Basic Mathematics for DSA

Foundation Course on Data Structures & Algorithm - Part I

Love Babbar • Lesson 15 • May 14, 2022

(w/) -> O(n)

Basic Maths :->

=> Prime number -> flowchart
-> Code

i/p -> number -> n

o/p -> Prime or Not

n = 10

10 -> rem = 0 -> not a prime no

-> 2, 3, 4, 5, 6, 7, 8, 9

Prime

2 -> <n

% -> 0

2 -> (n-1)

2 - 9

Code ->

```
bool isprime ( int n )
{
    for ( int i = 2 ; i <= n/2 ; i++ )
    {
        if ( n % i == 0 )
            return false;
    }
    return true;
}
```

$n > 1$ ← assumption

for loop → $O(n)$

if → $O(\frac{n}{2})$

return false; → $O(n)$

$\rightarrow \underline{O(n)}$

10

2 → pn

2 – 5

100

2 → 50

$x \times 2 = n$

$x = \frac{n}{2}$

$x \times 2 = n$

$x = \frac{n}{2}$

$< n$

$\frac{n}{2}$

500

2 → 250

$x \times 2 = n$

$x = \frac{n}{2}$

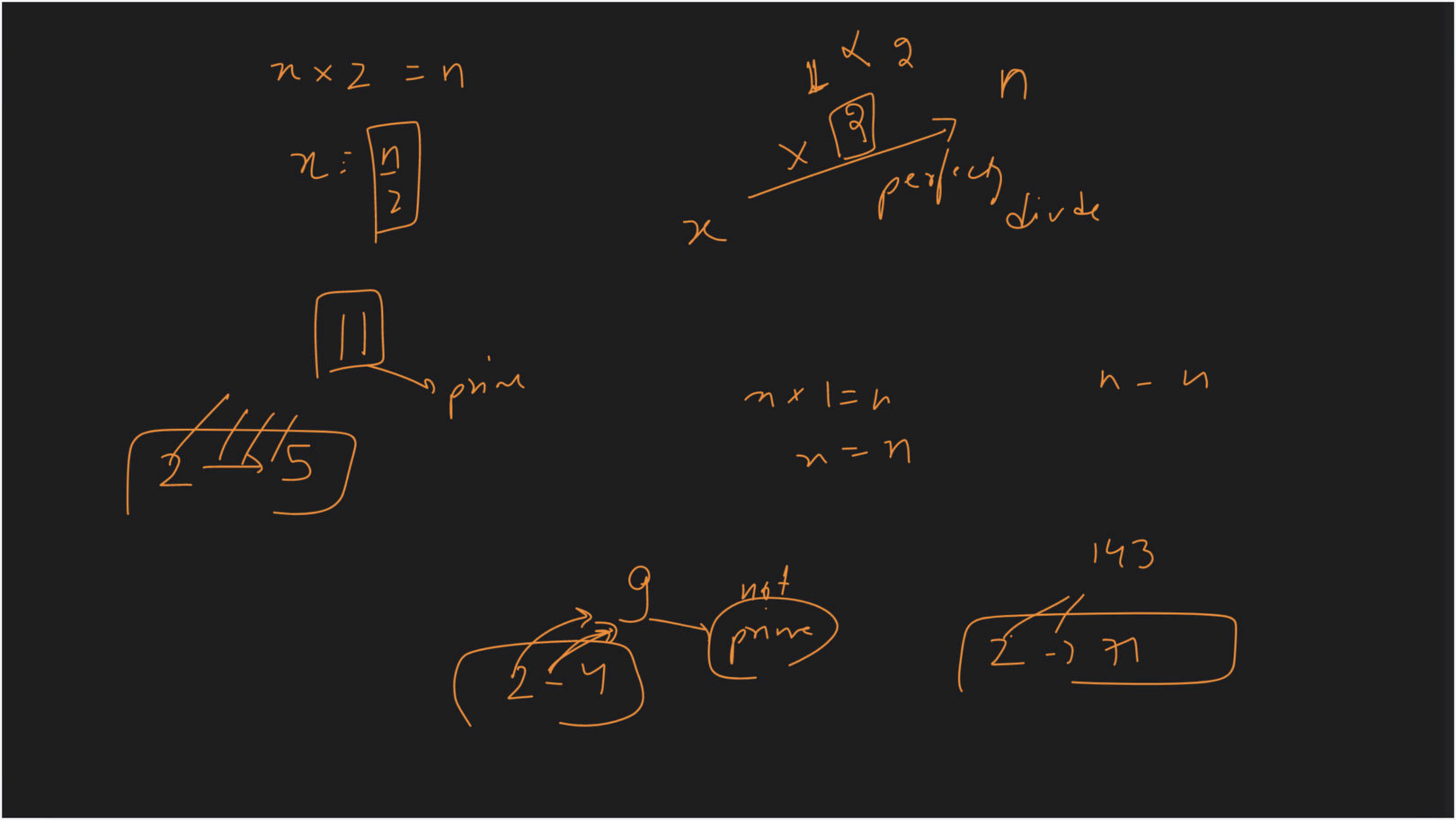
n → perfectly divide

table me aata hoga

$n \times 1 = n$

$n \times 2 \rightarrow$

n


n × 2 = n
n : n/2
11
prime
2 → 5
n × 1 = n
n = n
n - n
2 ≤ n
x
perfectly divide
9
not prime
2 → 4
143
2 → 11

0 → n

0 → n/2

0 → √n → n/2

i = 0; i<=√n ; i++

or

i * i <= n

√n

25 → prime

not prime

2 → √25

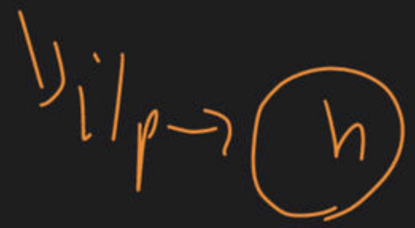
5

2 × √2 × 2√2

2 × 1.41 = 2

2 → 2.5

not prime



For N

(O(h))

for(i -> 0 -> n) ⟶ O(n)

for(i -> 0 -> √n) -> O(√n)

for(i = n ; i >= 0 , i = i/2)

↳ O(log n)

Binary search

Time → n ms

→ LeetCode → "Count Primes"

↳ Brute force → TLE

↓

Time Limit Exceeded

→ Optimisation

↳ Sieve of Eratosthenes → data that

↳ ~~ratta~~ maarlia

↓

disAdv

forget

↓

why?

No answer

95.5%

↓

Dry RUN

→ SOE

i/p → n n=10

o/p → 1 → n → count of prime no

even no

n%2 == 0 → true

n&1 == 0

false

1 — 1 2 3 5 7 8 9 10

ans = 4

Brute

Large → n = 100000

→ Time large → optimise K brute

o/p → heavy time consuming

→ SOE → int n = 30

5.5 × 2 = 11

2 × 2 = 4

cnt = 1

I 2 | II 3 | III 5 | IV 7 | V 11 | VI 13 | VII 17 | VIII 19 | IX 23 | X 29

Algo

(I) → mark every no prime initially

(II) → 0, 1 → not prime

(III) → table-wise non-prime mark karde

1 — 30

cnt = 10

Code

int arr[10]

for (j = 2*i ; j <= n ; j = j+i)

j*i

2 × 1 = 2

2 × 2 = 4

2 × 3 = 6

2 × 4 = 8

2 × 5 = 10

2 × 6 = 12

2 × 7 = 14

2 × 8 = 16

2 × 9 = 18

2 × 10 = 20

not prime

dynamic array

vector<int> arr(n);

arr.size() ⟶ exact size

vector<int> arr(n, 1);

n

arr

vector <bool> prime (n+1, true)

n = 30

n

0 → 25

n+1

0 → 30

→ DRY RUN

i/p → n = 10

O(n+1)
O(n)

cnt = 0 2 4 7

① int cnt = 0

② vector<bool> prime(n+1, true)

prime: 0 1 2 3 4 5 6 7 8 9 10

③ prime[0] = prime[1] = false;

④ for(int i = 2 → <n)

{ if(prime[i])

{ cnt++; → cout << i

for(int j = 2*i; j < n; j = j + i)

prime[j] = 0;

}

i = 6

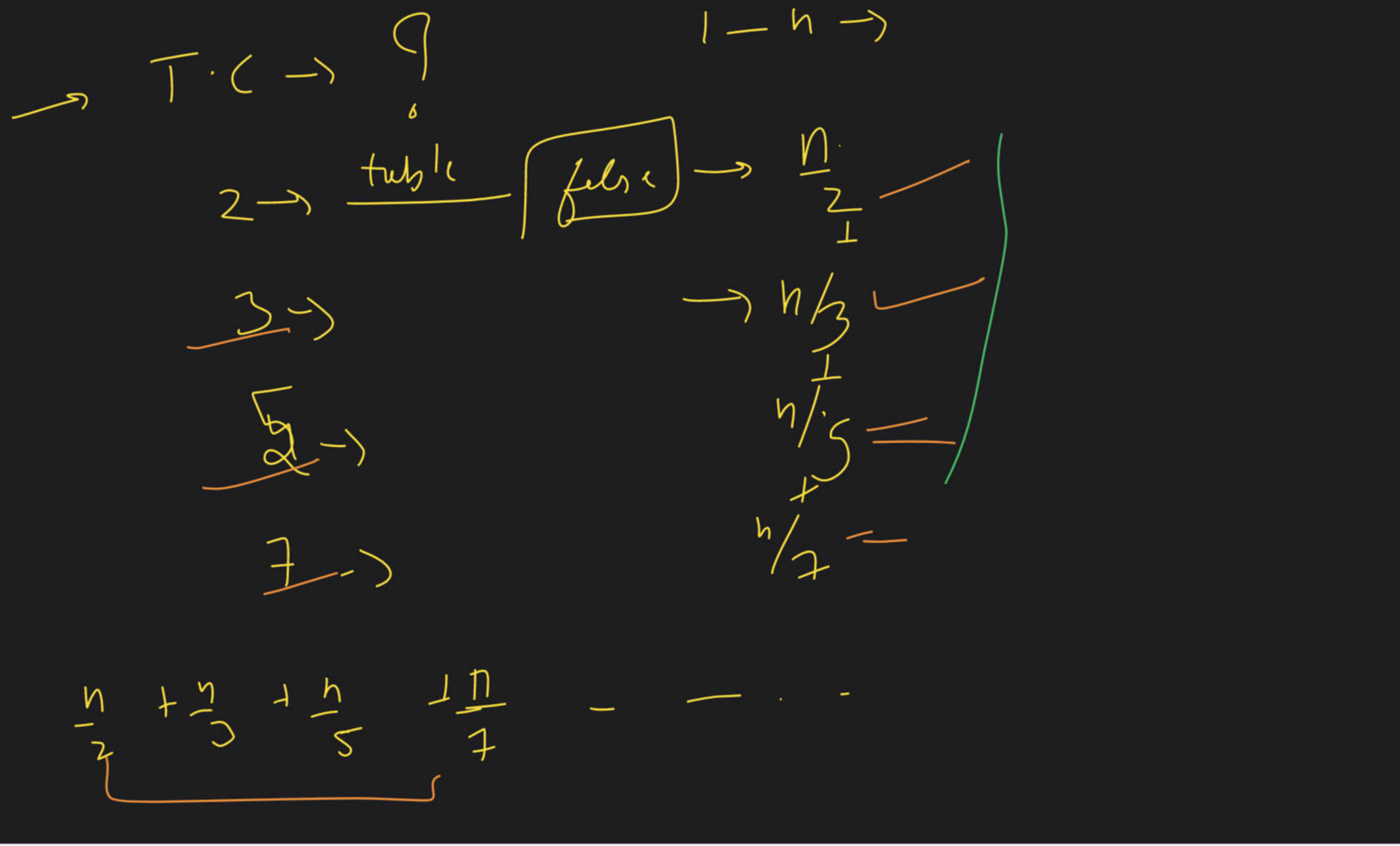
j = 2*6 = 12

i = 8

j = 2*8 = 16

i = 7

j = 2*7 = 14


1 — n →
T.C →
q
2 →
tuple
false
→ n/2
3 →
→ n/3
5 →
n/5
n/7
7 →
n/2 + n/3 + n/5 + n/7 — — . .

$n \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \ldots \right)$ lallu

int arr[n] ✗

H.P → H/W

H.P → $\log(\log n)$

S-C → v(n+1)

$O(n+1)$

$O(n)$

shoutout to bhaiya → $O(n * \log(\log n))$ → n^2

→ 1 — n → print no

n = 10

count → 4

int n   cin>>n

→ Segmented Sieve → H/W

→ 5 dry Run

GCD

HCF

GCD (12, 8)

2	12
2	6
3	3
	1

2	8
2	4
2	2
	1

Euclid's Algo:-

$$gcd(a, b) = g(a - b, b)$$

$$gcd(a \% b, b)$$

(72, 27)

$12 = 2 \times 2 \times 3$

$8 = 2 \times 2 \times 2$

HCF $= 2 \times 2 = 4$

$gcd(a, b) = g(a-b, b)$ gc

bada small

$gcd(12, 8)$

$\rightarrow gcd(12-8, 8) = gcd(4, 8)$

$= gcd(8, 4)$

$gcd(8, 4) = gcd(8-4, 4)$

$= gcd(4, 4) \rightarrow gcd(4-4, 4)$

$gcd(0, 4)$

ans

gcd(72, 24)

↳ gcd(72 − 24, 24) → gcd(48, 24)

gcd(72, 24)

↓

gcd(72 % 24, 24)

gcd(0, 24)

↓

gcd(48, 24)

↓

gcd(48 − 24, 24)

↓

gcd(24, 24)

↓

gcd(24 − 24, 24)

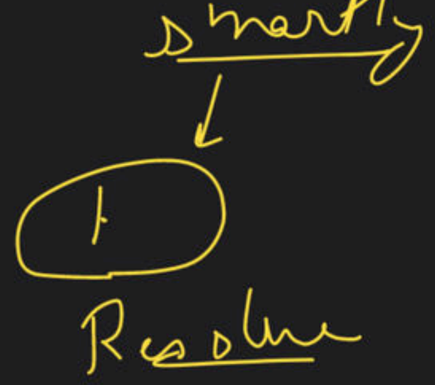
↓

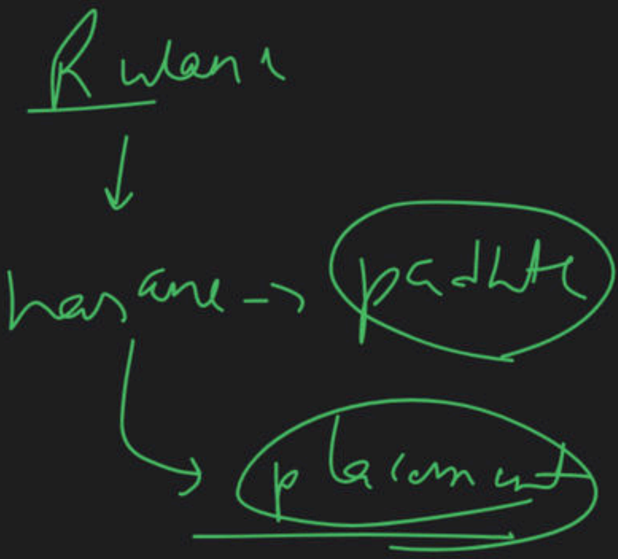
gcd(0, 24)

↓

ans


1 min break


Smartly
1
Resolve


Rules
hardware -> padlock
placement

Code

```
int gcd (int a, int b)
{
    if (a == 0)
        return b;

    if (b == 0)
        return a;

    while (a != b)
    {
        if (a > b)
            a = a - b
        else
            b = b - a;
    }
    return a;
}
```

a == b
break;

-> minus

return b;

O(n)

O(n^2)

i * i <= n

a - b

a % b

N/w

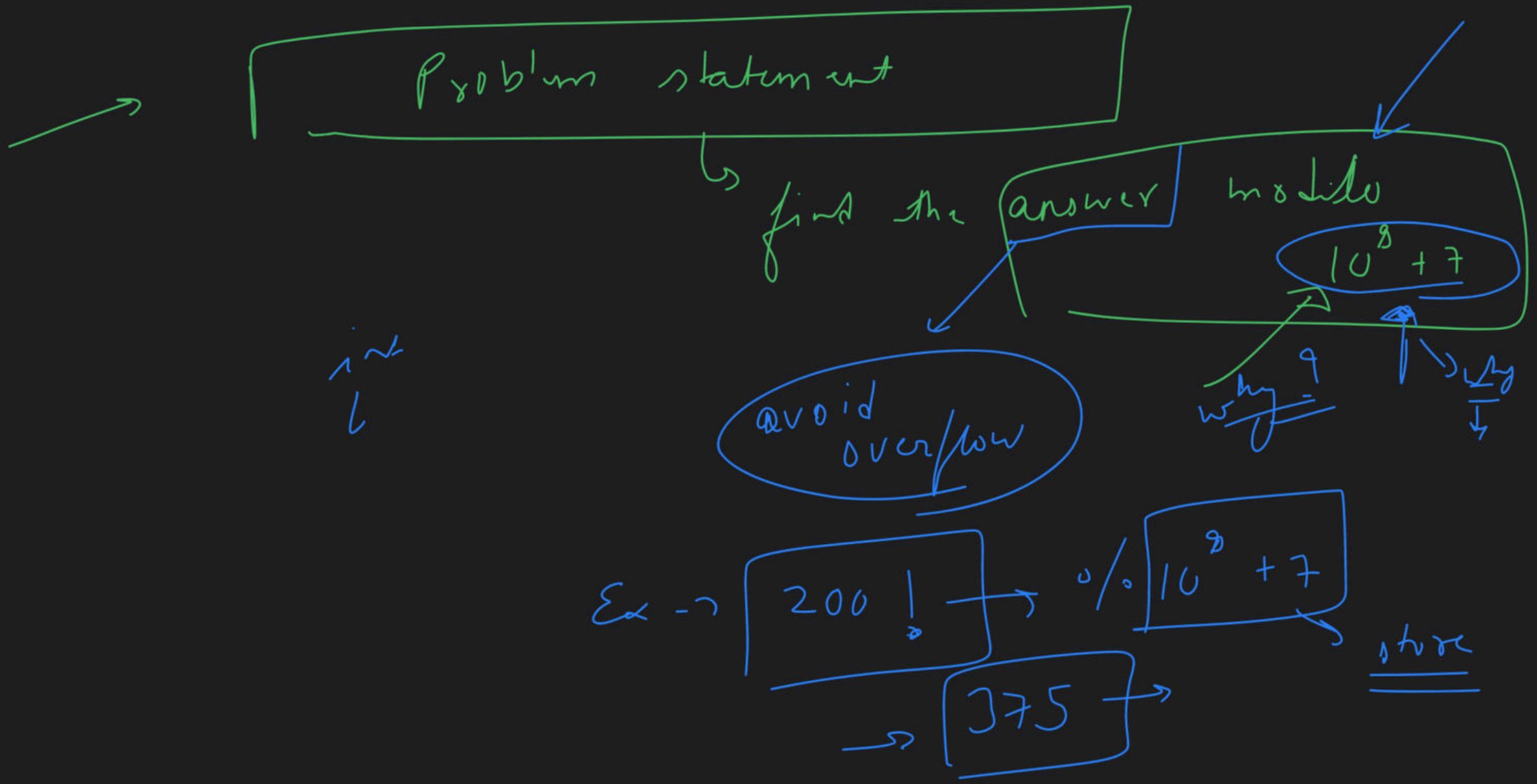
→ $lcm(a, b) \times gcd(a, b) = a * b$

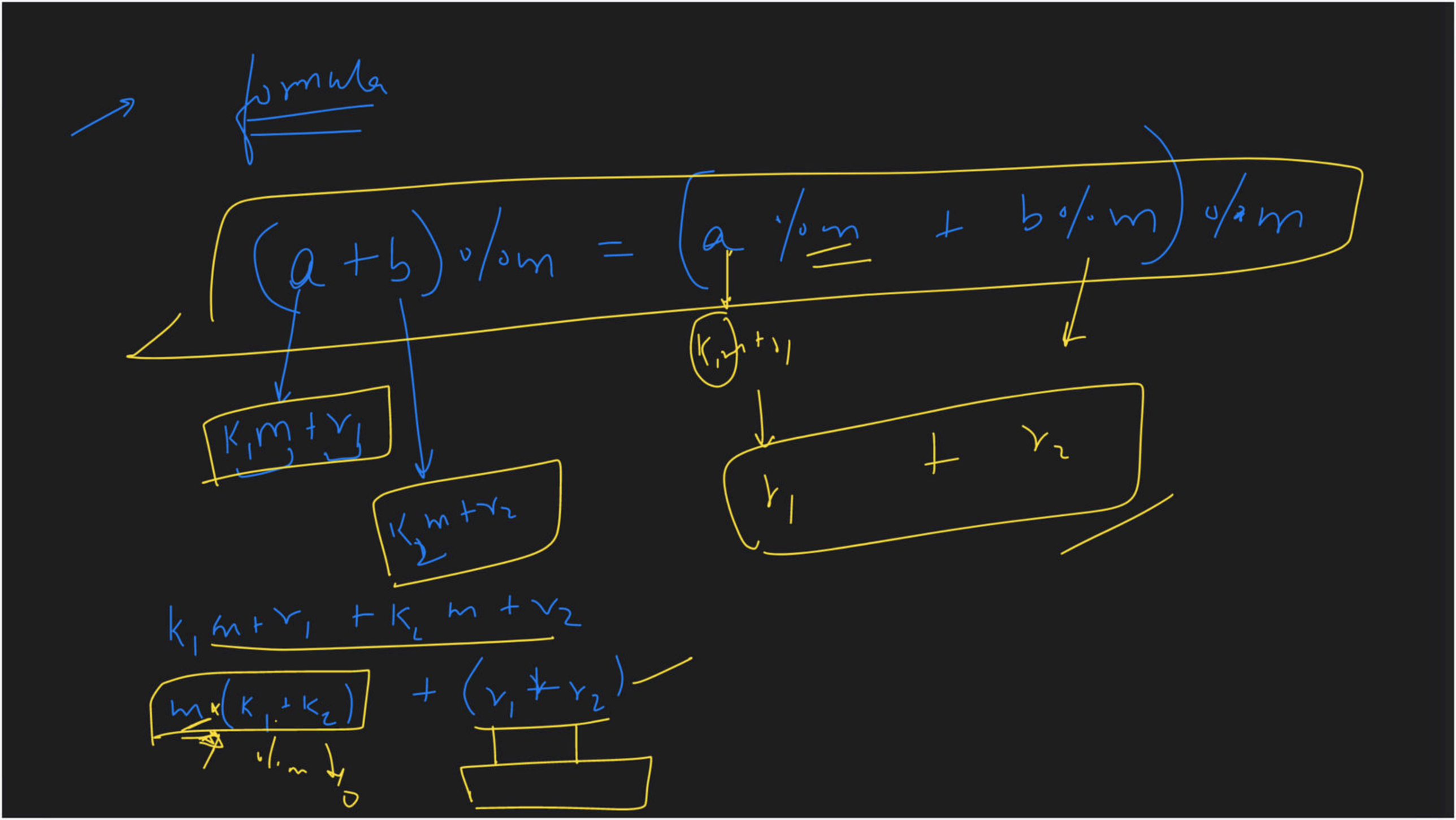
→ co-prime

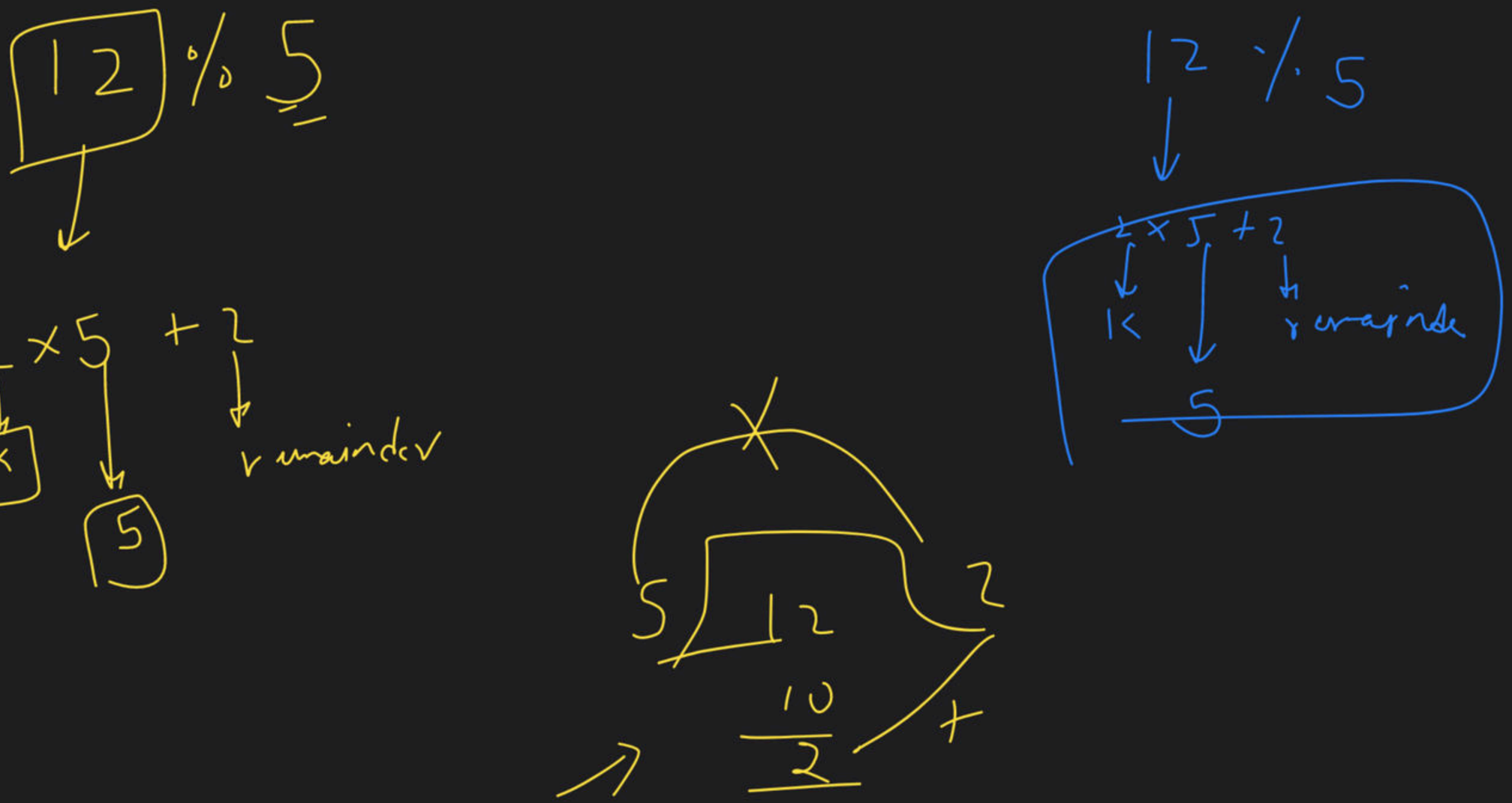
$gcd(a, b) = 1$

Modulo arithmetic

10 % 5 → 0
11 % 5 → 1
12 % 5 → 2
13 % 5 → 3
14 % 5 → 4
15 % 5 → 0
16 % 5 → 1


Problem statement
find the answer
modulo
$10^9 + 7$
avoid overflow
Ex -> 200!
% $10^9 + 7$
375
store


formula
(a+b) %m = (a %m + b %m) %m
K1m + r1
K2m + r2
K1m + r1 + K2m + r2
m(K1 + K2) + (r1 + r2)
%m → 0


12 % 5
2 × 5 + 2
1<
5
remainder
12 % 5
1< × 5 + 2
remainder
5
5
12
2
10
2
+

$(a+b)\%m = (a\%m + b\%m)\%m$

$(a-b)\%m = (a\%m - b\%m + m)\%m$

Unacceptable

Runtime

why ?

$a - b$

$0 - m-1$

$0 - m-1$

$-m$ $\quad +m \rightarrow$

$\Rightarrow (a * b) \% m = (a \% m) * (b \% m)$

Next $\rightarrow$ $(a/b) \% m \rightarrow$ inv modul LD $\}$ CP, concepts

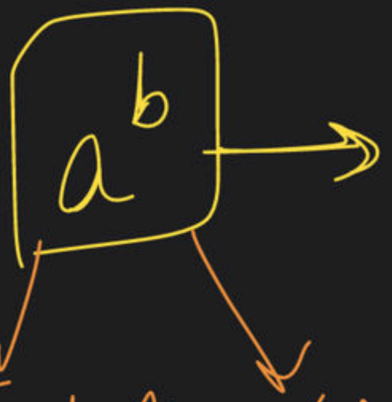
Acel

Codeforces $\rightarrow$ Modulo arithmetic for beginners

article

fast exponentiation

pow(a, b) ?



flowchart

code

#1 approach → Brute force

$a^b = a * a * a * a \cdots * a$ (b times)

```
int ans = 1
for (int i = 0; i < b; i++)
    ans = ans * a
return ans;
```

Real

→ optimisation

$a^b \longrightarrow (a^{b/2})^2$

b is even → $a^b = (a^{b/2})^2$

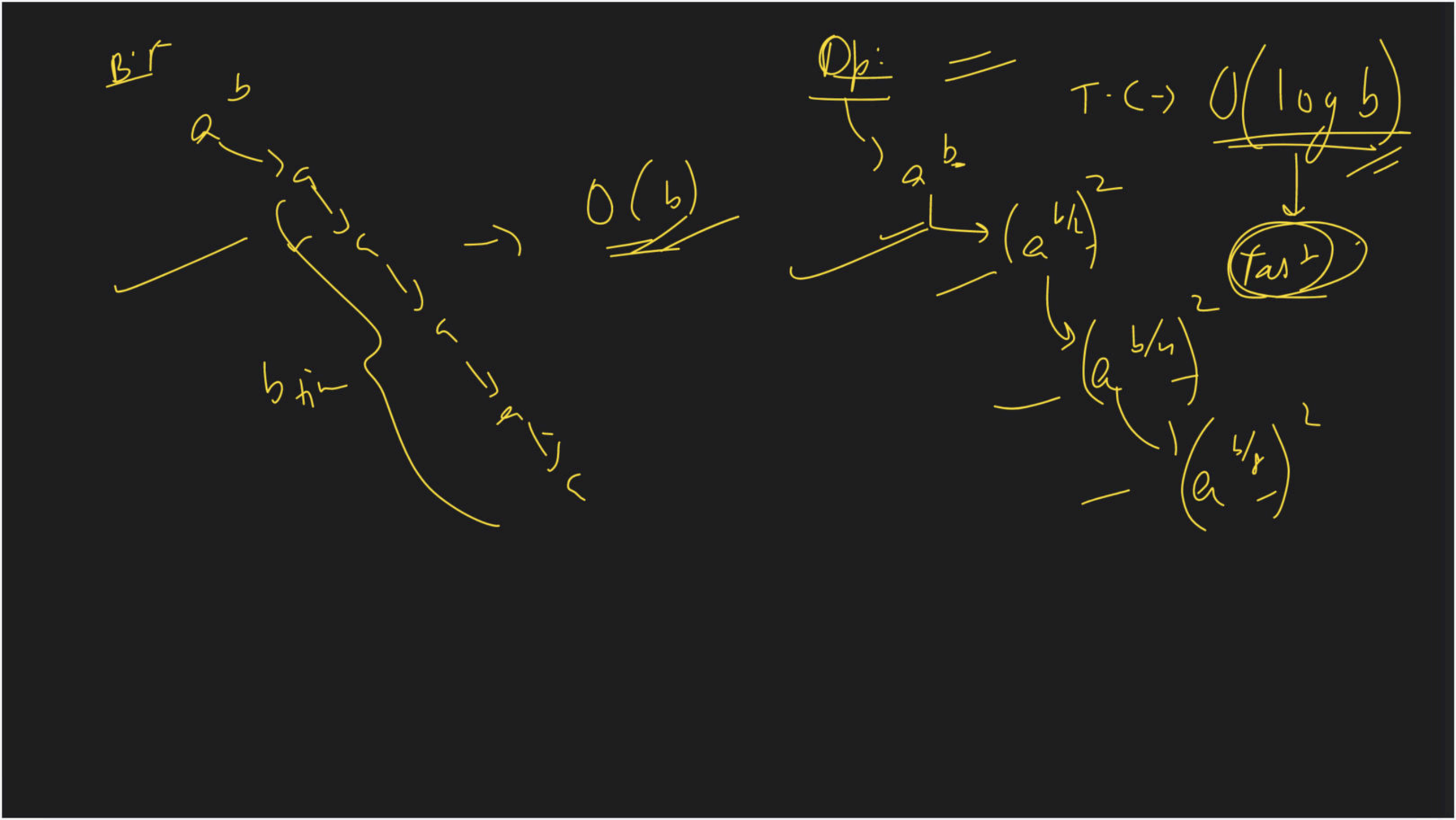
b is odd → $a^b = (a^{b/2})^2 * a$

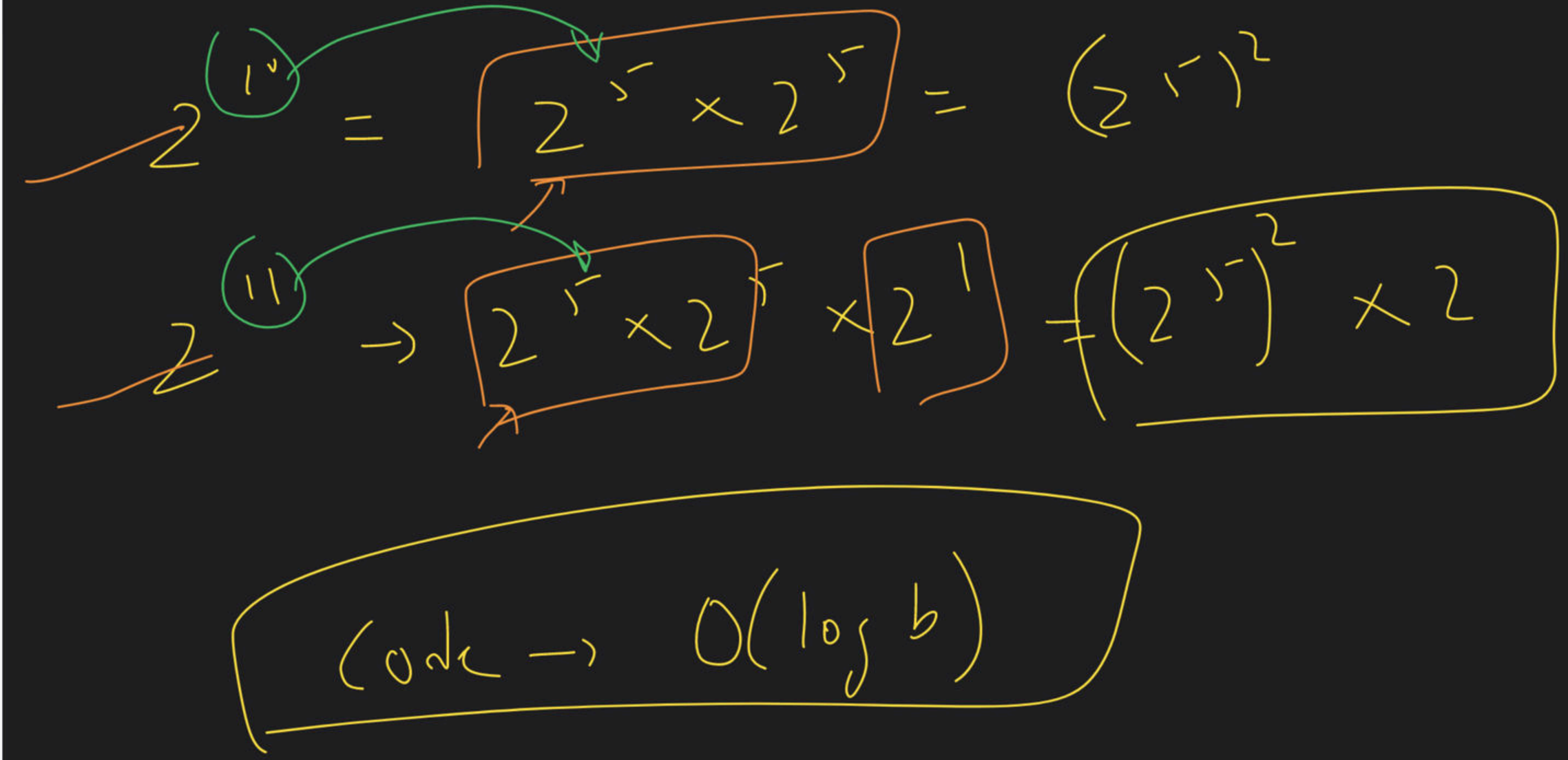
$2^{10} = (2^5)^2$

$2^{11} \rightarrow (2^{5})^2$

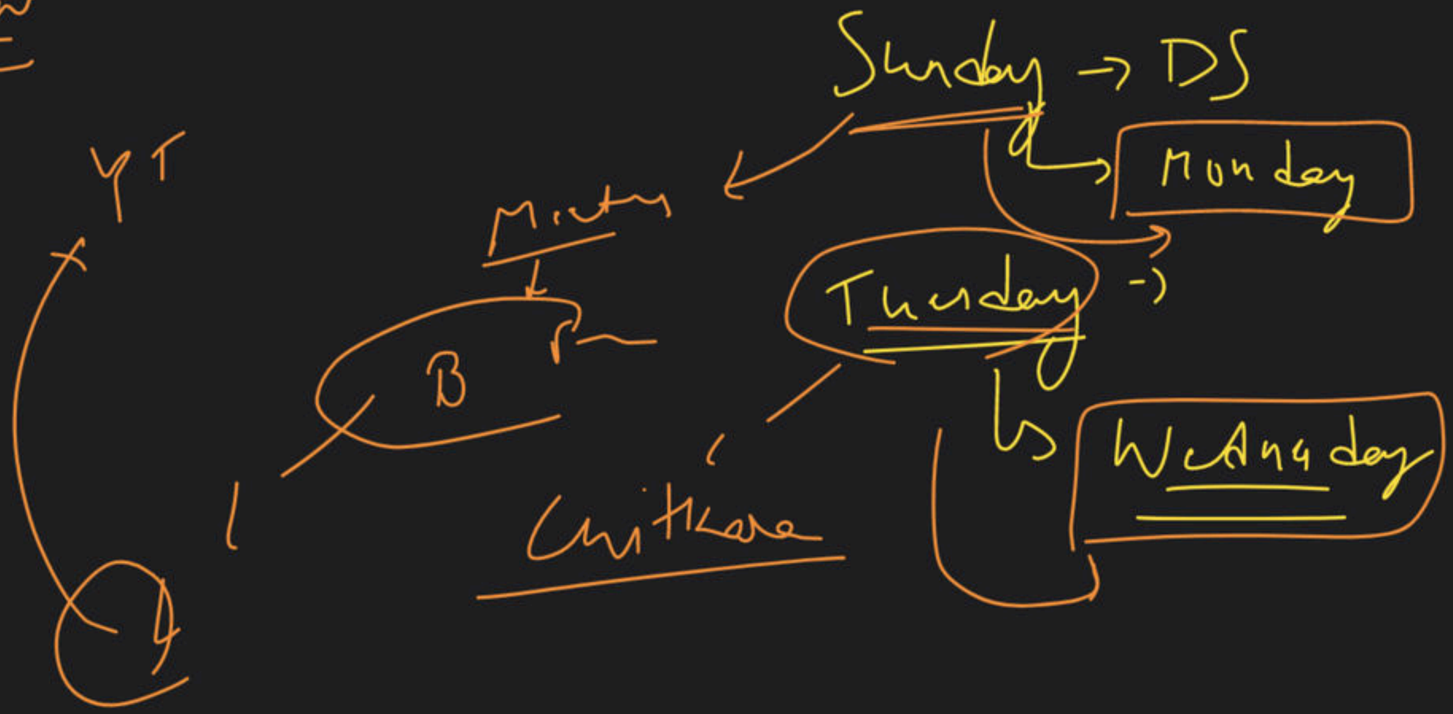
$2^{10} = (2^{5})^2$

$2^{10} \rightarrow (2^5)^2 * a$


Bir
b
O(b)
bir
Dp:
T.C -> O(log b)
fast


$2^{10} = 2^5 \times 2^5 = (2^5)^2$
$2^{11} \rightarrow 2^5 \times 2^5 \times 2^1 = (2^5)^2 \times 2$
Code → O(log b)

Announcement



x^n

```
int res = 1;
while (n > 0)
{
    // odd
    if (n & 1)
        res = res * x;
    x = x * x;
    n = n >> 1  or  n = n/2
}
return res;
```

n → n/2 → n/4 → n/8 → ... 0

n = n/2

>>

n = n >> 1

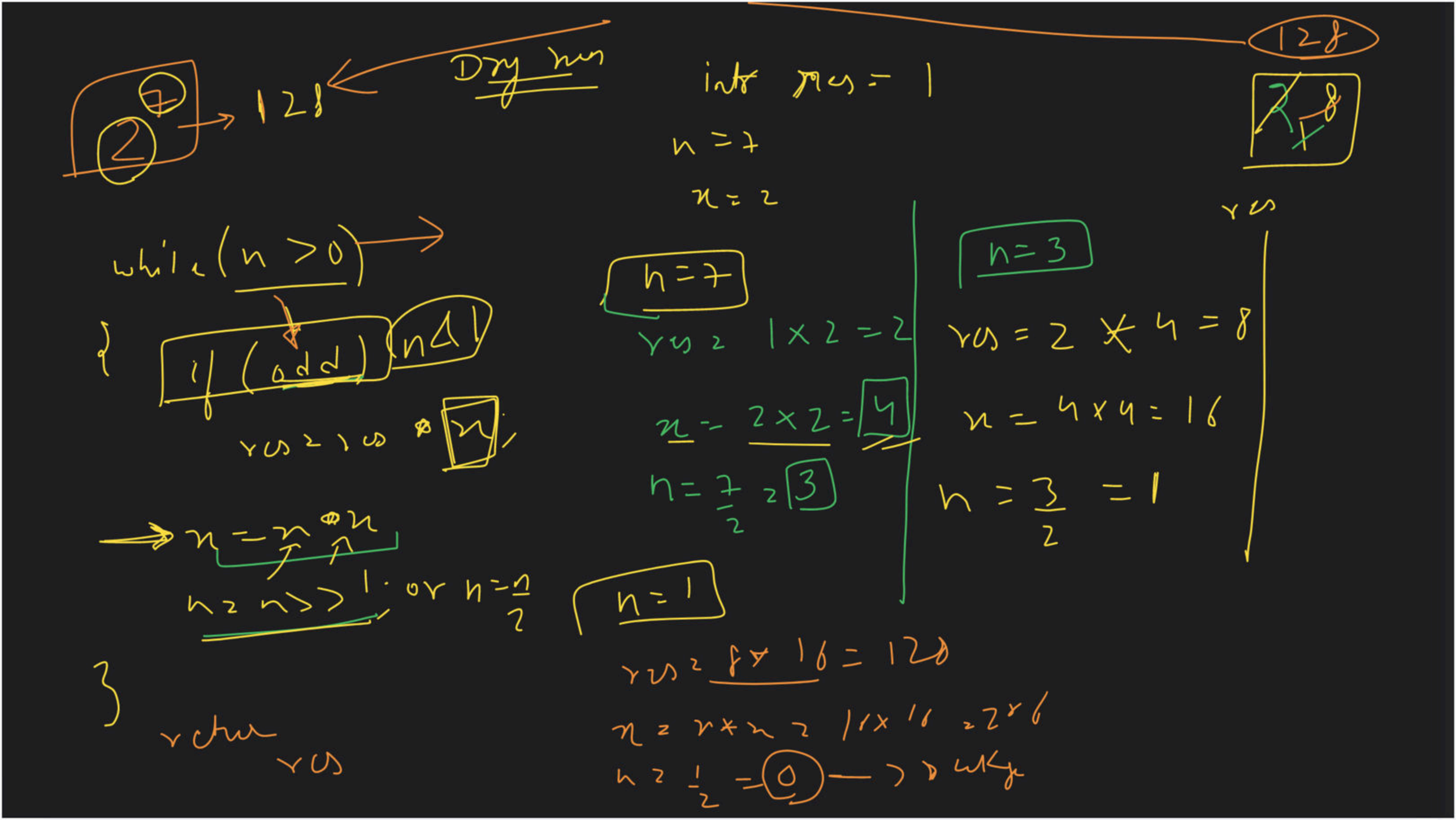
n = n >> 1 → / by 2

OOPS

Low-Level Design

New word

Multiply


Dry run
128
int res = 1
n = 7
n = 2
while (n > 0)
{
if (odd) (n&1)
res = 1 * n;
n = n * n
n = n >> 1 or n = n/2
}
return res
n = 7
res = 1 x 2 = 2
n = 2 x 2 = 4
n = 7/2 = 3
n = 3
res = 2 * 4 = 8
n = 4 x 4 = 16
n = 3/2 = 1
n = 1
res = 8 * 16 = 128
n = n * n = 1 x 16 = 256
n = 1/2 = 0 ---> while
res
R
8

a^b

b is even → $\left(a^{b/2}\right)^2$

b is odd → $\left(a^{b/2}\right)^2 \times a$

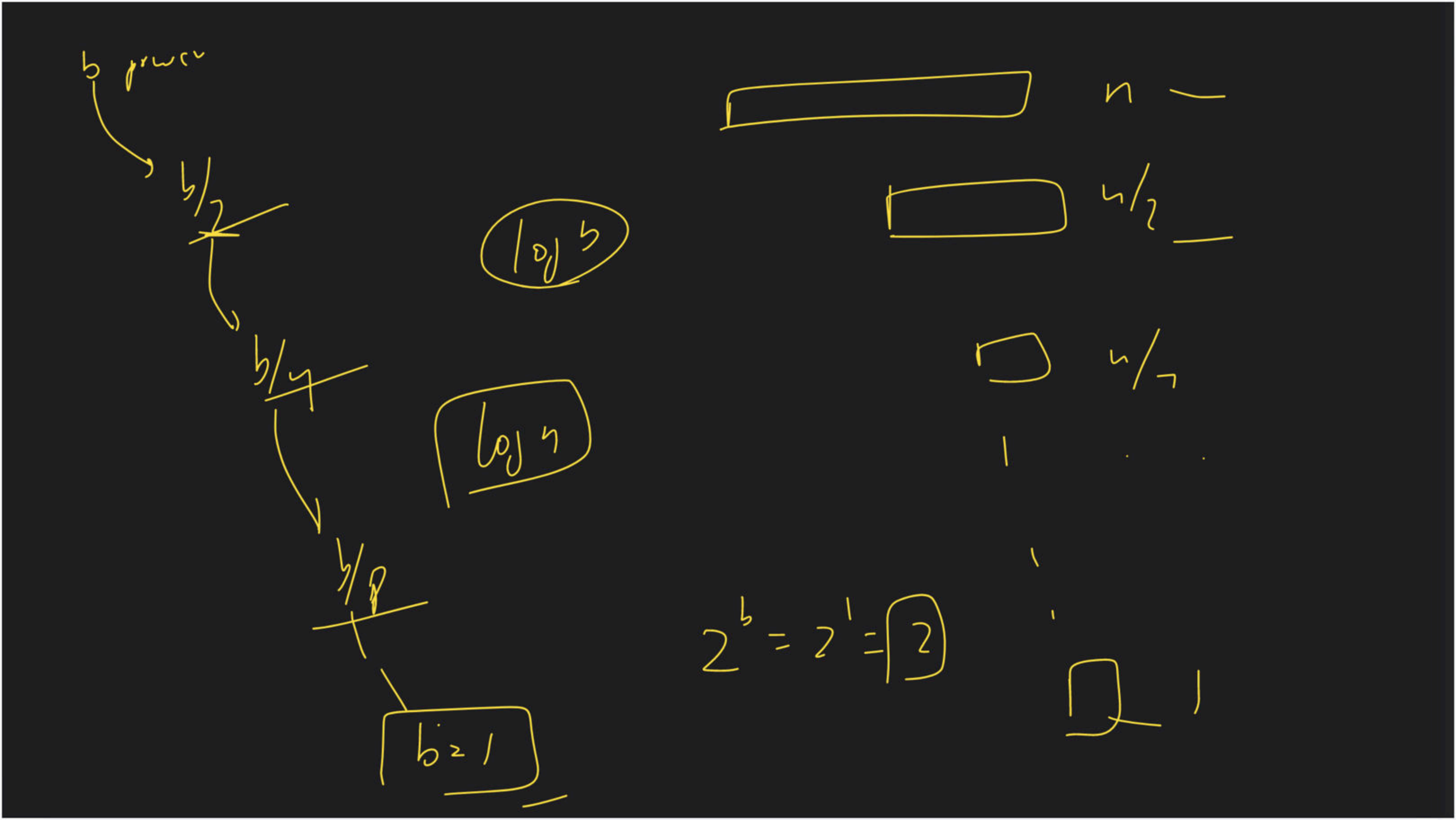
$2^{10} \rightarrow 2^5 \times 2^5$

$= (2^5)^2$

$= \left(2^{10/2}\right)^2$

$2^{11} \rightarrow 2^5 \times 2^5 \times 2$

$= (2^5)^2 \times 2$


b proces
b/2
b/4
b/8
b=1
log b
log n
n
n/2
n/4
1
$2^b = 2^1 = 2$
1

Even -> $2^{b/2} \times 2^{b/2}$

Odd -> $2^{b/2} \times 2^{b/2} \times 2$

DRY -> Fast

RUN

Simple

if cond

→ Prime no

→ Sieve of Eratosthenes

→ GCD

→ GCD / LCM

→ Euclid Algo

→ Modulo Arith

→ fast exp

Topper

H/w

→ PigeonHole principle

Tom's research favourite

Catelon number

↓ formula

Incl - Excl principle

10 min

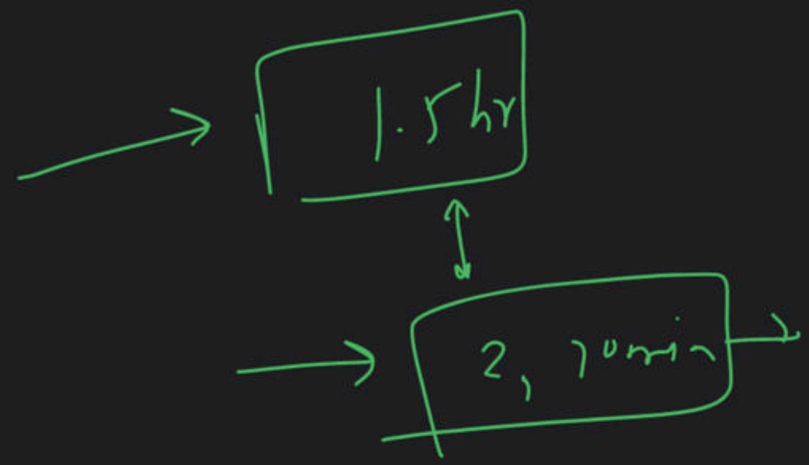
→

tree →

BST

probability

↳ Catelan no



ignore

Breakfast

Session

Monday

Delhi