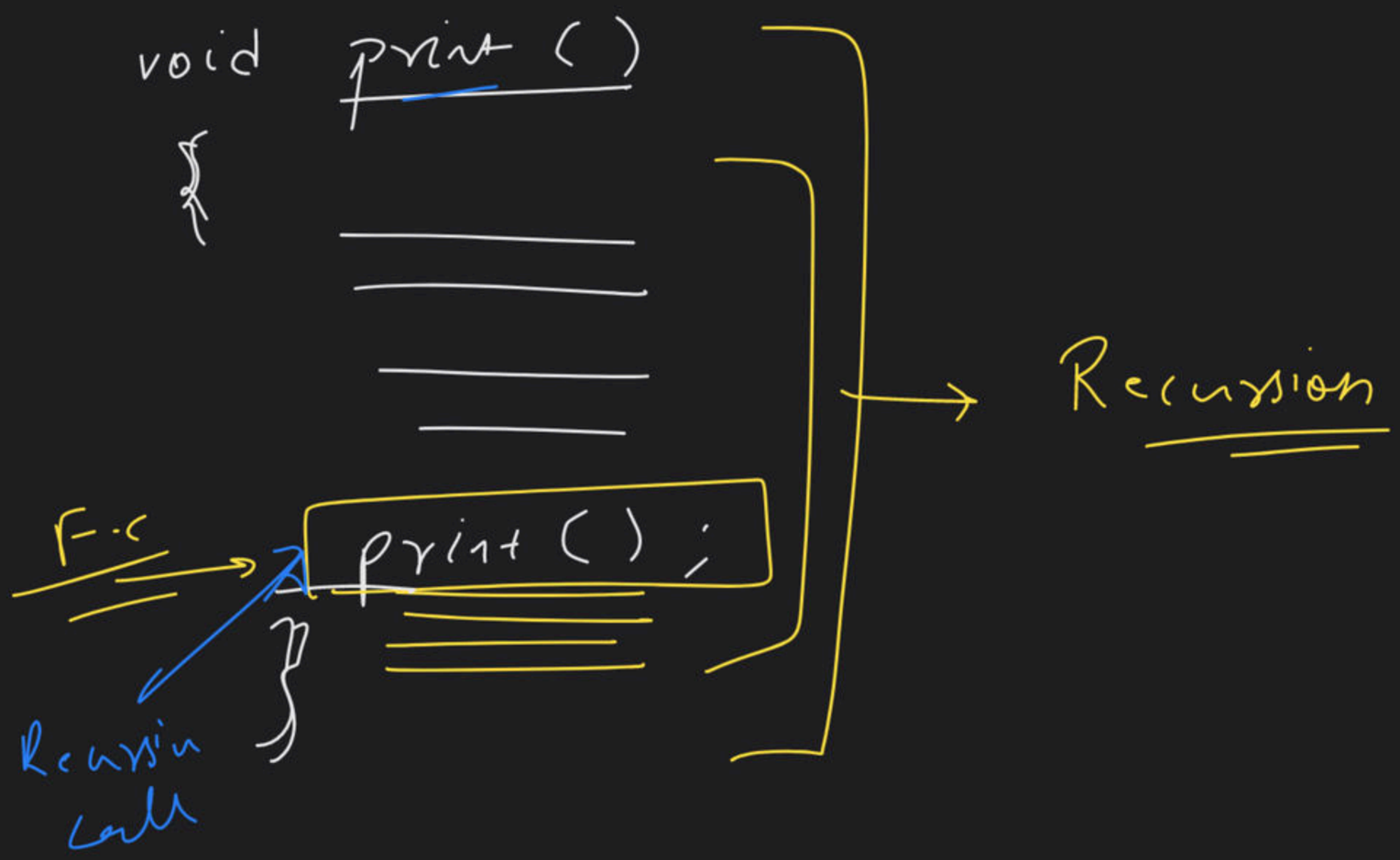
Hi Everyone

Recursion - I

Foundation Course on Data Structures & Algorithm - Part I

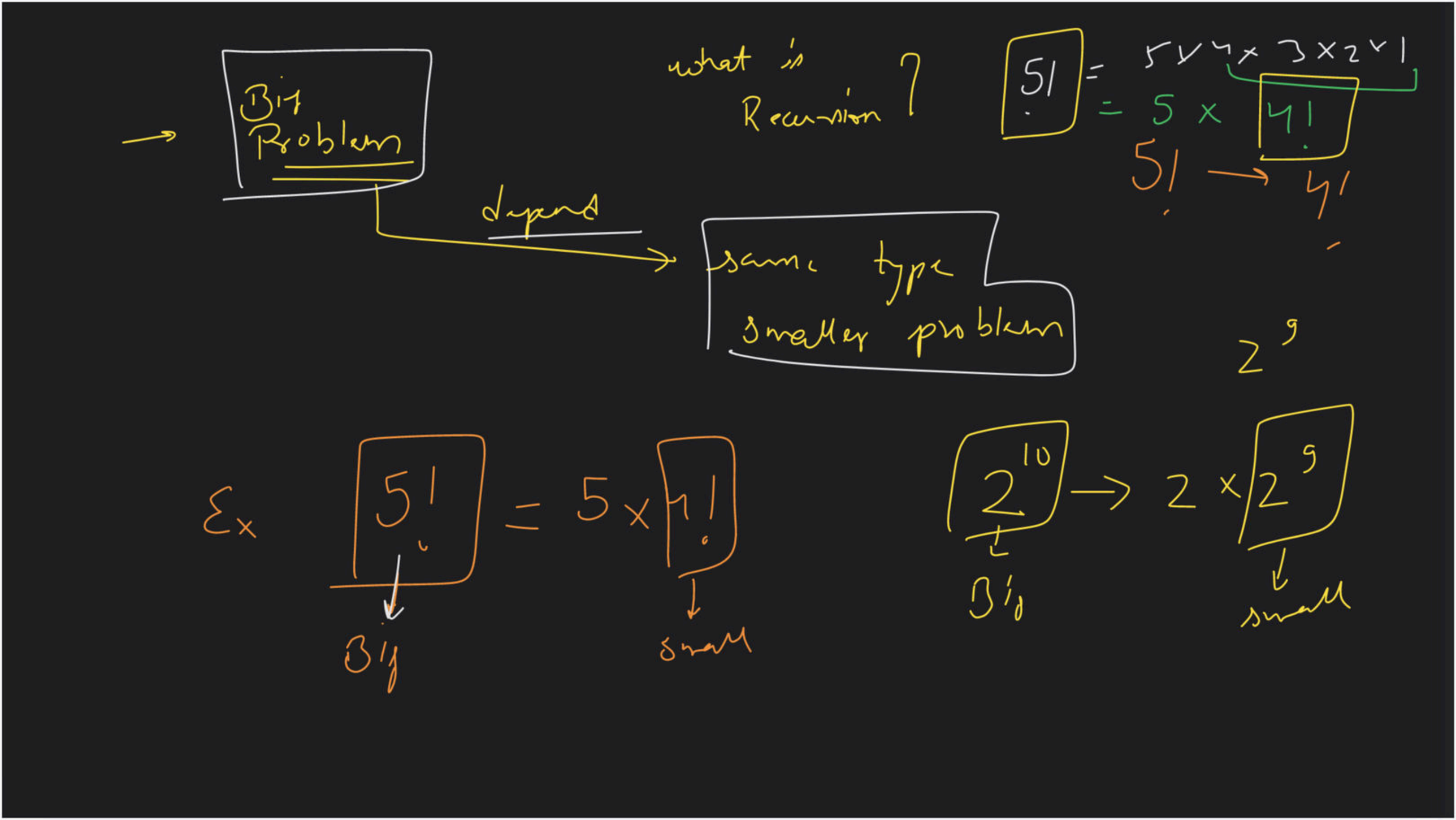
Love Babbar • Lesson 19 • May 25, 2022

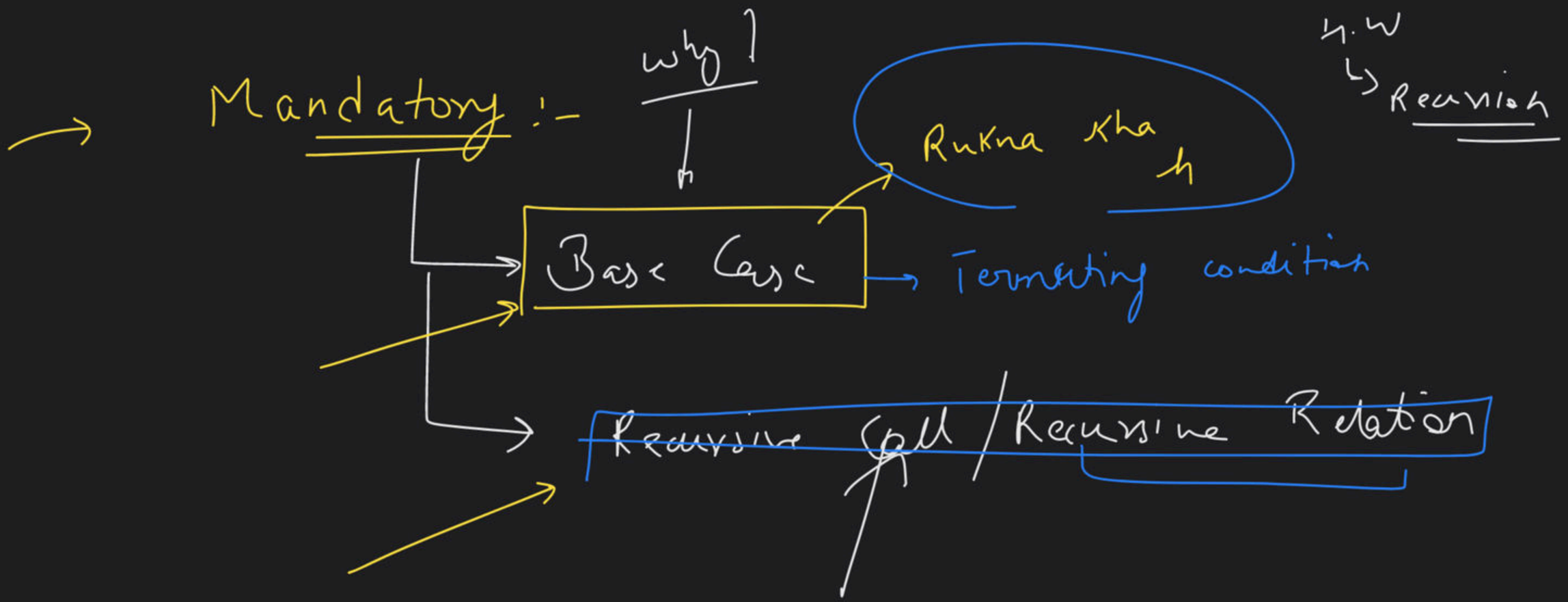
Recursion :- [when a function calls itself]



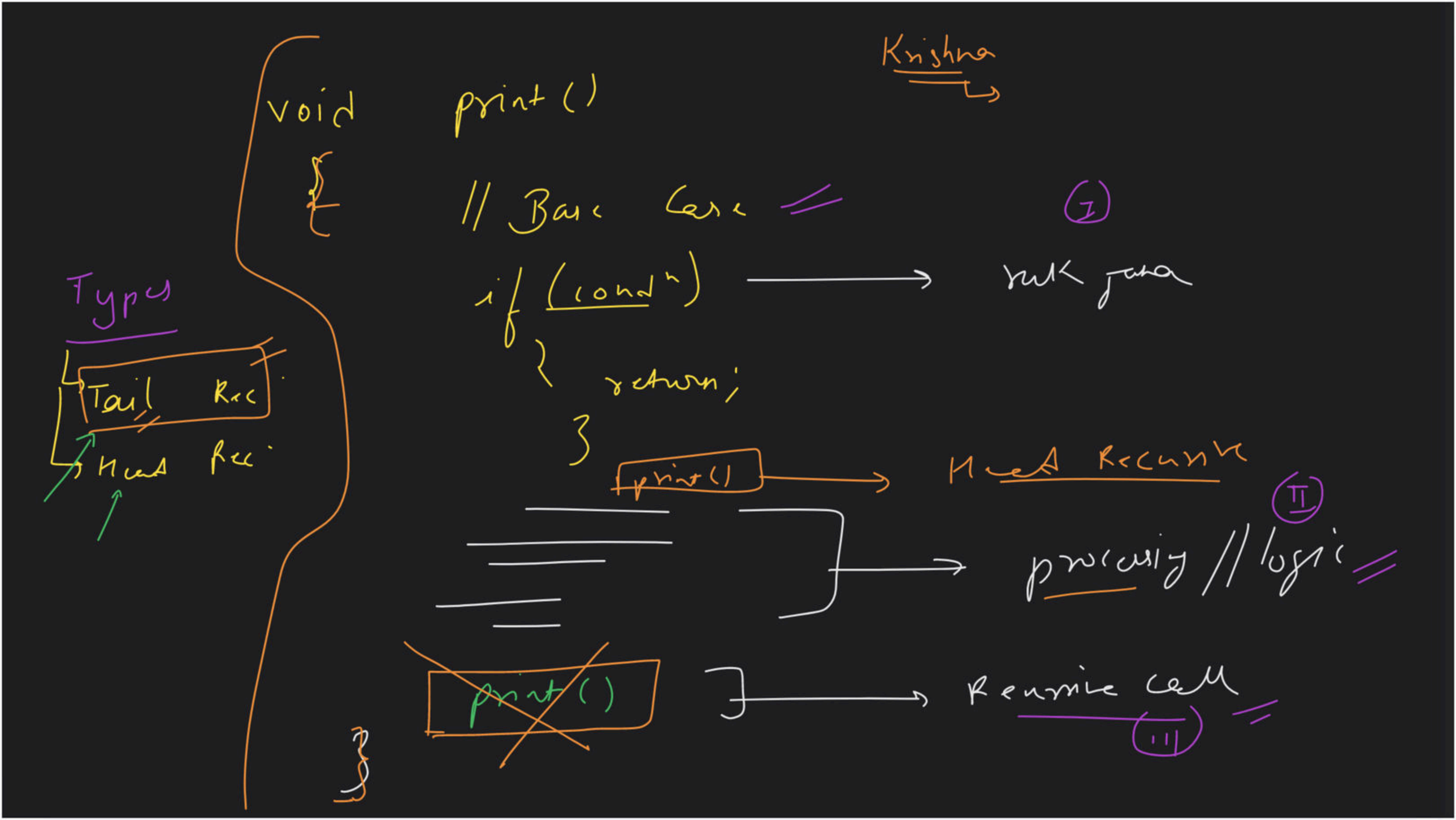
Book

Recursion


Big Problem
depends
same type
smaller problem
what is Recursion ?
5!
= 5 × 4 × 3 × 2 × 1
= 5 × 4!
5!
4!
Ex
5!
= 5 × 4!
Big
small
2^10
2 × 2^9
Big
small
2^9



H.W
↳ Recursion


Krishna
void print()
{
// Base Case
if (condn)
return;
}
Tail Rec
Head Rec
print()
print()
}
(I)
Work done
Head Recursion
(II)
processing // logic
Recursive call
(III)

→ Example :- Factorial

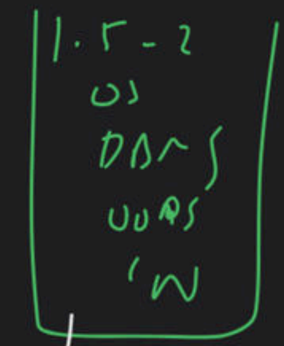
$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 1$

$(n-1)!$

$n! = n \times (n-1)!$ ← Recurrence relation:

Big

Smaller



n = 5

$n! = n \times (n-1)!$

$5! = 5 \times 4!$

$4! = 4 \times 3!$

$3! = 3 \times 2!$

$2! = 2 \times 1!$

$1! = 1 \times 0!$

n > 0

n > 0

Rekurencja

1! → return 1;

n == 0 or n == 1

rekurencja

return 1;

n == -1

return 1;

5! -> 5 x 4!

n = n * (n-1)!

1 case might solve karna h, baaki recursion sambhal lega

Recursive rule

5! = 5 x 4!

n! → n x (n-1)!

chhoti problem

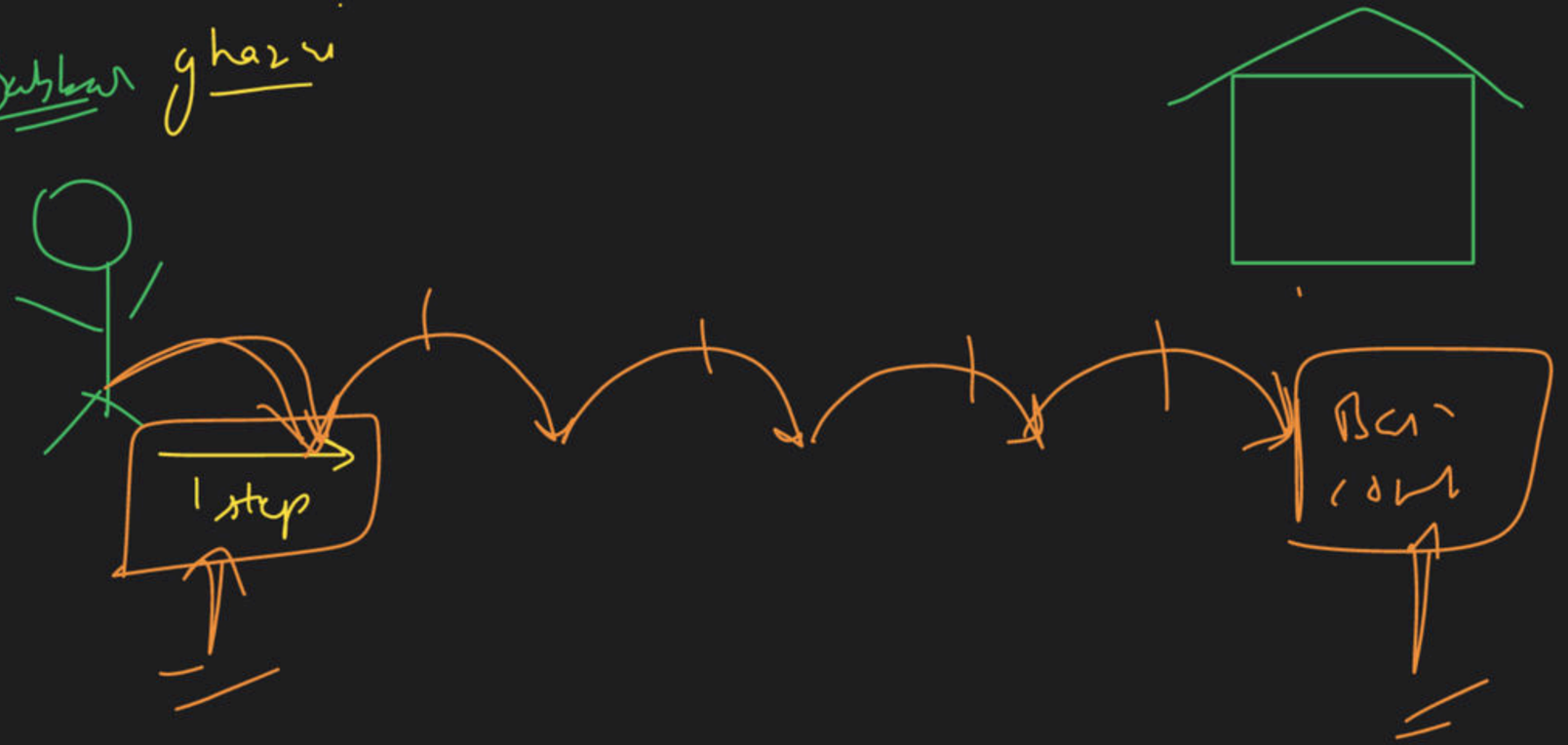
n = 1

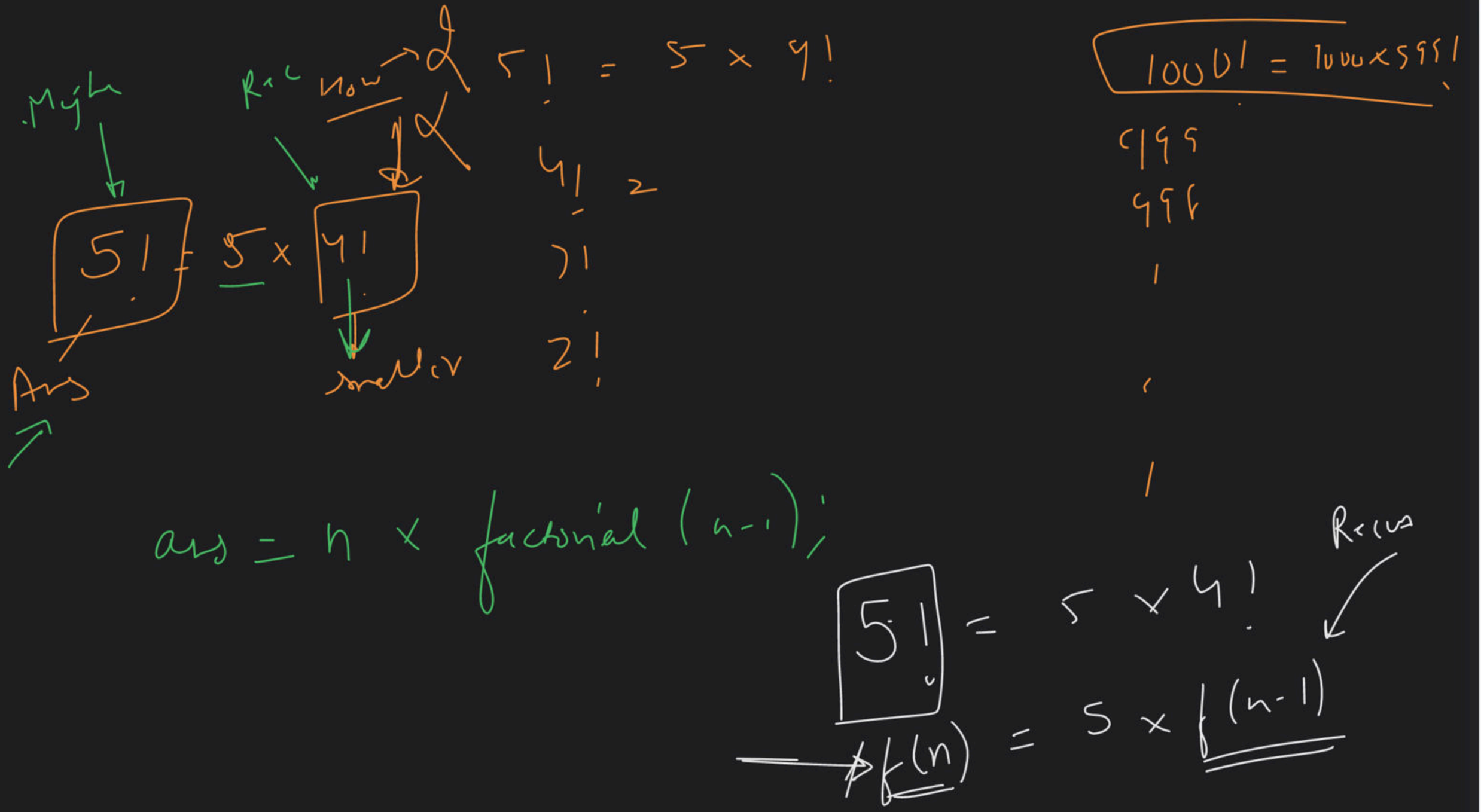
arrays → 5:30

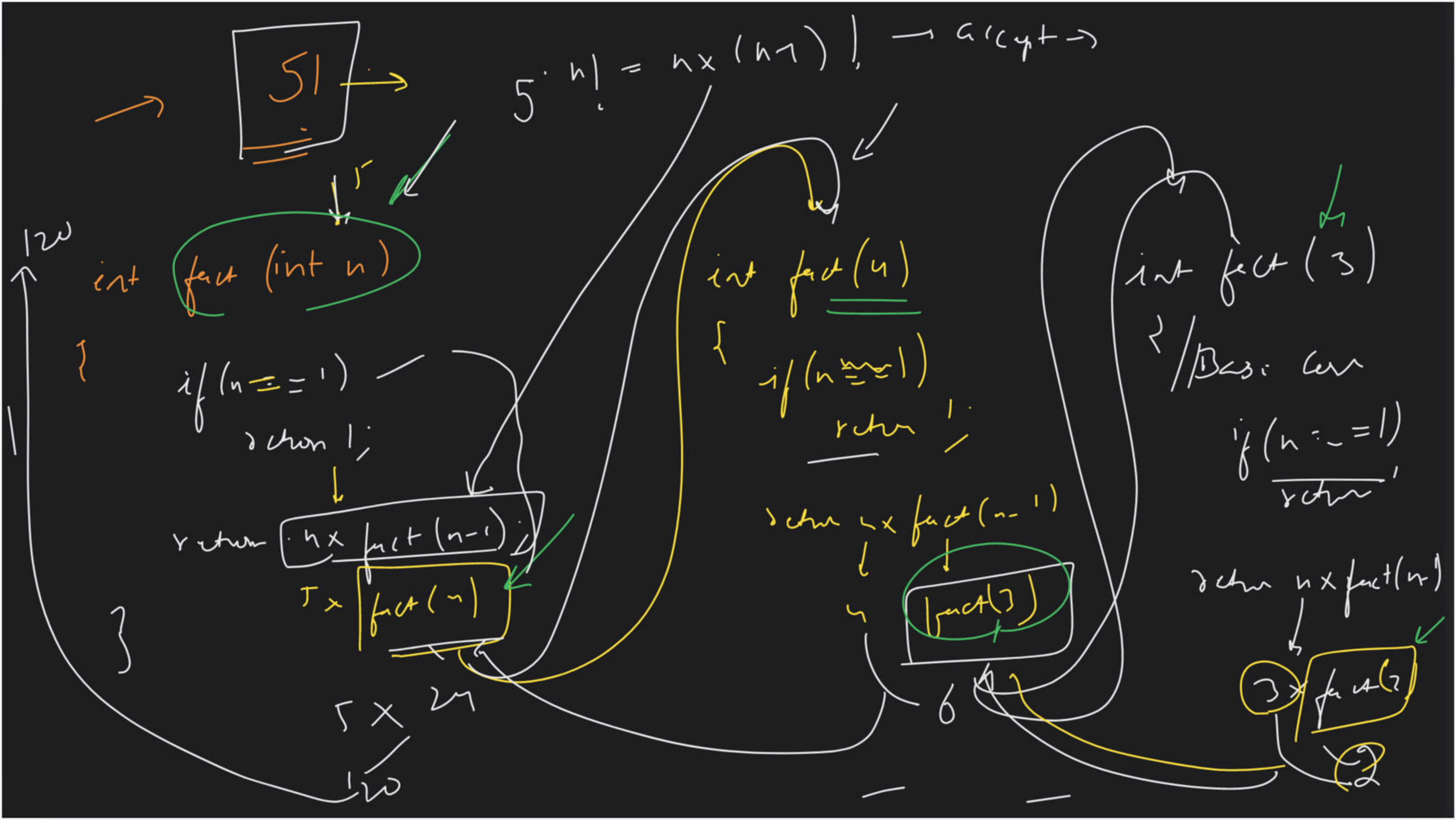
1 step

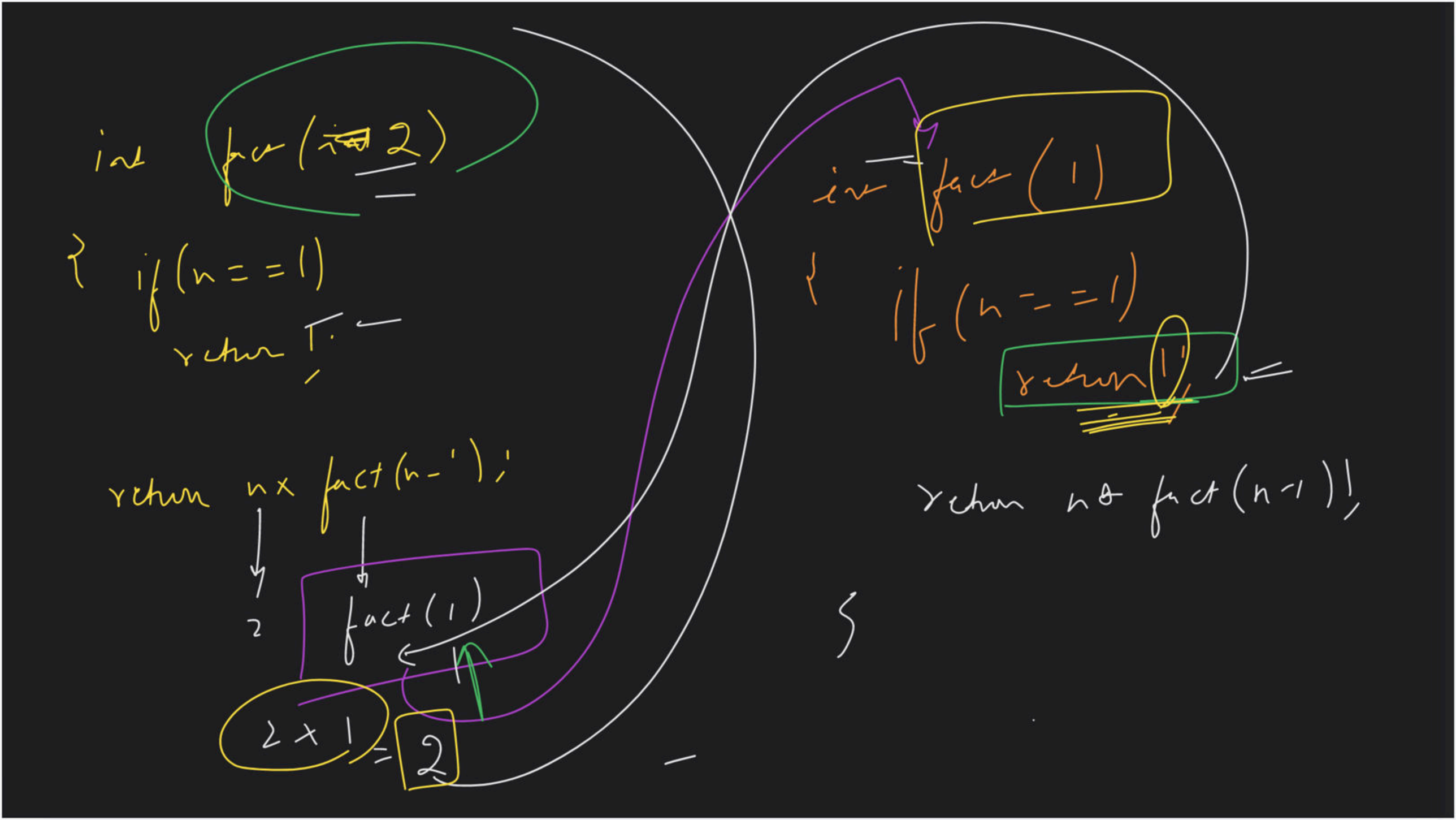
Needs fuel

Babbar ghazi

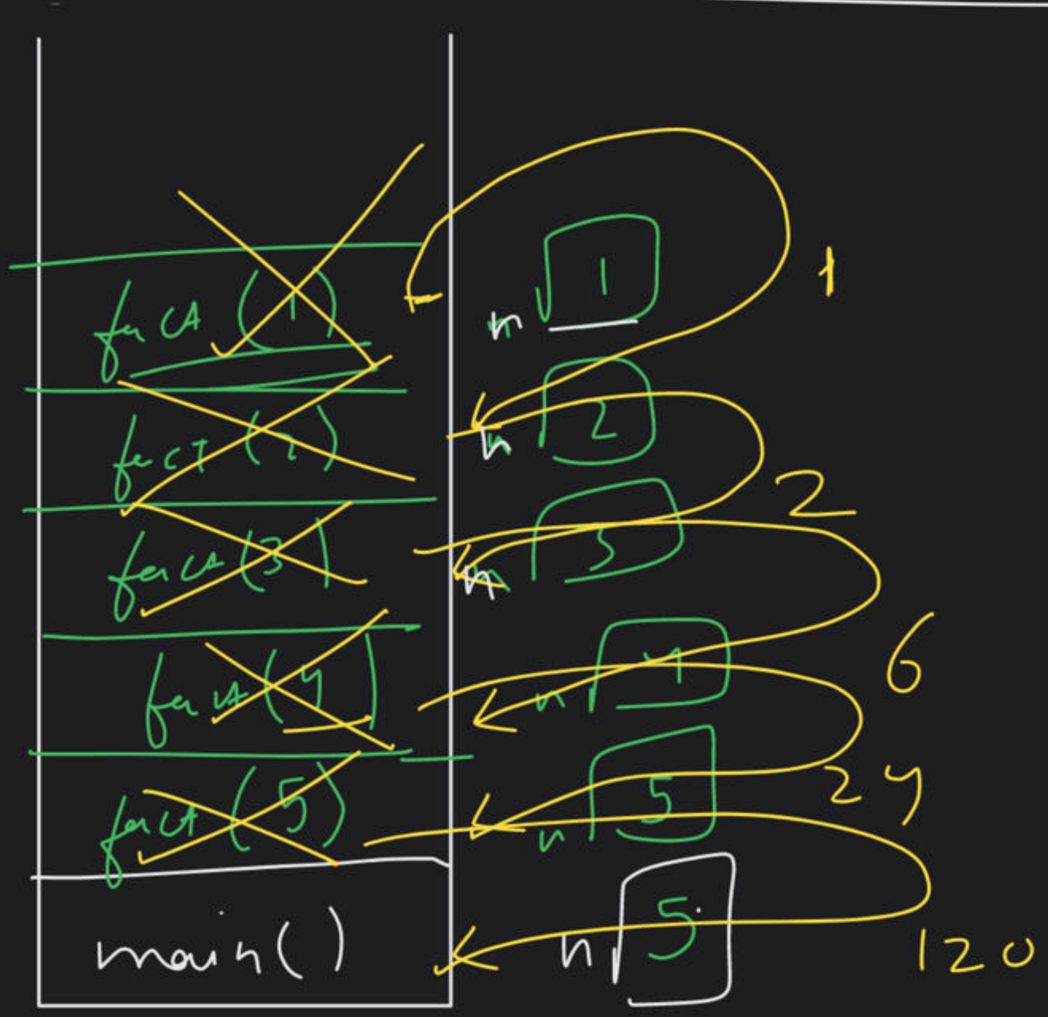



Main
Rec Now
5! = 5 x 4!
4! = 4 x 3!
3! = 3 x 2!
2!
5! = 5 x 4!
Ans
smaller
100! = 100 x 99!
99!
1
ans = n x factorial(n-1);
5! = 5 x 4!
Recur
f(n) = 5 x f(n-1)


5!
5! = n x (n-1)!
int fact (int n)
{
if (n == 1)
return 1;
return n x fact(n-1);
int fact(4)
int fact(3)
int fact(2)
int fact(1)
5 x 24
4 x 6
3 x 2
2 x 1
120
24
6
2
1


int fact(int n)
{ if(n==1)
return 1;
return n * fact(n-1);
2
fact(1)
1
2 * 1 = 2
int fact(1)
{ if(n==1)
return 1;
return n * fact(n-1);
}

Recursive Call Stack



α

$O(1)$

n → copy

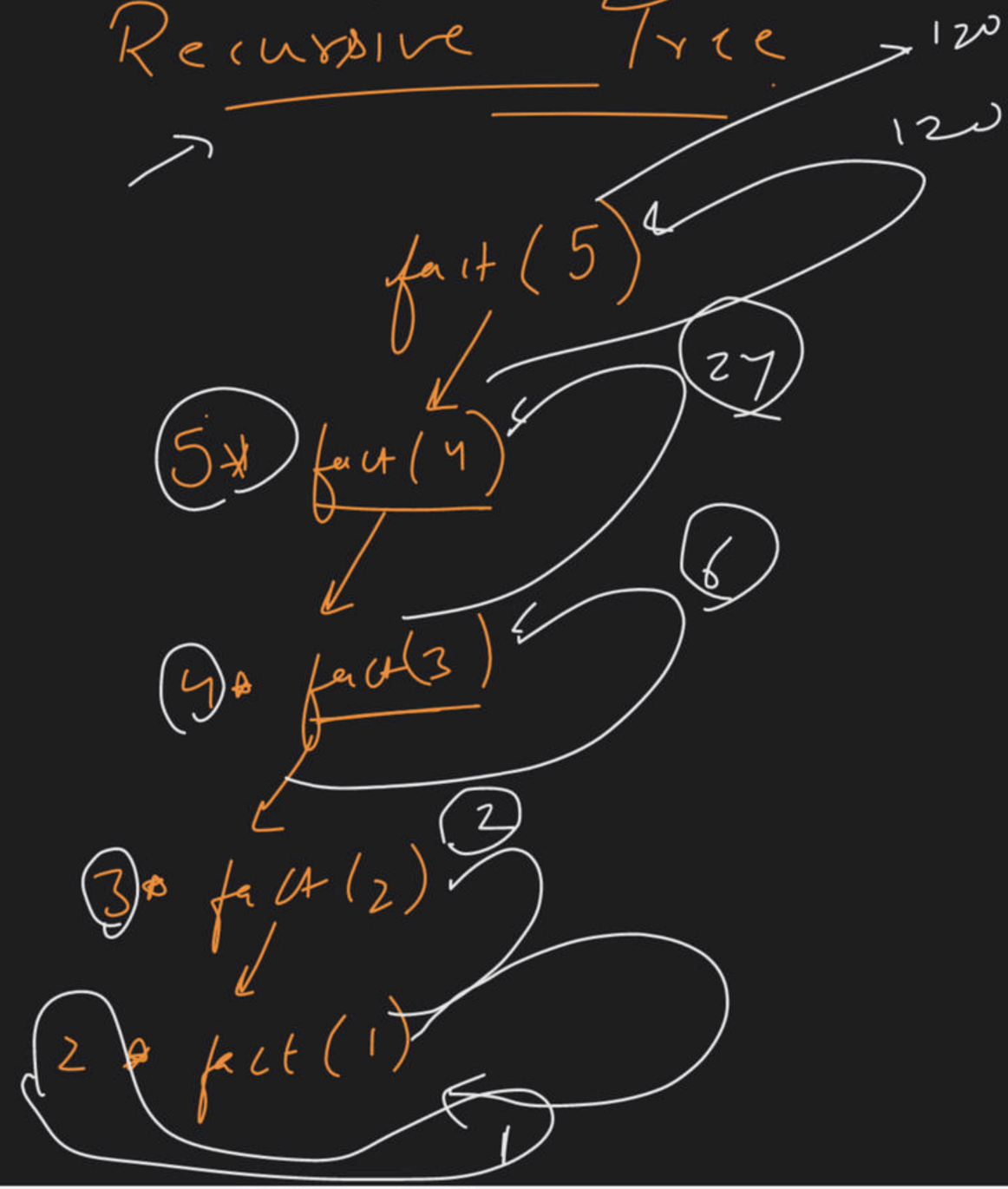
m → recursion calls

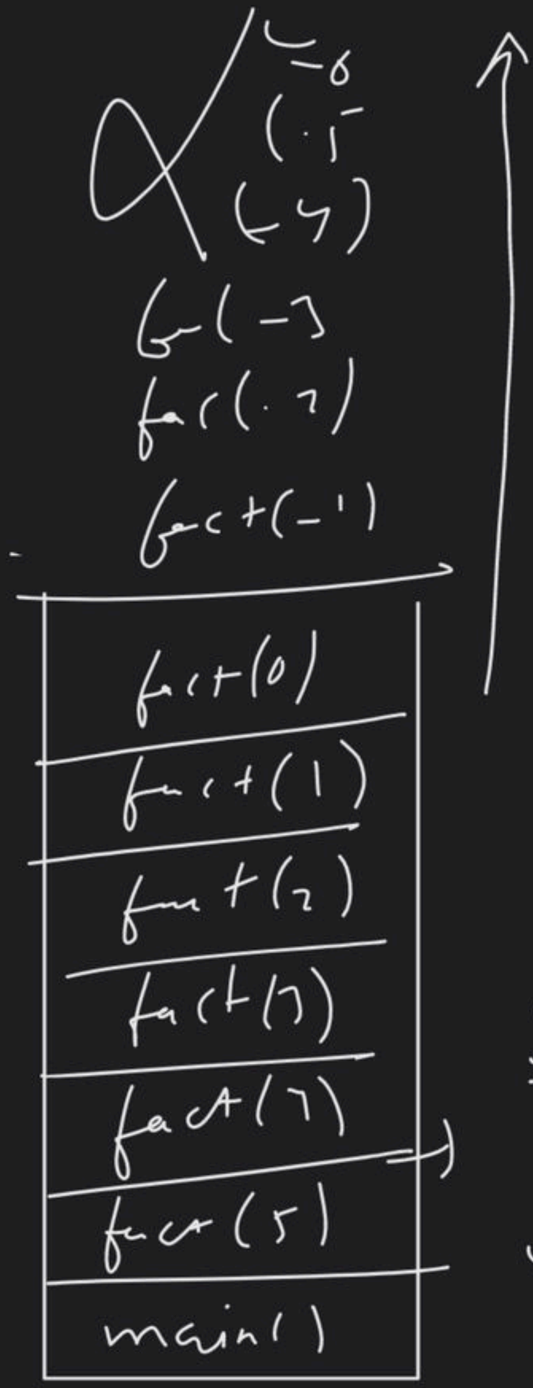
m hard

$m \times O(1)$

$= O(m)$

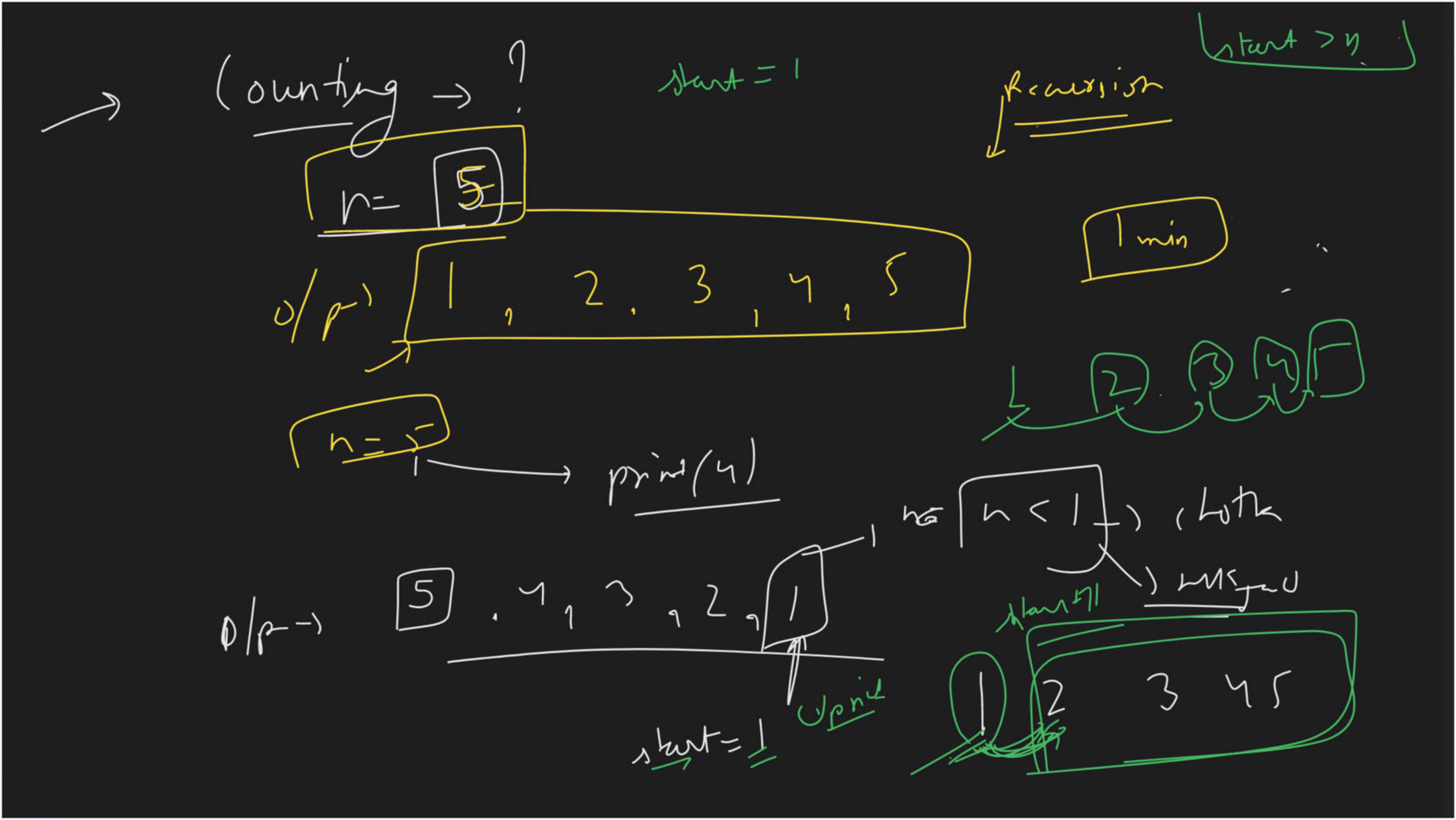
Recursive Trace

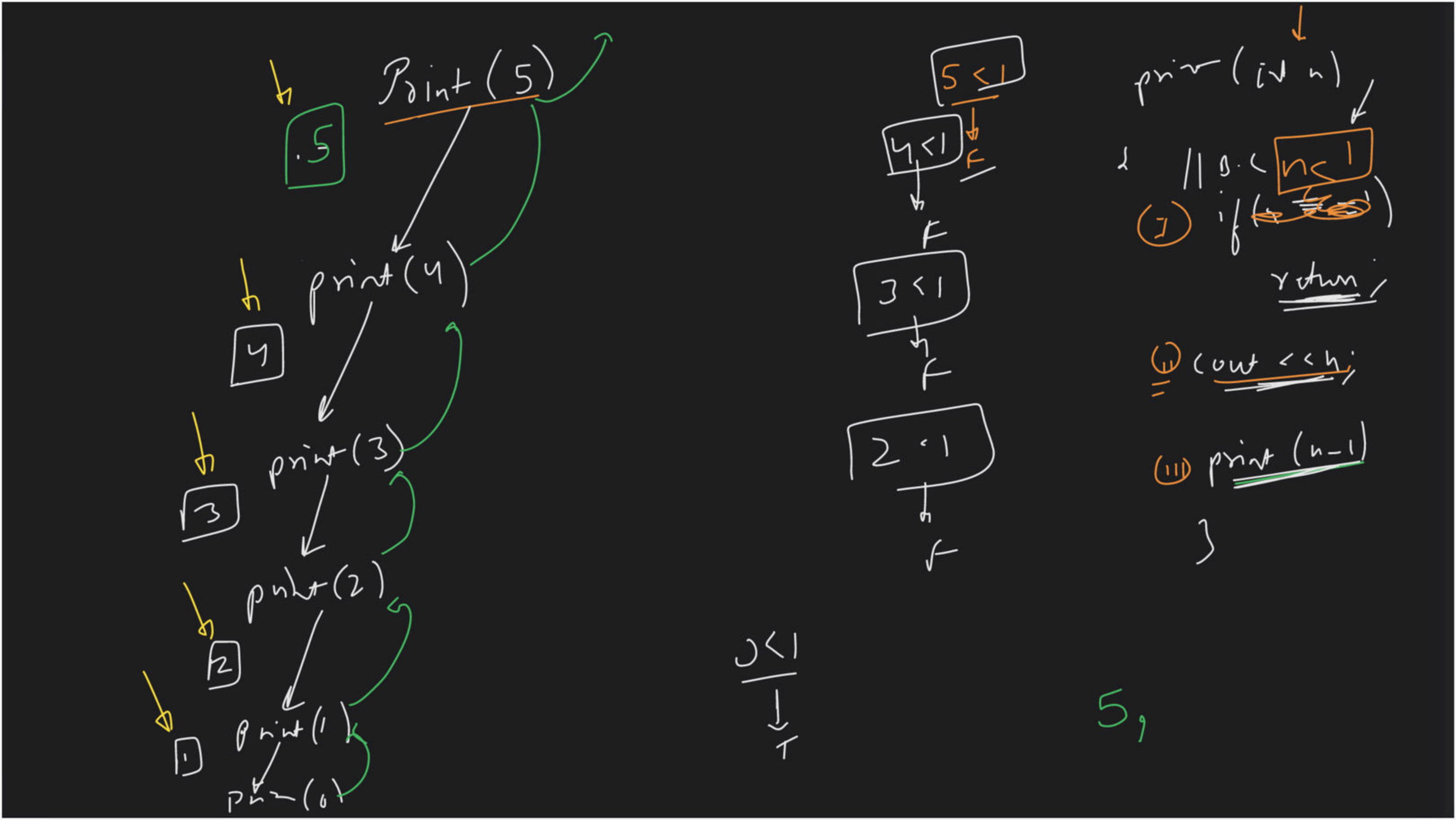


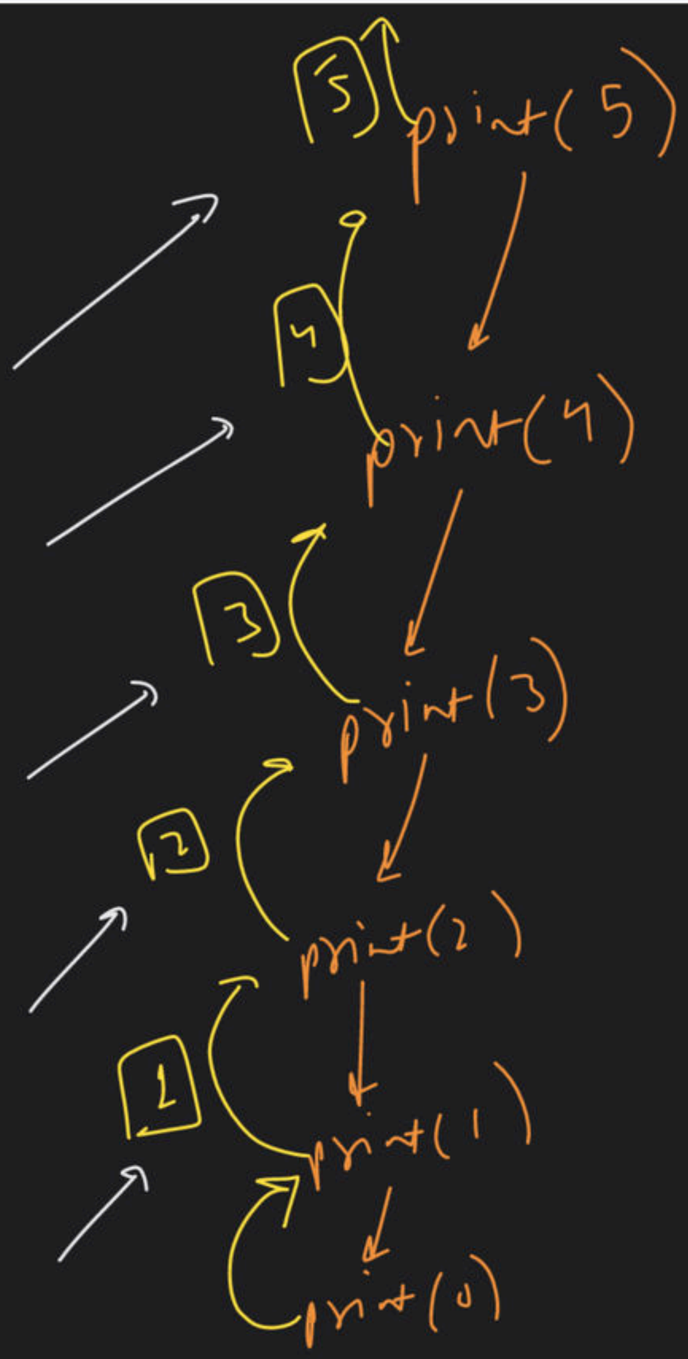


Stack

5 · Overflow


Counting → ?
start = 1
Recursion
start > n
n = 5
o/p → 1, 2, 3, 4, 5
1 min
n = 5
print(4)
o/p → 5, 4, 3, 2, 1
n < 1
start = 1
start = 1
print
1 2 3 4 5


print(5)
5
print(4)
4
print(3)
3
print(2)
2
print(1)
1
print(0)
5<1
4<1
F
F
3<1
F
2<1
F
0<1
F
print(int n)
{
if(n<1)
return;
cout<<n;
print(n-1)
}
5,

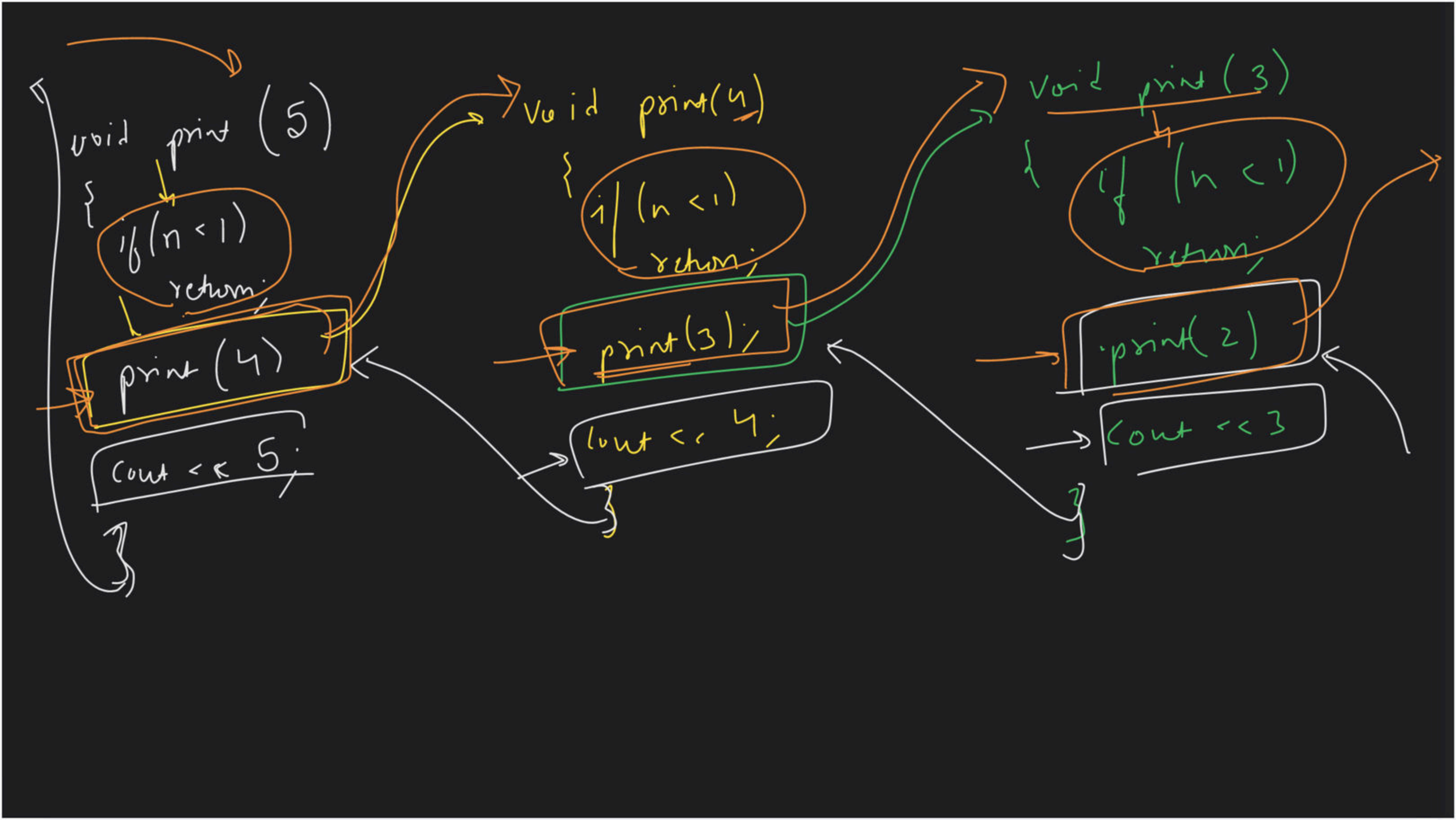


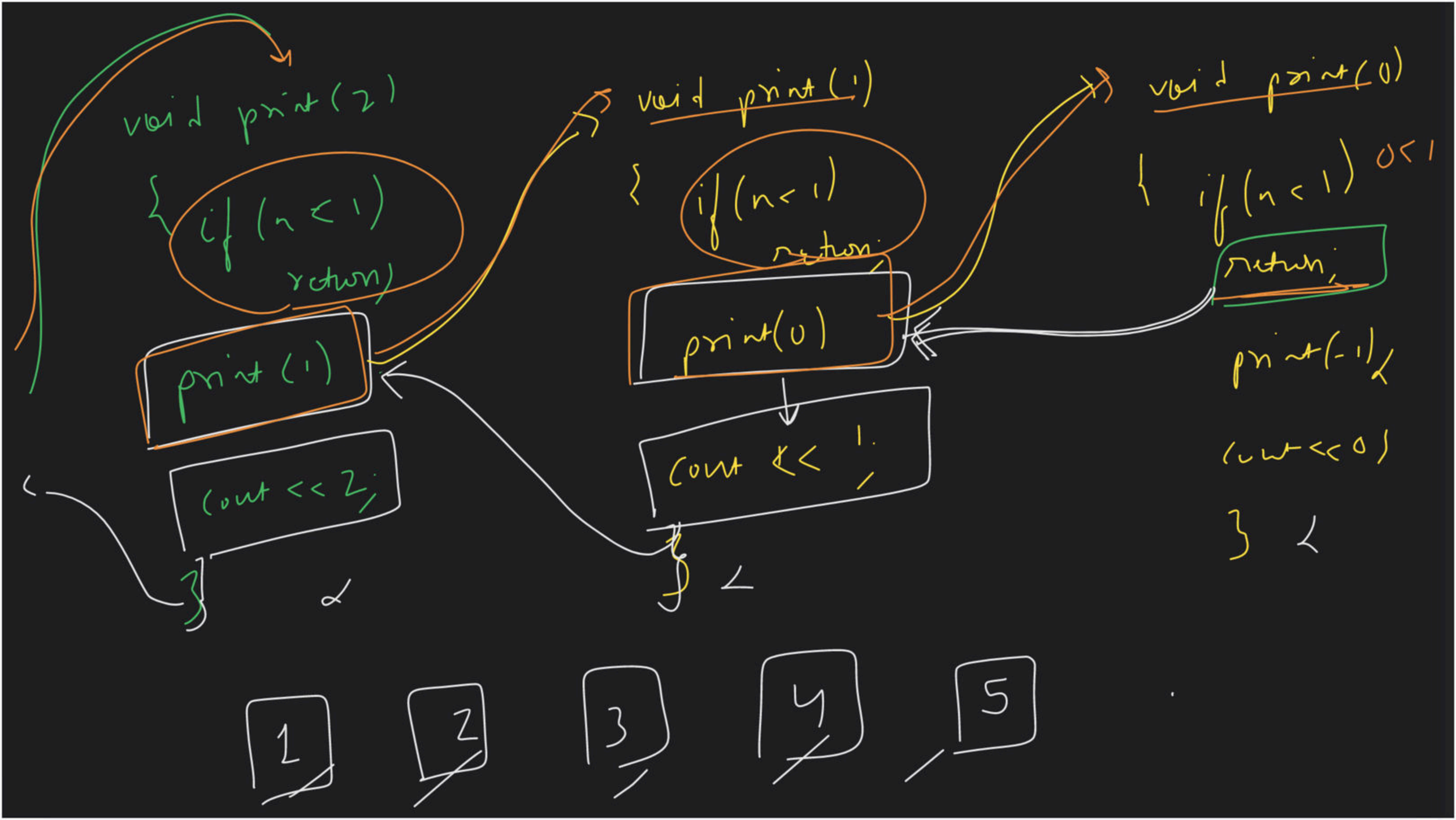
5<1
F
4<1
F
3<1
F
2<1
F
1<1

0<1
T

```
int print(int n)
{
(2) if(n<1)
      return;
(1) print(n-1);
(iii) cout << n;
}
```

Head / Tail


void print (5)
{
if(n<1)
return;
print (4)
Cout << 5;
Void print(4)
{
if (n<1)
return;
print(3);
Cout << 4;
Void print (3)
{
if (n<1)
return;
print(2)
Cout << 3


void print(2)
{
if (n < 1)
return;
print(1)
cout << 2;
}
void print(1)
{
if (n < 1)
return;
print(0)
cout << 1;
}
void print(0)
{
if (n < 1)
return;
print(-1);
cout << 0;
}
1
2
3
4
5

→ code -> fast exponentiation

↳ complex

2^n

n is odd -> $2^{n/2} \Rightarrow 2^{n/2} \times 2$

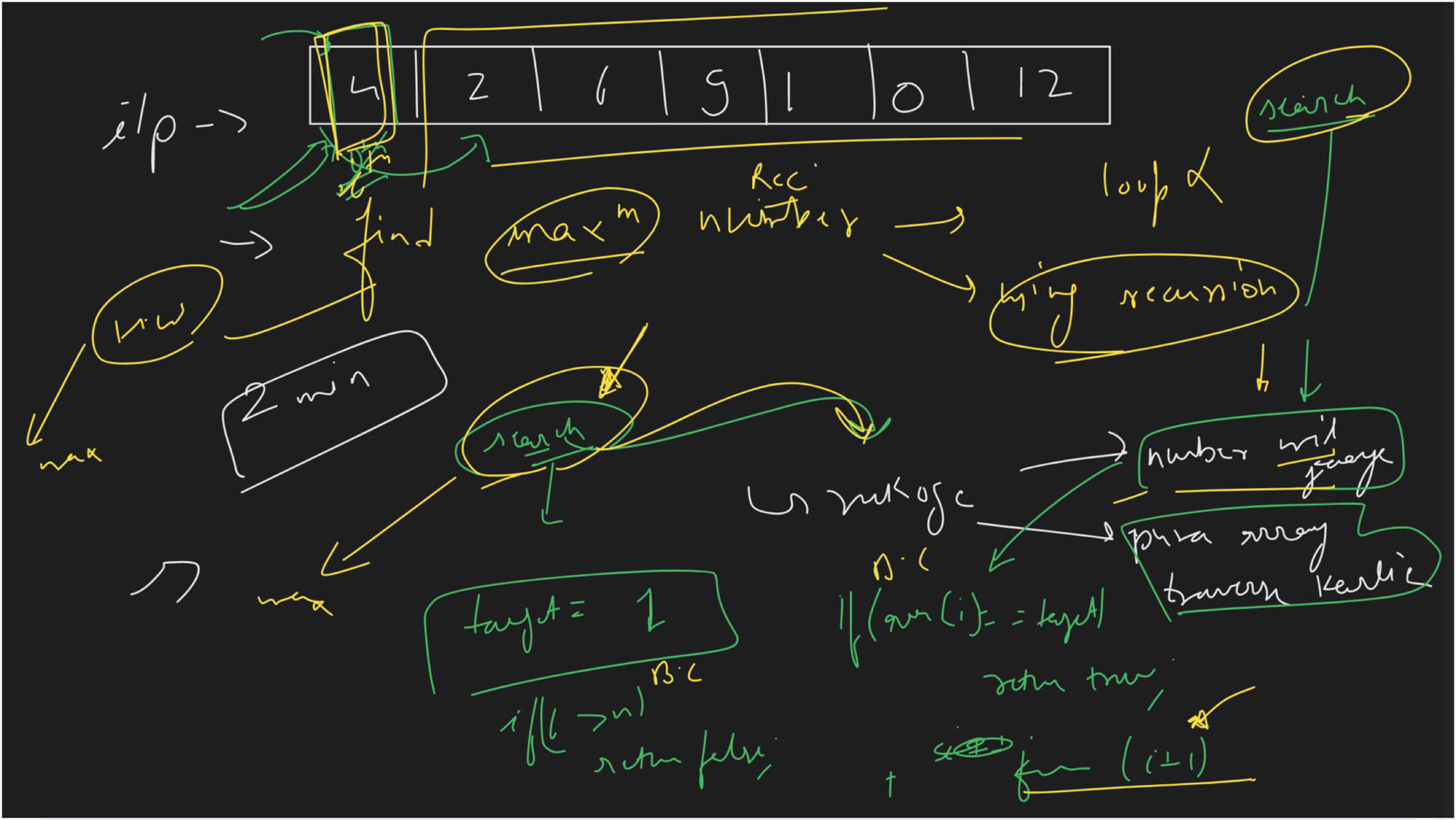
n is even -> $2^{n/2} \times 2^{n/2}$

$1 < 1$

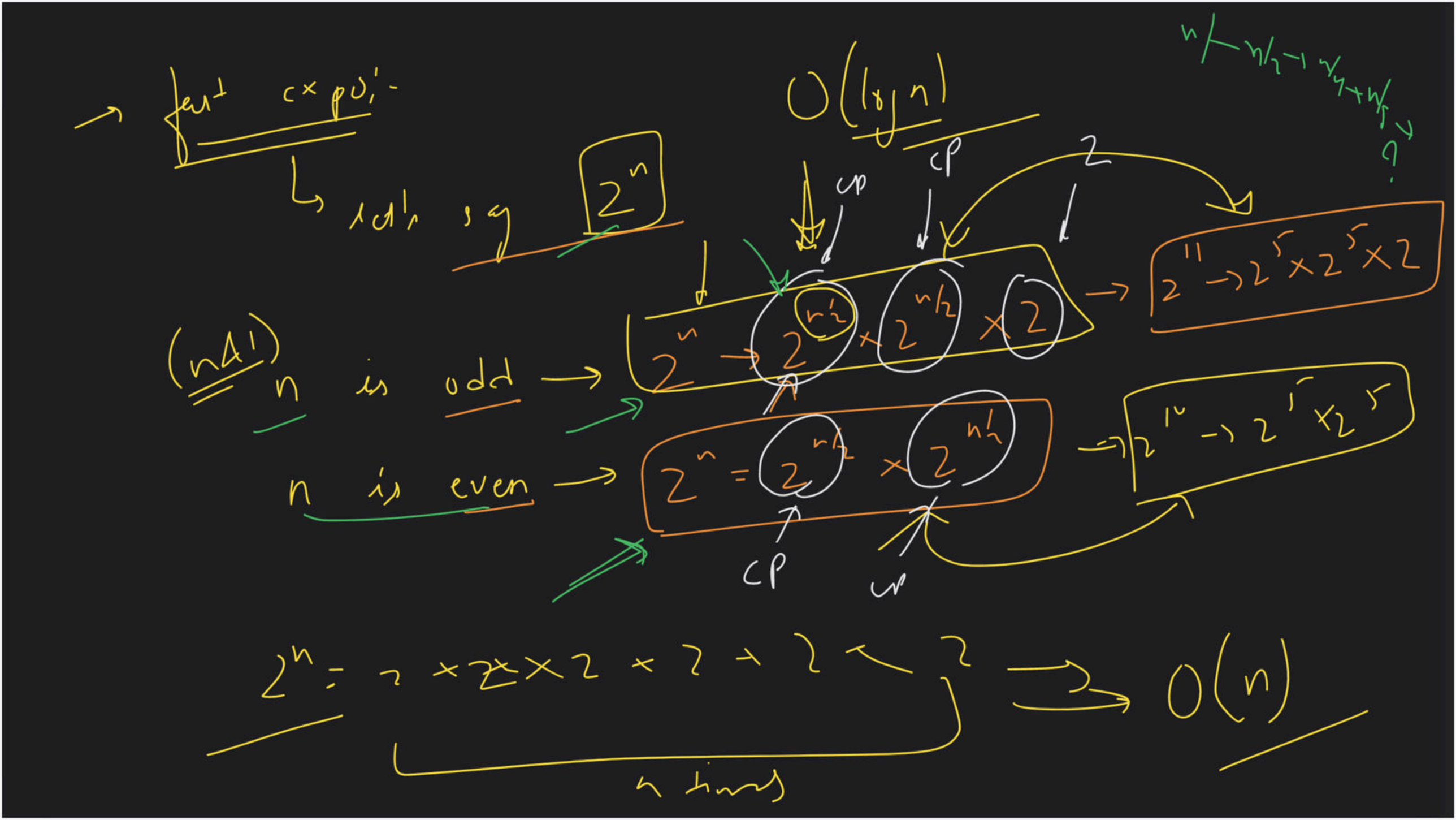
↳ false

$0 < 1$

↳ true



i/p ->
4 | 2 | 6 | 9 | 1 | 0 | 12
find
max
2 min
search
loop
recursion
number
target = 1
B.C
if(i >= n)
return false;
if(arr[i] == target)
return true;

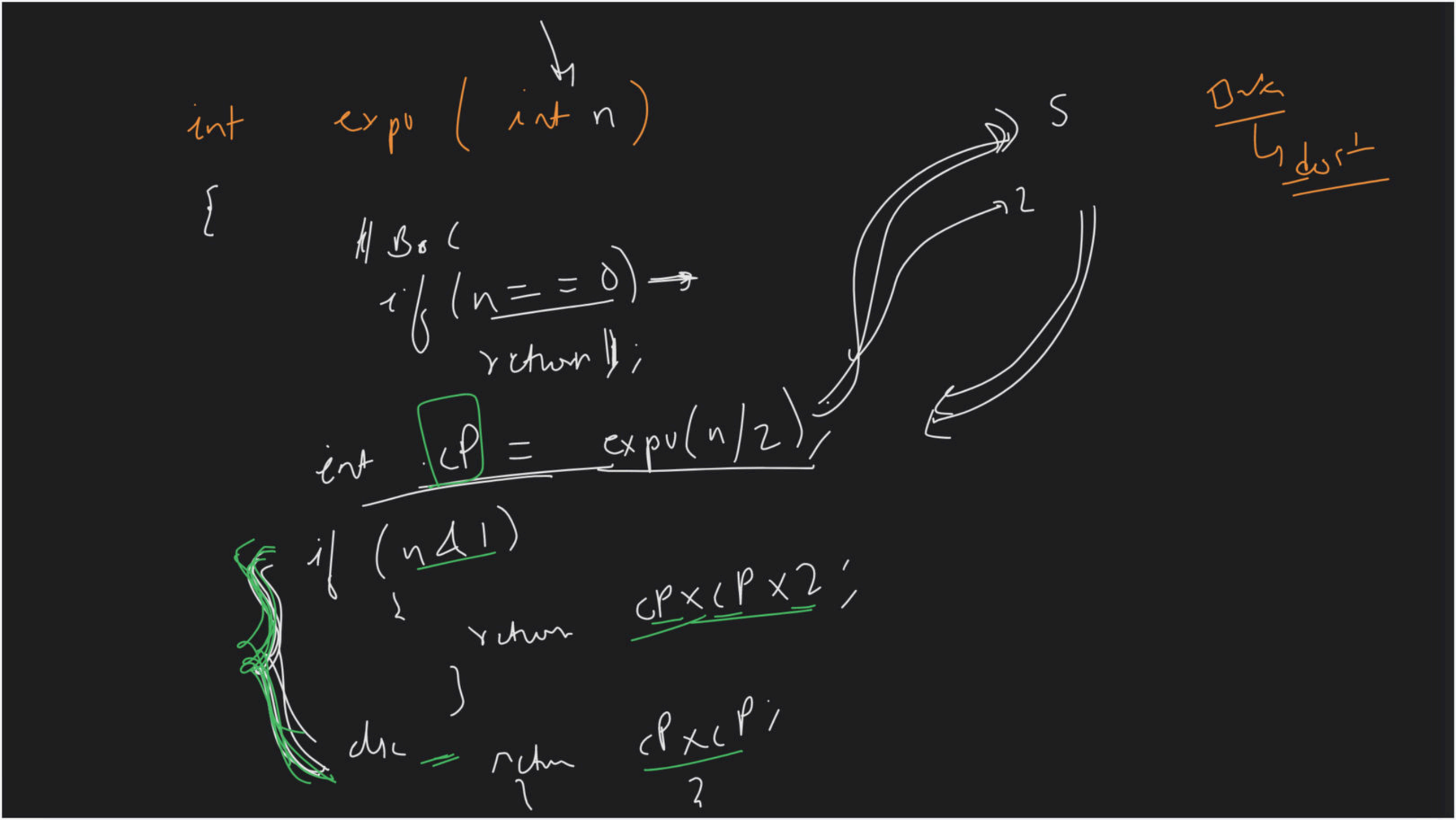

fast exponentiation
O(log n)
2^n
n/2 → n/2 → ... → 1
n is odd
n is even
2^n → 2^(n/2) × 2^(n/2) × 2
2^11 → 2^5 × 2^5 × 2
2^n = 2^(n/2) × 2^(n/2)
2^10 → 2^5 × 2^5
cp
cp
cp
cp
2
2^n = 2 × 2 × 2 × 2 × ... × 2
n times
O(n)

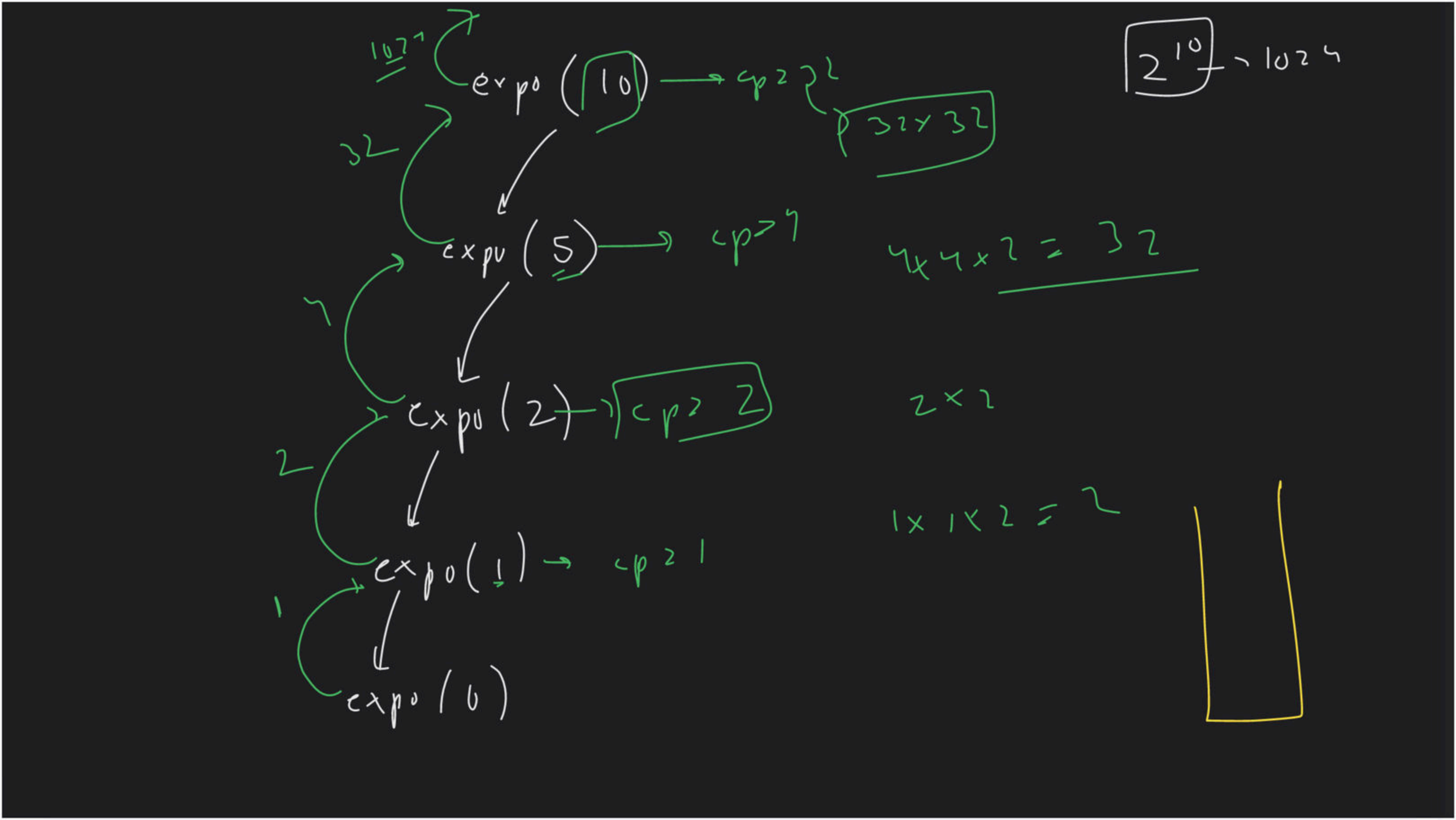
n → n/2 → n/4 → 0

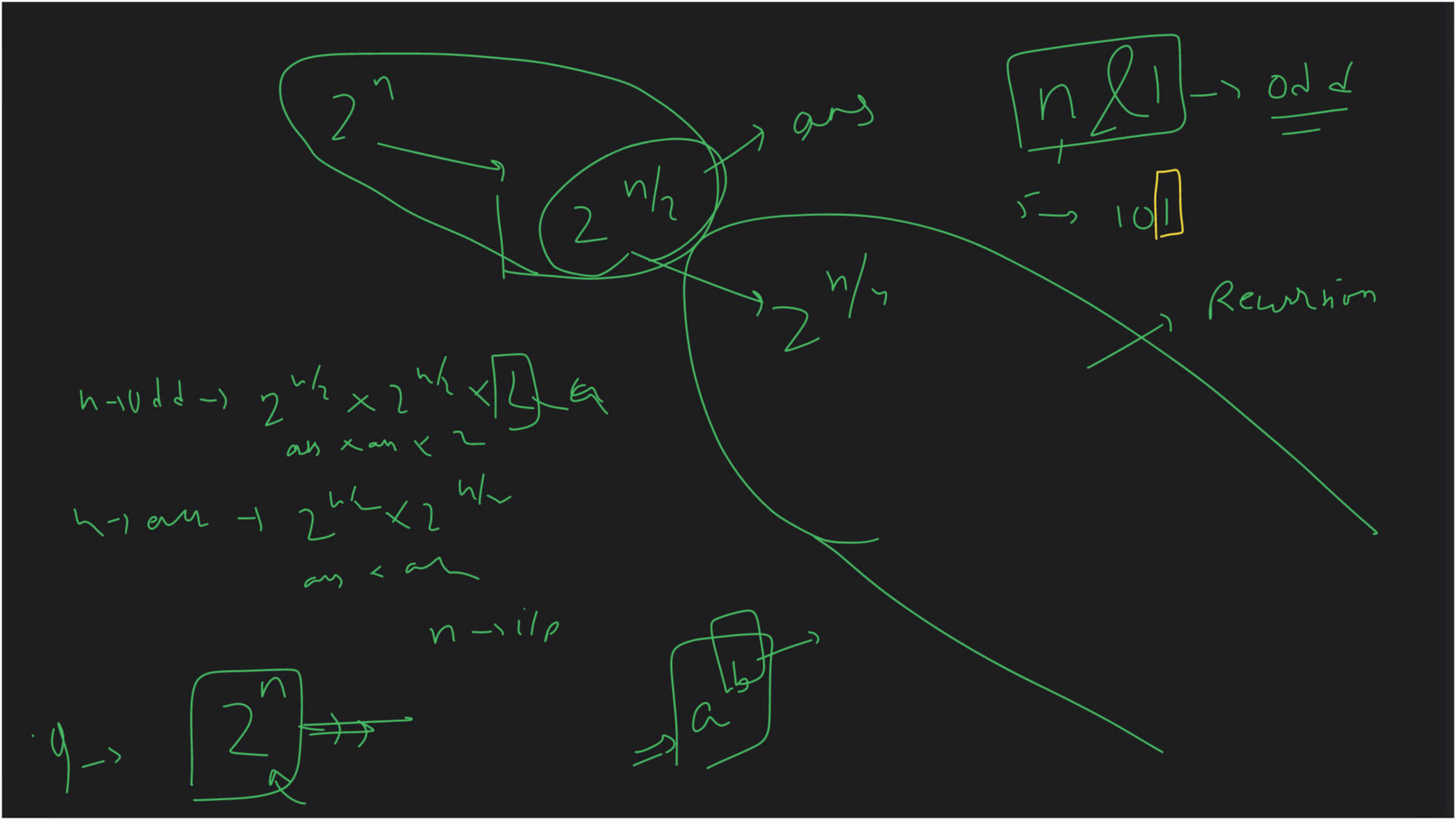
$2^0 = 1$

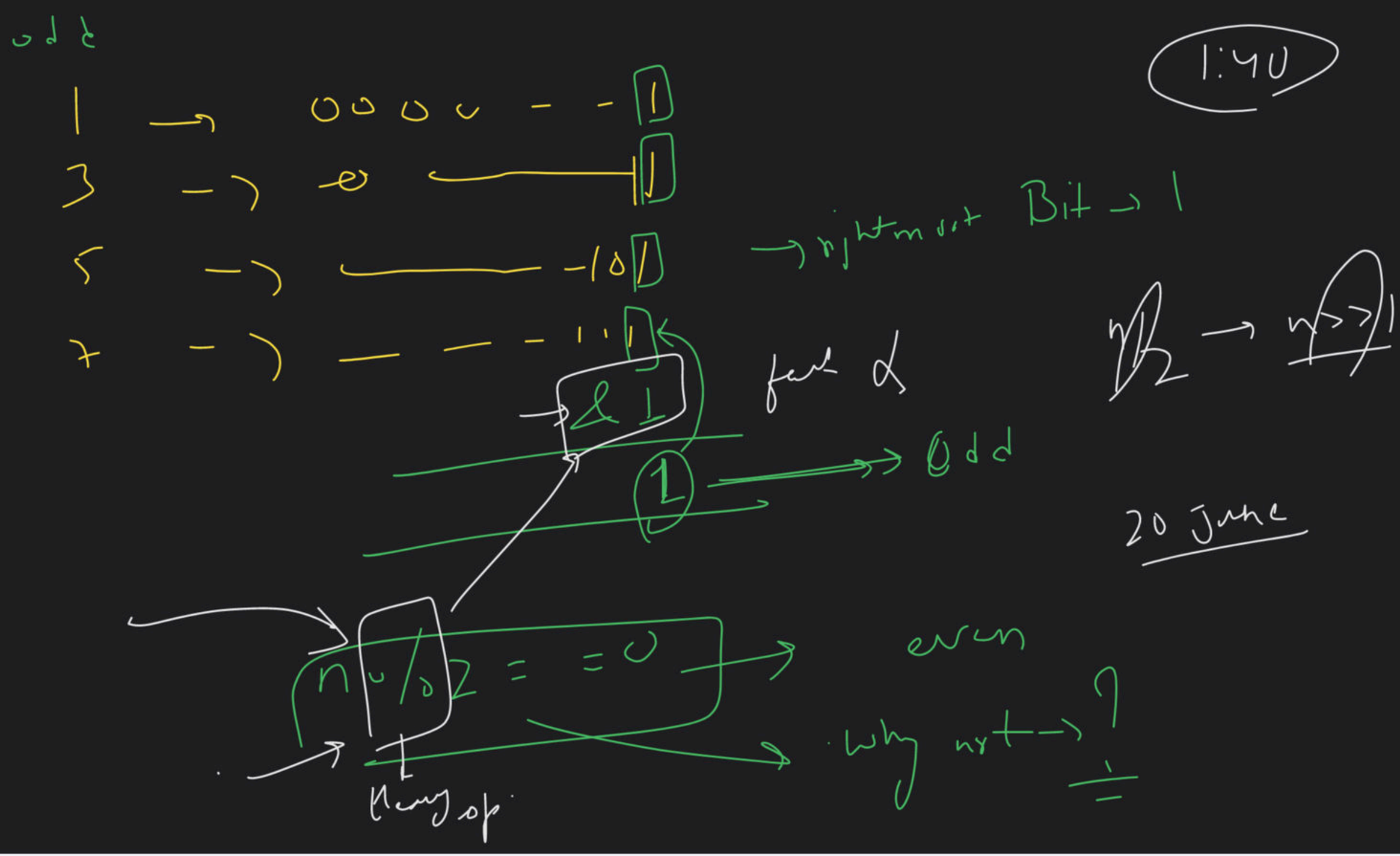
B.C

```
if (n == 0)
    return 1;
```


int expu (int n)
{
// Base
if (n == 0)
return 1;
int cp = expu(n/2);
if (n%2 == 1)
return cp*cp*2;
else
return cp*cp;
5
2
Divide and conquer


1024
expo(10) → cp=32
32
32x32
expo(5) → cp=?
?
4x4x2 = 32
expo(2) → cp=2
2
2x2
expo(1) → cp=1
1
1x1x2 = 2
expo(0)
2^10 → 1024

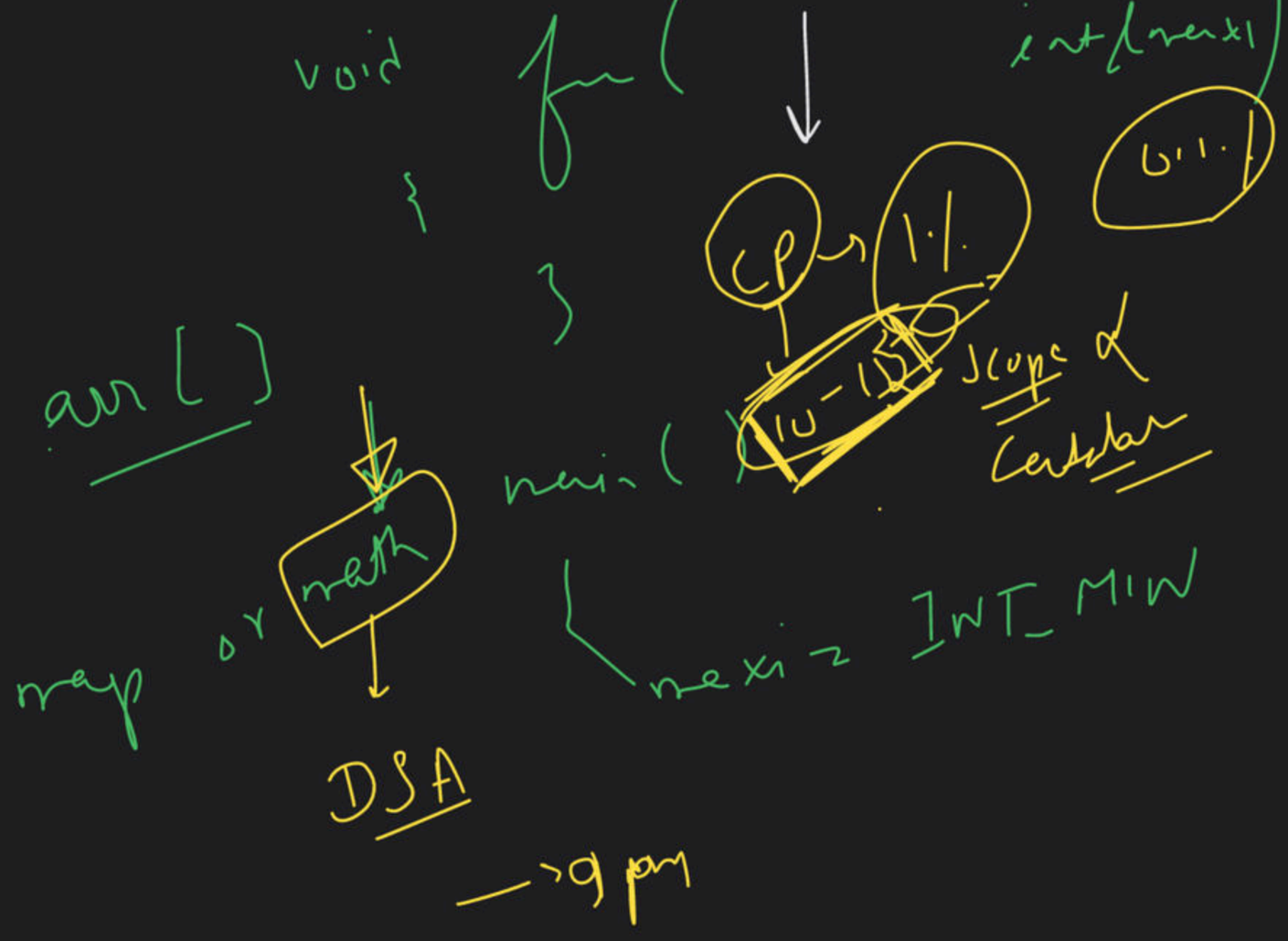

2^n
$2^{n/2}$
ans
n % 1
Odd
i → 101
$2^{n/2}$
Recursion
n-Odd → $2^{n/2} \times 2^{n/2} \times 1$ Eq
ans x ans < 2
n-1 ans → $2^{n/2} \times 2^{n/2}$
ans < ans
n → i/p
2^n
a
b

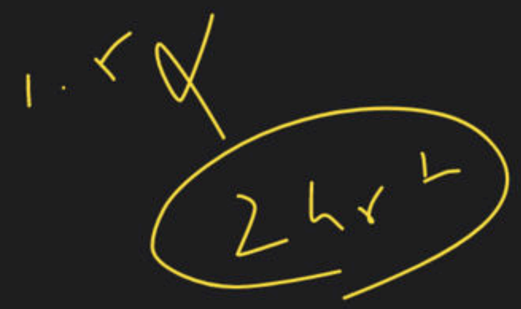

odd
1:40
1 → 0 0 0 0 ... 1
3 → ... 1
5 → ... 101
7 → ... 111
→ rightmost Bit → 1
fast
n>>1
1
Odd
20 June
n%2==0
even
why not →
Happy

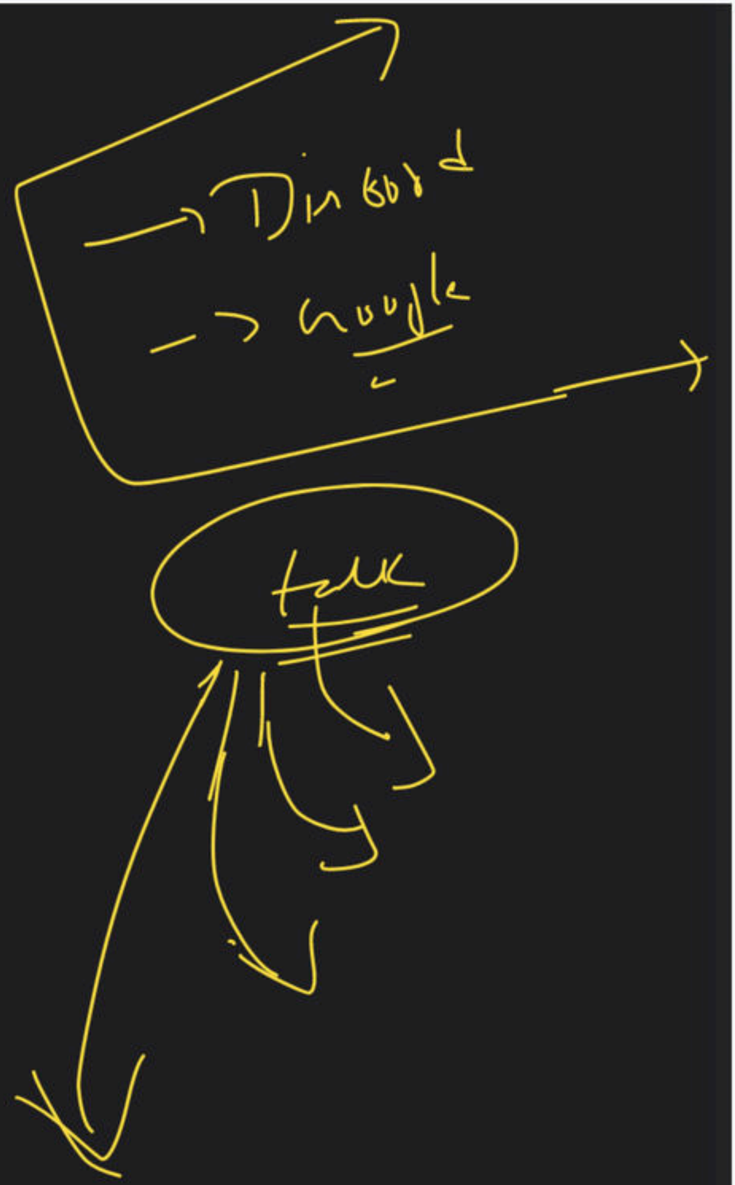

Recursion
Static
default
1 month
DS
Repeat
Recursion
Binary Search
Rec
for
Rec
slow
week
watch
judge
20-45

string → int

baba → 11




2 hr


Dinner
Snuggle
talk