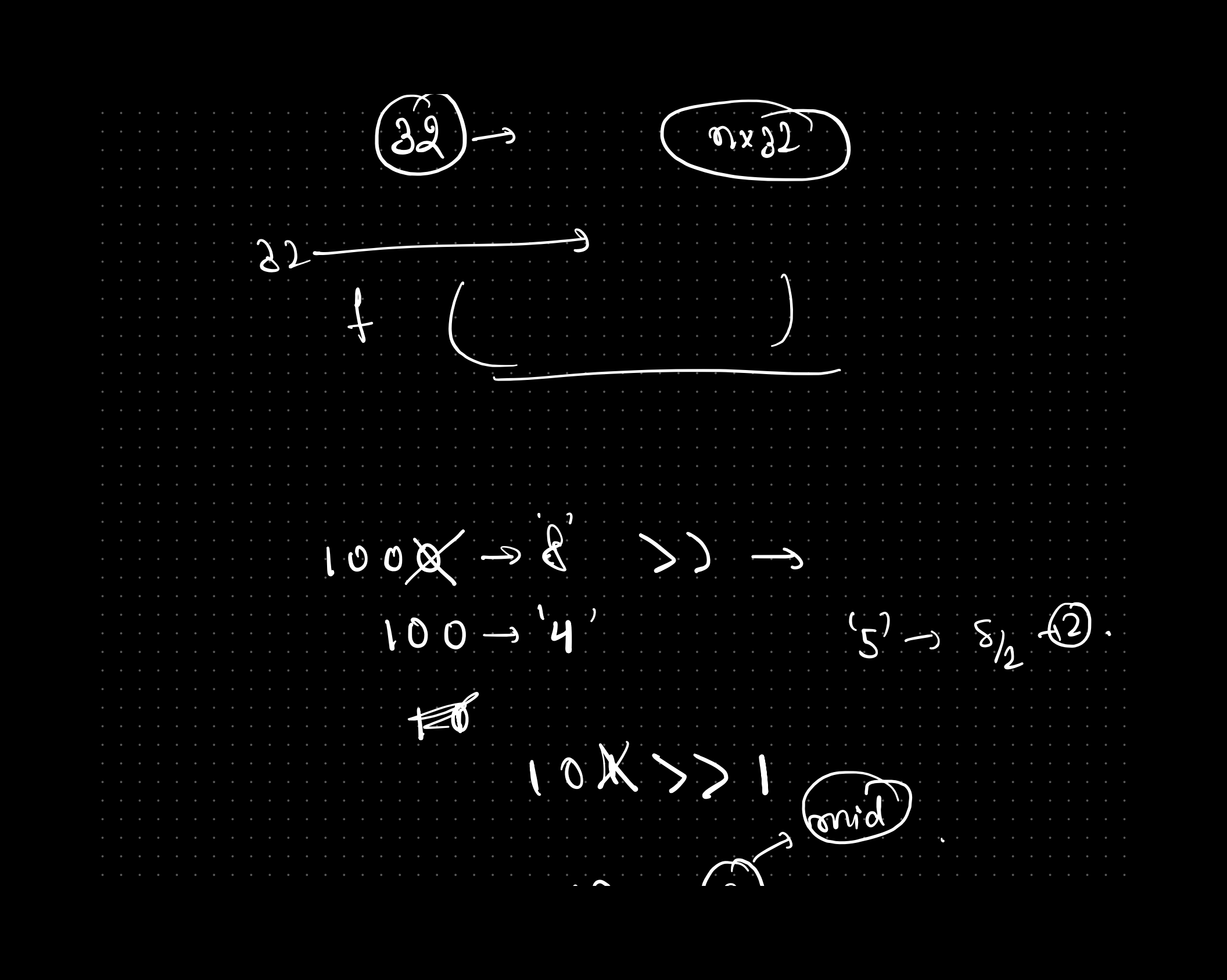
1 0 1 1 0 0

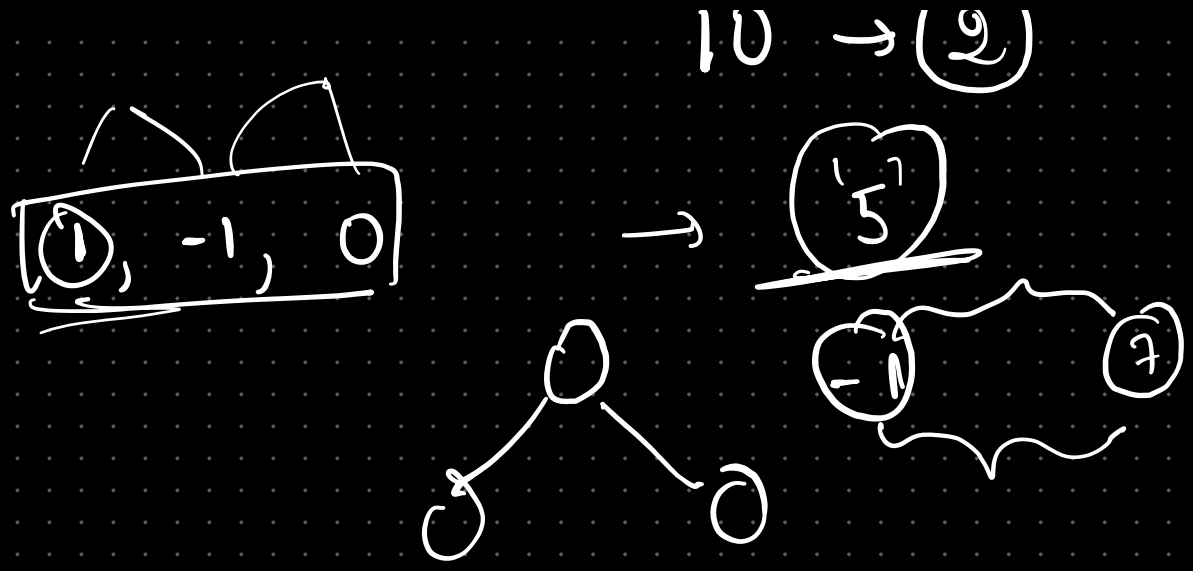
loop 0 → 31


← <<
>> →
loop 0 to 31
(1<<0)
int +1 +1 +1
→ 3
101100
3


10000 → 16 → 2^4
01111
+ 1
10000
10000
- 1
01111
a & (a-1) == 0
10000
01111
00000
a & (a-1) == 0
then it is
power of


0 1 1 1 1
(a)
0 1 1 1 0
(a-1)
0 1 1 1 0
non-zero
32
n → sized array
bit


32 →
n x 32
32
f ( )
100 → '8' >> 1 →
100 → '4'
'5' → 8/2 = 2.
100 >> 1
mid


10 → (2)
1, -1, 0
5
-1
7