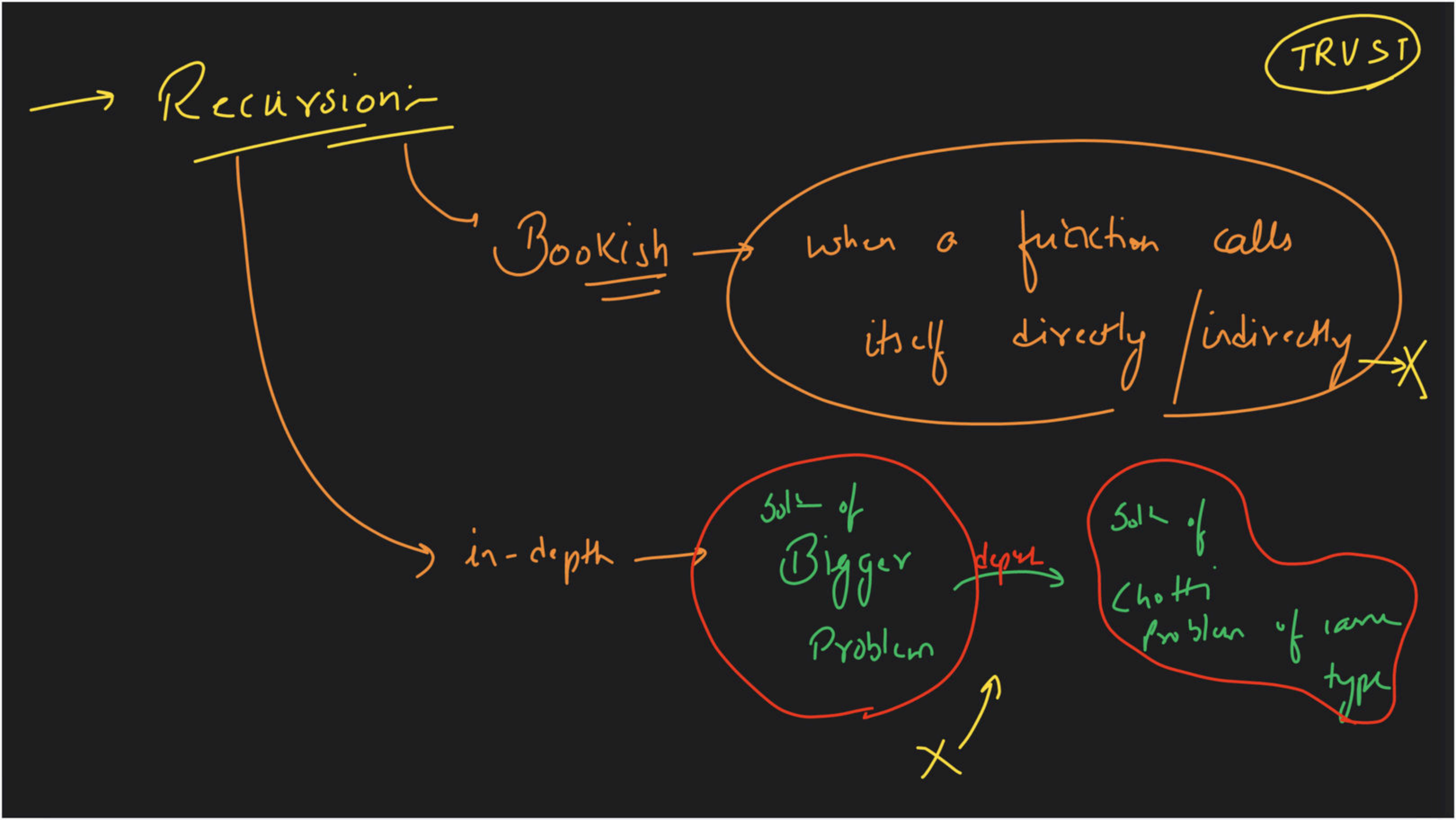
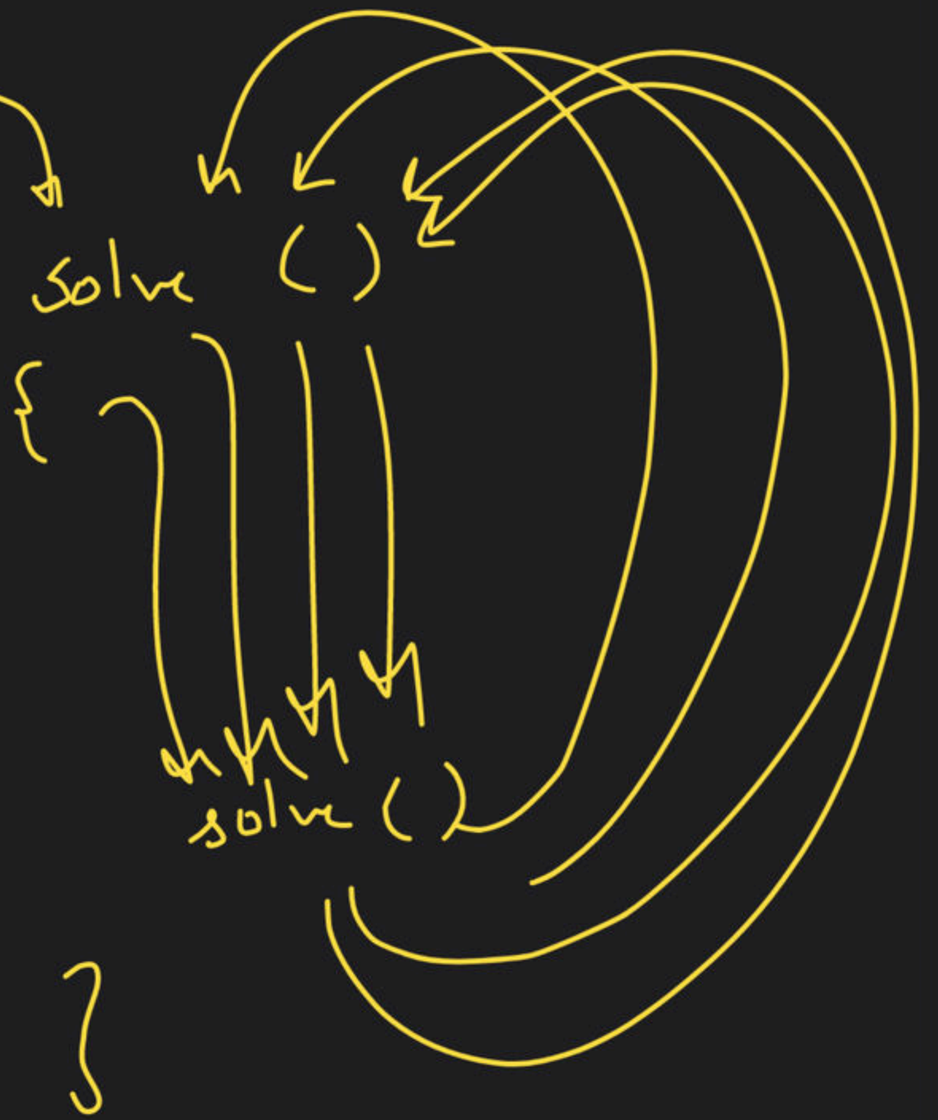
Recursion - Class 1

Special class

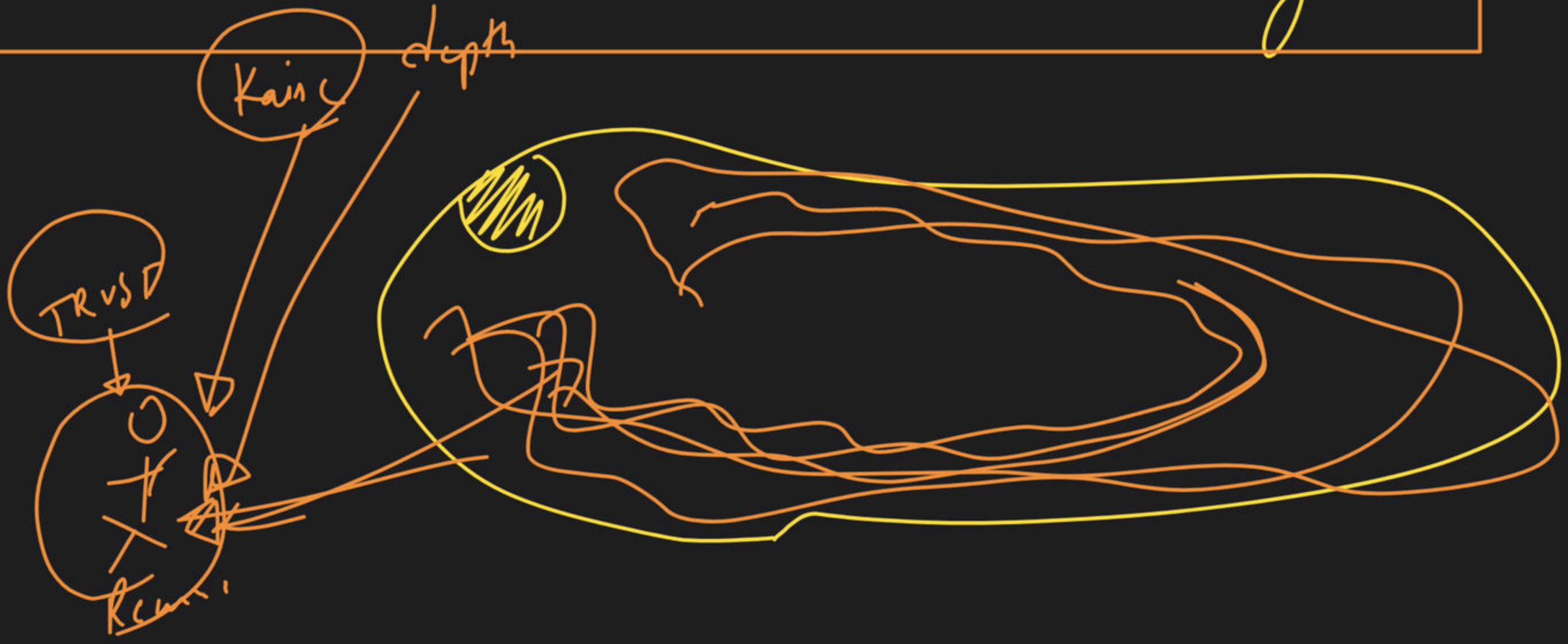
Love Babbar • Oct 9, 2023

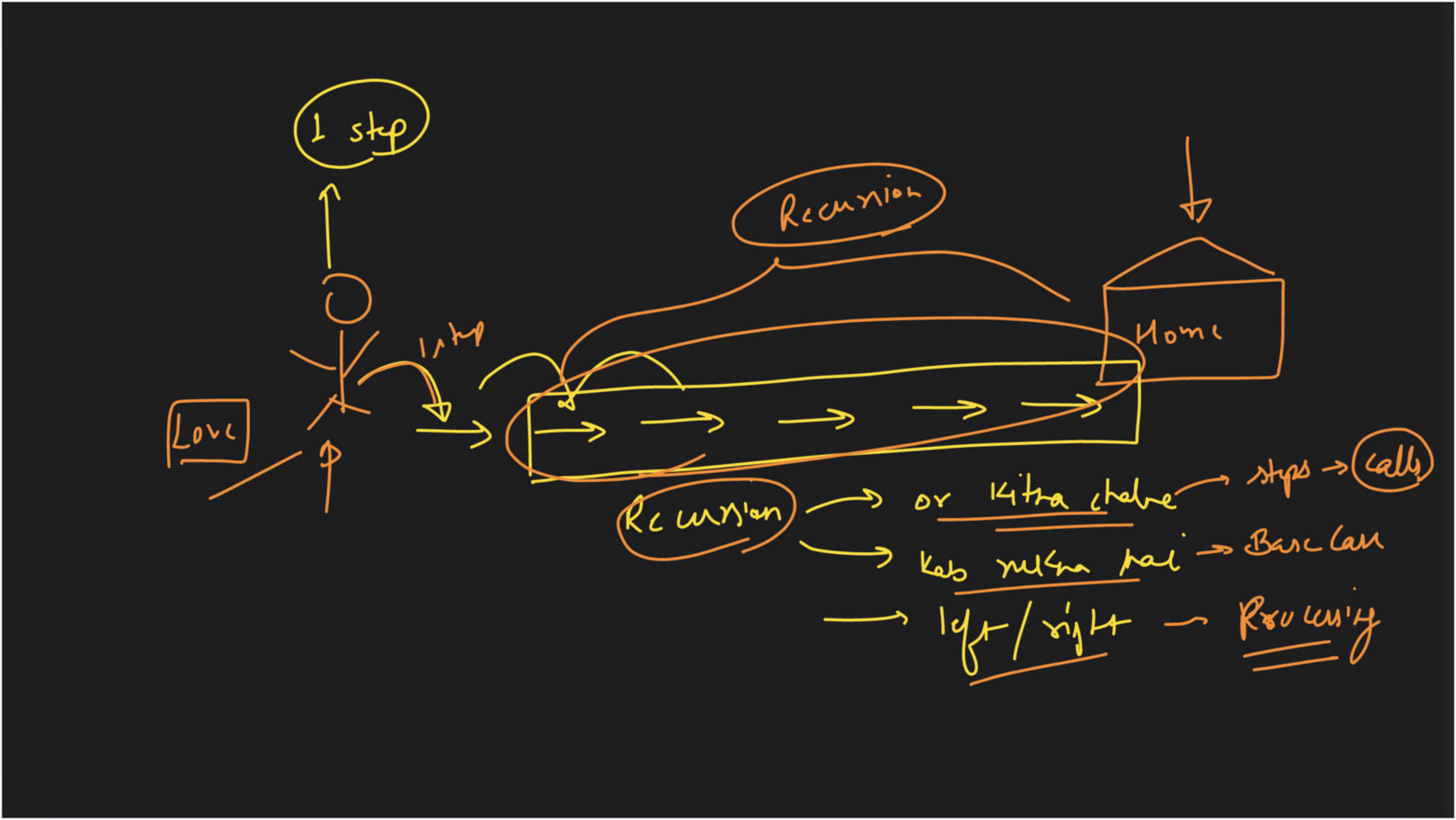

TRUST
Recursion
Bookish
when a function calls itself directly / indirectly
X
in-depth
Sol^n of Bigger Problem
depend
Sol^n of Chotti problem of same type
X

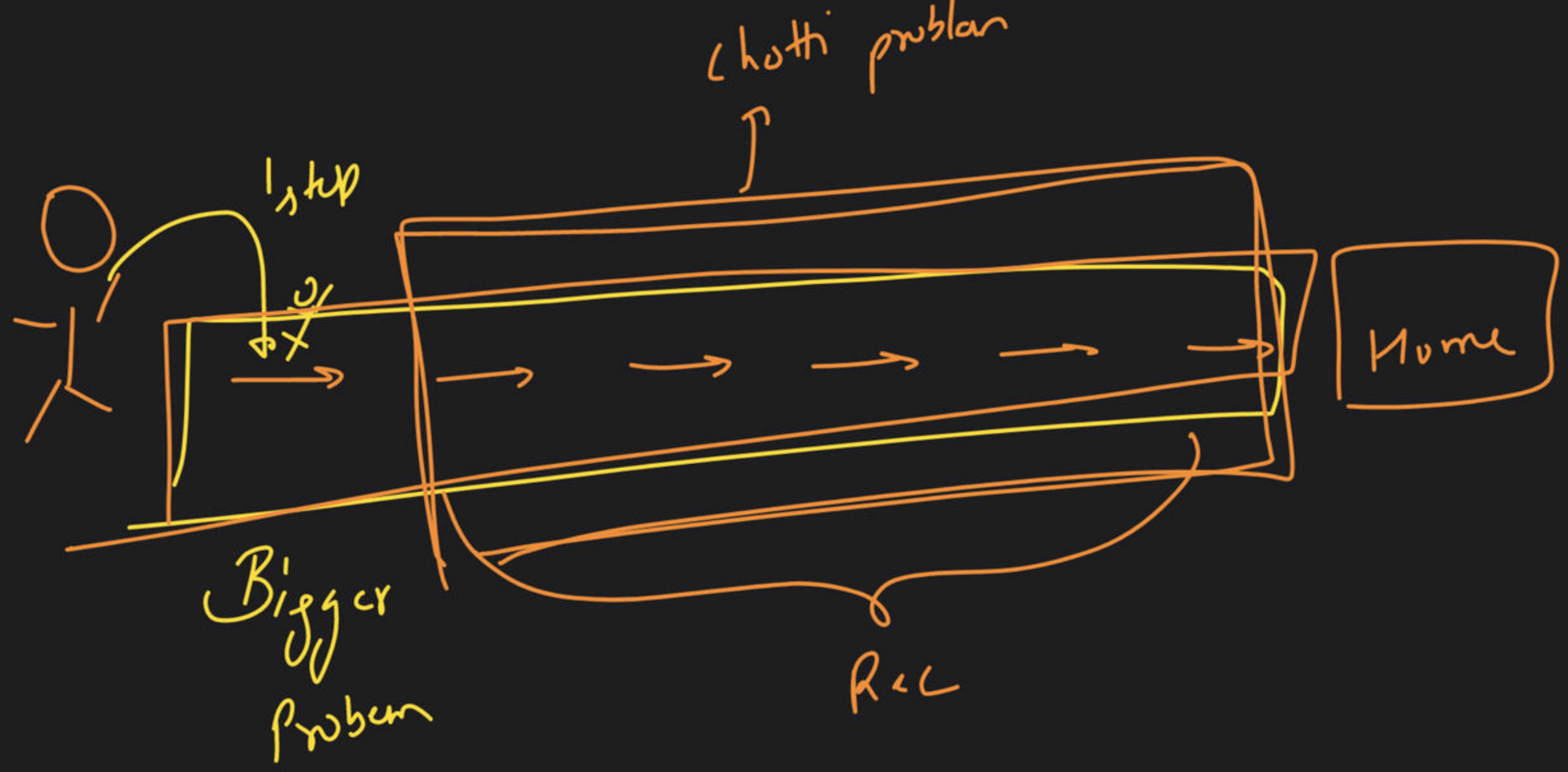
Recursion :-



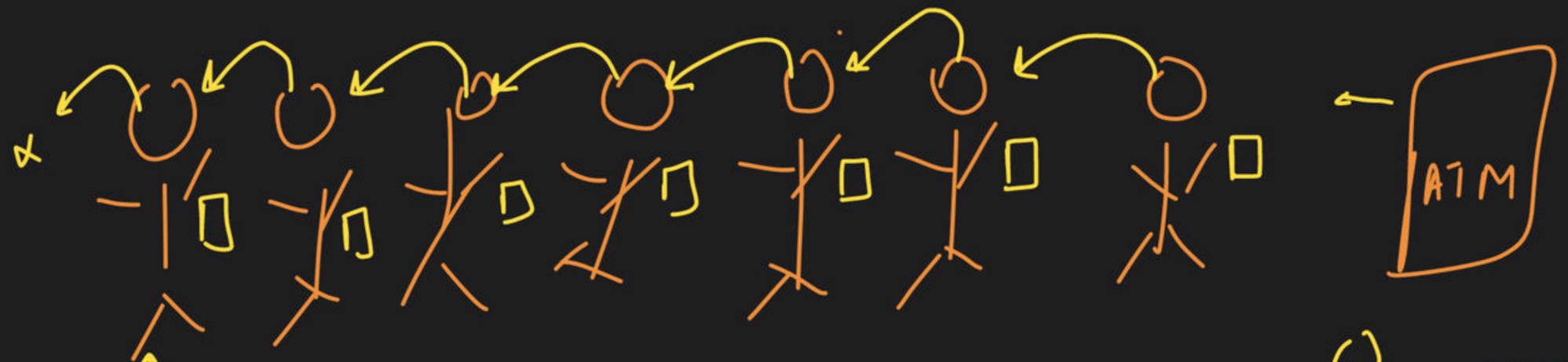
→ 1 case tum solve karo, baaki recursion
sambhal lega




1 step
Recursion
Home
1 step
Love
Recursion
or kitna chalne → steps → calls
kab rukna hai → Base Case
left / right → Recursion


Chotti problem
1 step
Bigger Problem
R<L
Home

→ To understand Recursion, you need to first understand Recursion

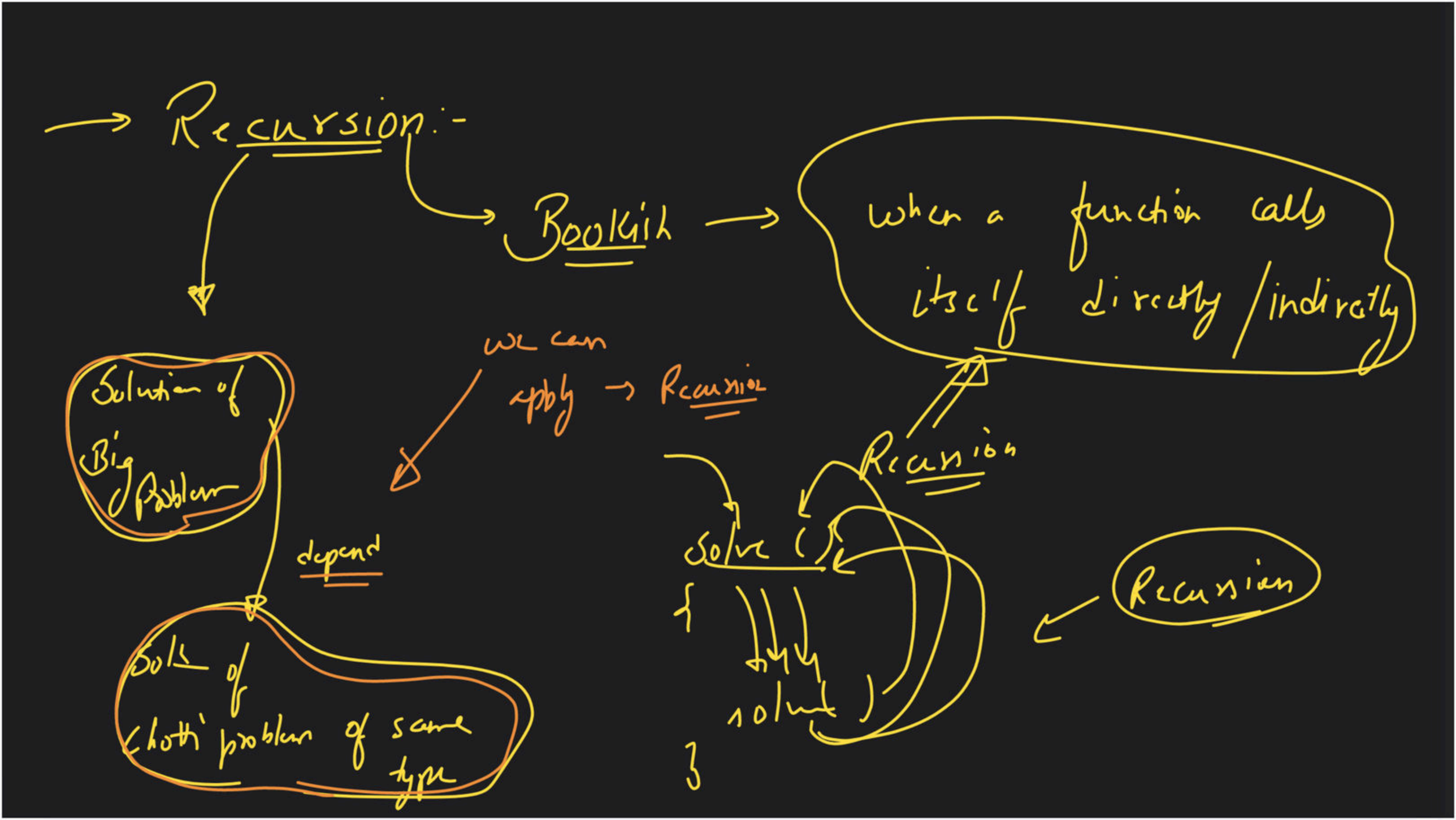


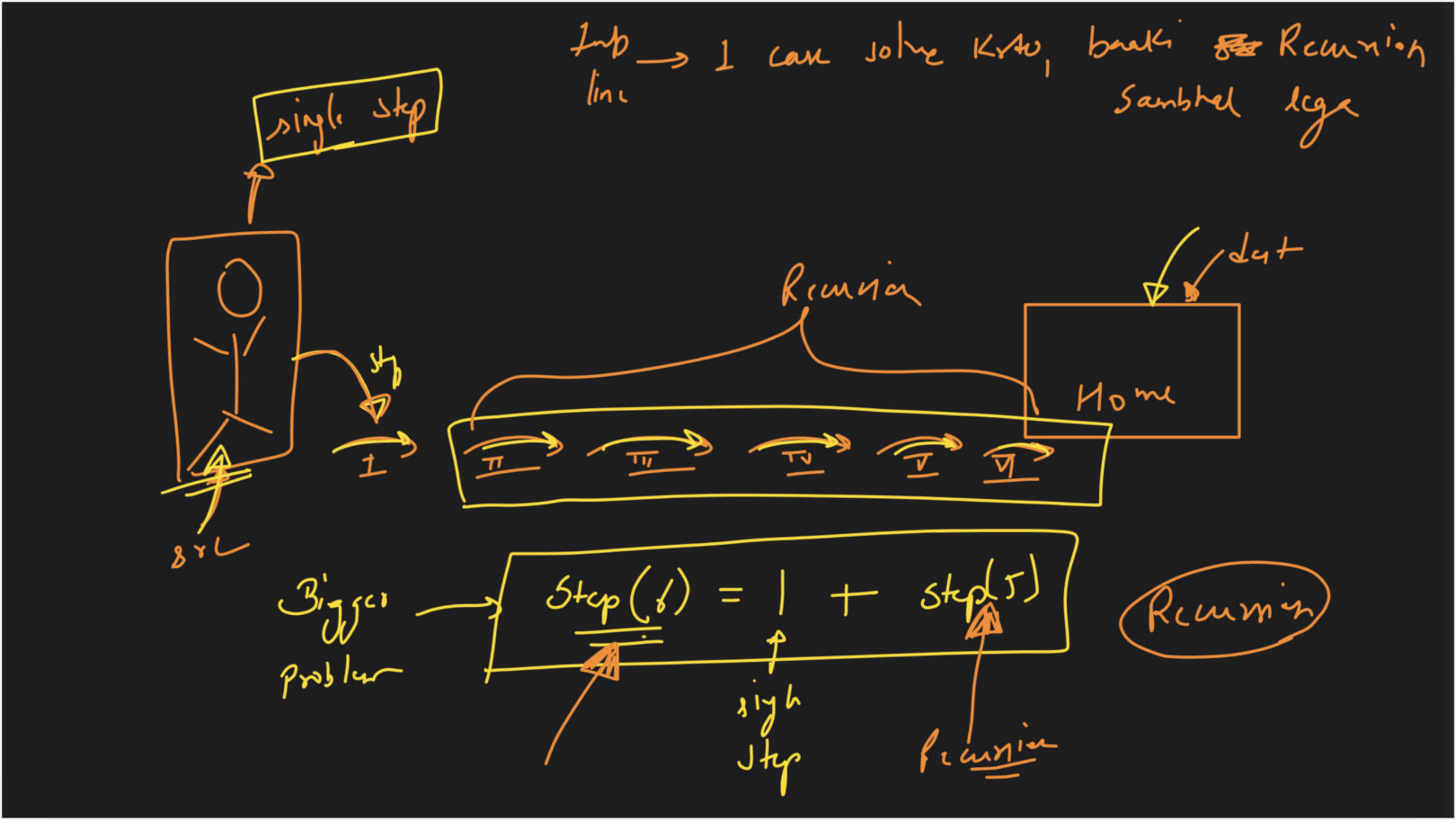
nuk gyl

Ruk →

(A) pichi koi vyakti pass paper

(B) agar koi nahi khada pichi, ruk jao


Recursion :-
Bookish
when a function calls itself directly / indirectly
Solution of Big Problem
we can apply → Recursion
depend
Sols of thoda problem of same type
Recursion
solve ()
{
...
solve ()
}
Recursion


single step
Recursion
Home
Step(6) = 1 + step[5]
Recursion

~~f(n)~~

solve(n) ⟶ 2^n

solve(n-1) = 2^{n-1}

Bigger problem → 2^n

$solve(n) = 2^n$

$solve(n) = 2 \times 2^{n-1}$

$solve(n) = 2 \times solve(n-1)$

Big

Chotti

solu(n) → (n ⟶ 1) counting print

solu(n-1) → (n-1 ⟶ 1) counting

solu(n) → n, n-1, n-2, ---, 1

solu(n) = n, (n-1, n-2, --- 1)

solu(n) = n, solu(n-1)

Factorial

$$5! = 5 \times 4 \times 3 \times 2 \times 1$$

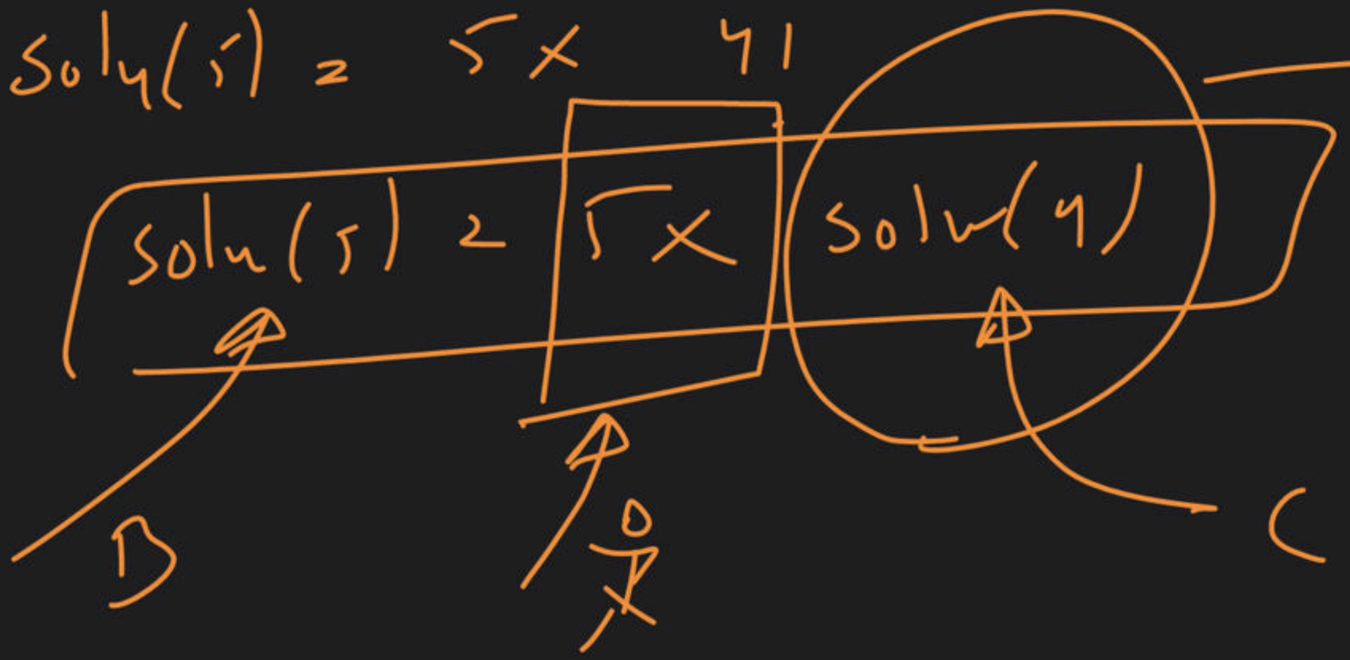
soln(5) = 5!

soln(5) = 5 × 4 × 3 × 2 × 1

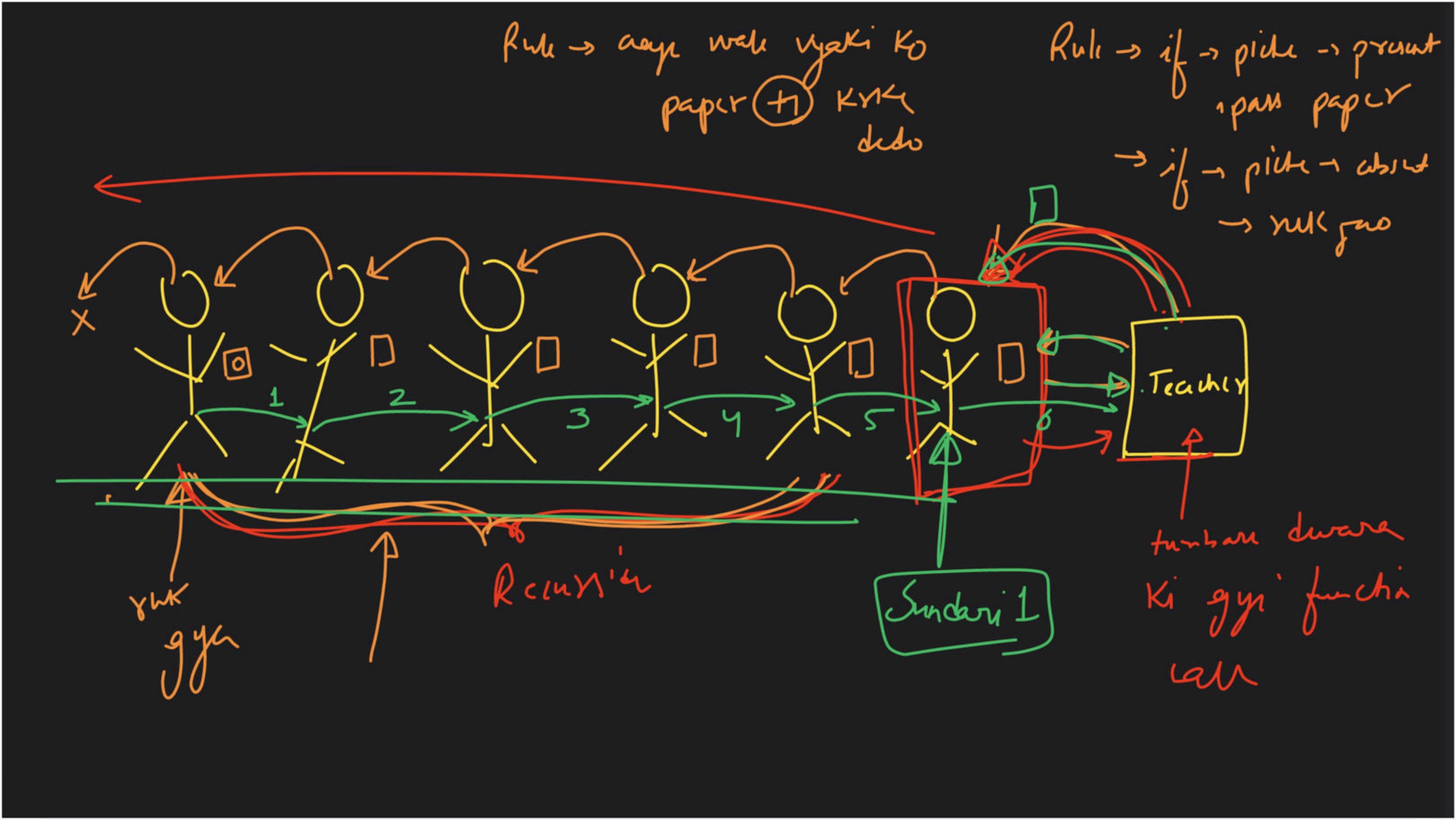
soln(4) = 4!

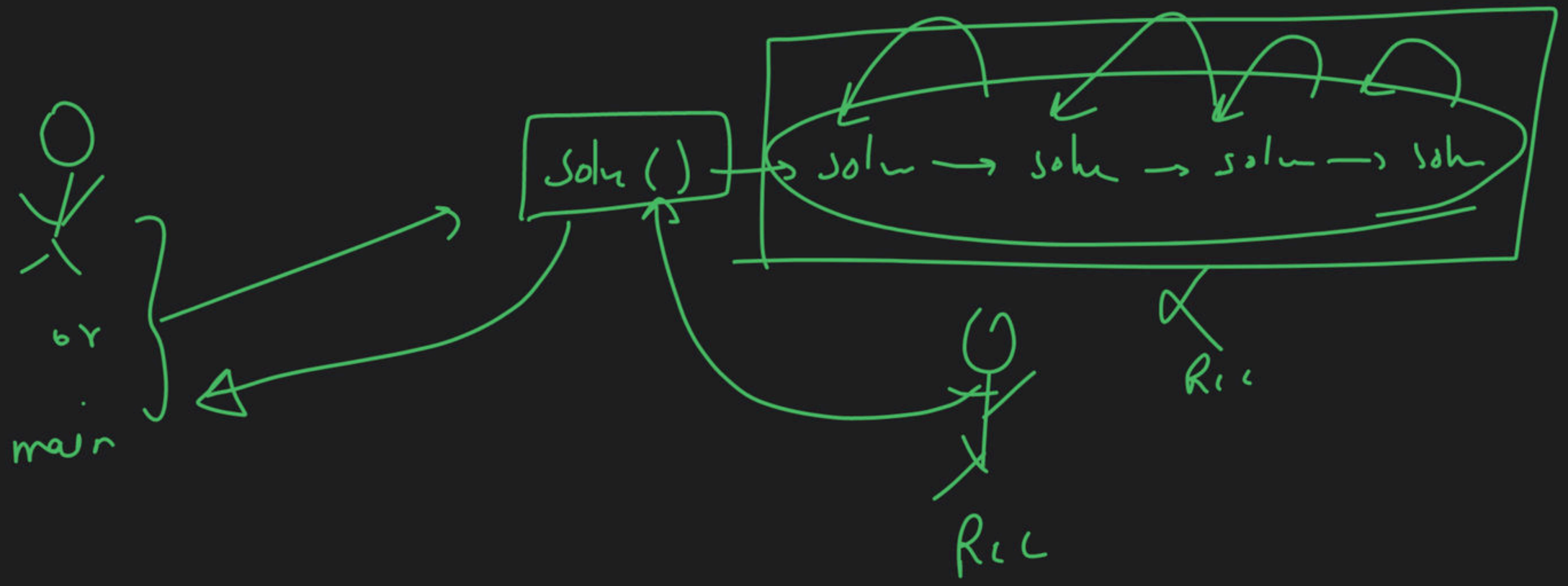
= 4 × 3 × 2 × 1

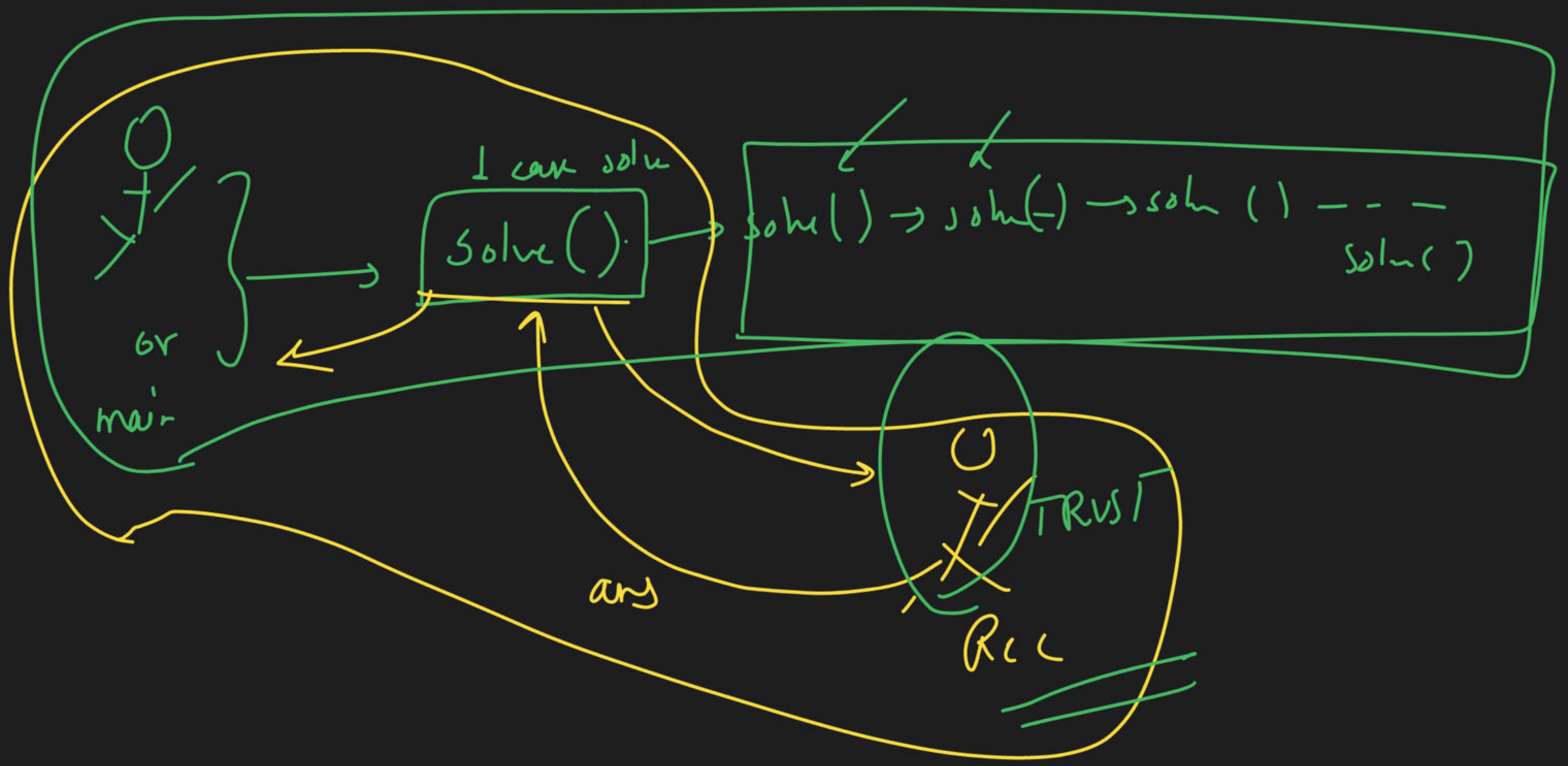
soln(5) = 5 × soln(4)

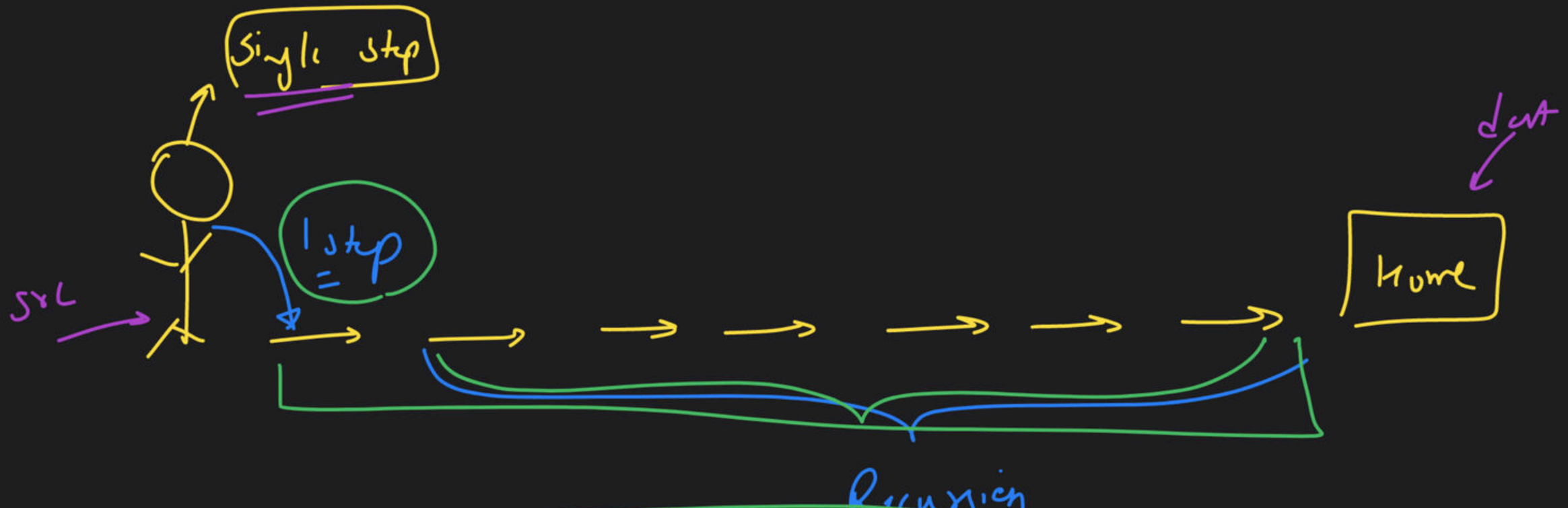


→ Recursion


Rule → aage wale vyakti ko
paper (+1) krke
dedo
Rule → if → pick → present
→ pass paper
→ if → pick → absent
→ sab ko
X
0
1
2
3
4
5
6
Teacher
Sundari 1
Recursion
gya


Solu ( )
Solu → Solu → Solu → Solu
or
main
Rec
Rec


I can solve
Solve()
solve() → solve() → solve() - - -
solve()
or
main
any
TRUST
Rec



Overall = 1 + Recursion Ans

Ans

Big 2 1 + Cloth

Factorial →

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

$5! = 5 \times 4 \times 3 \times 2 \times 1$

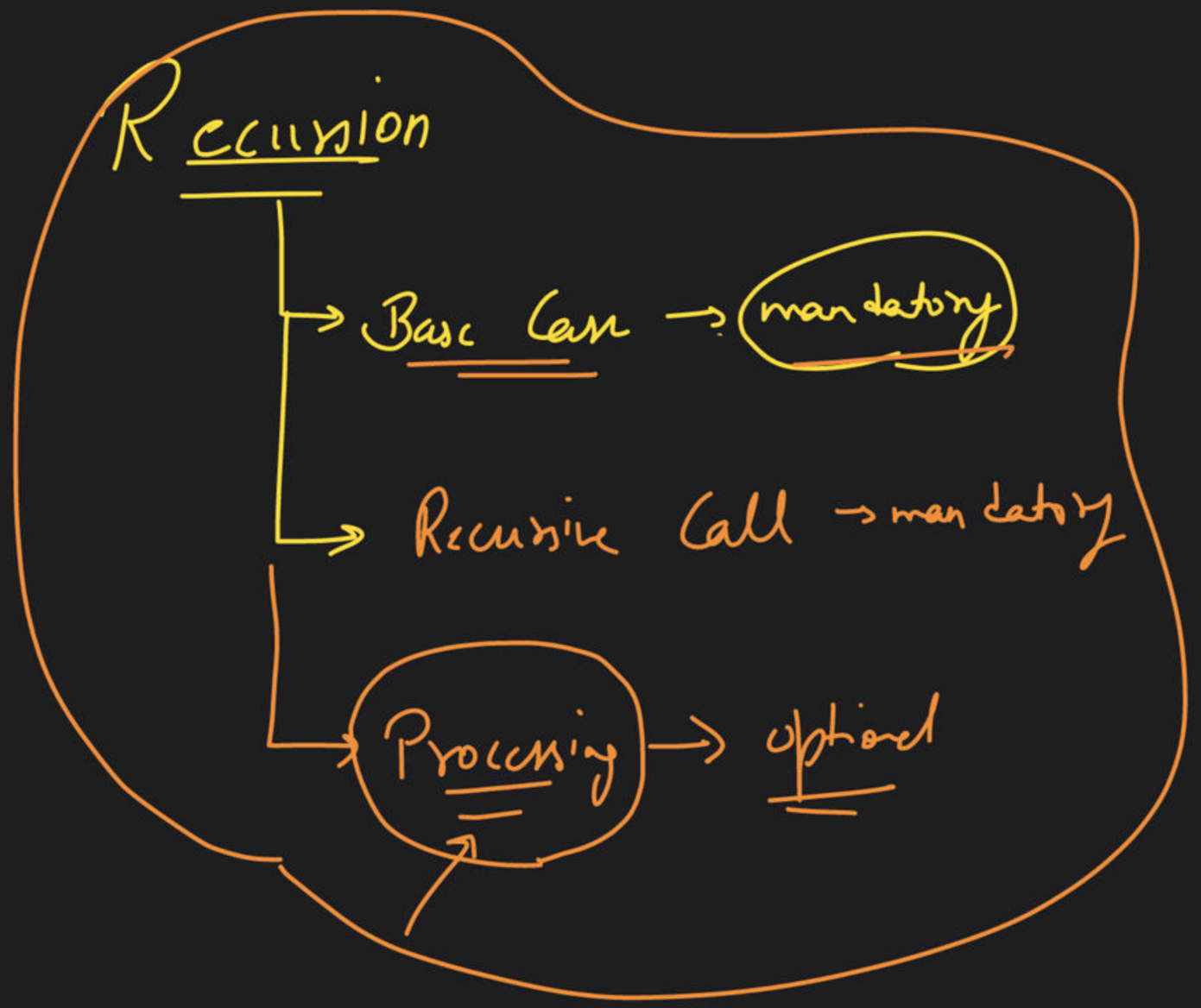
$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$

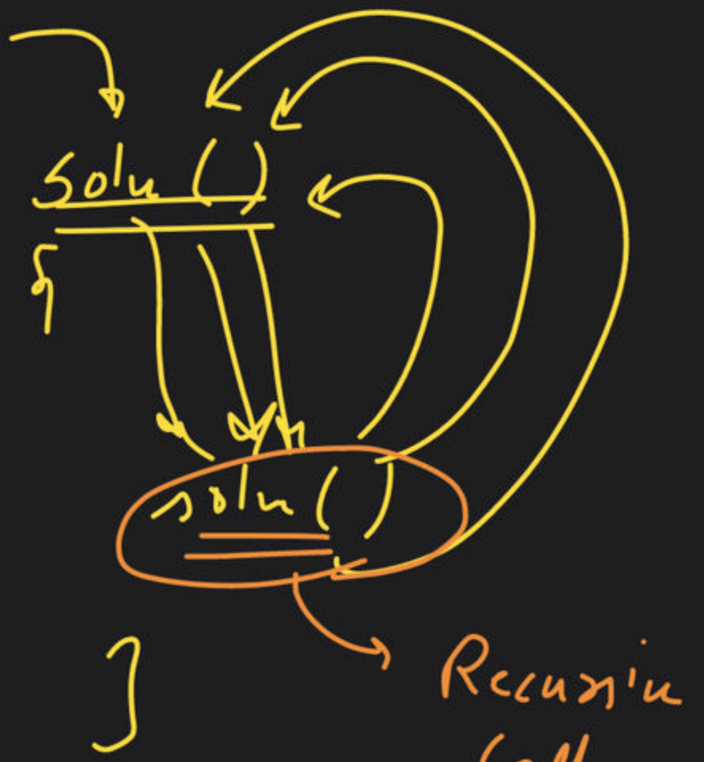
if(fact(1)) return 1

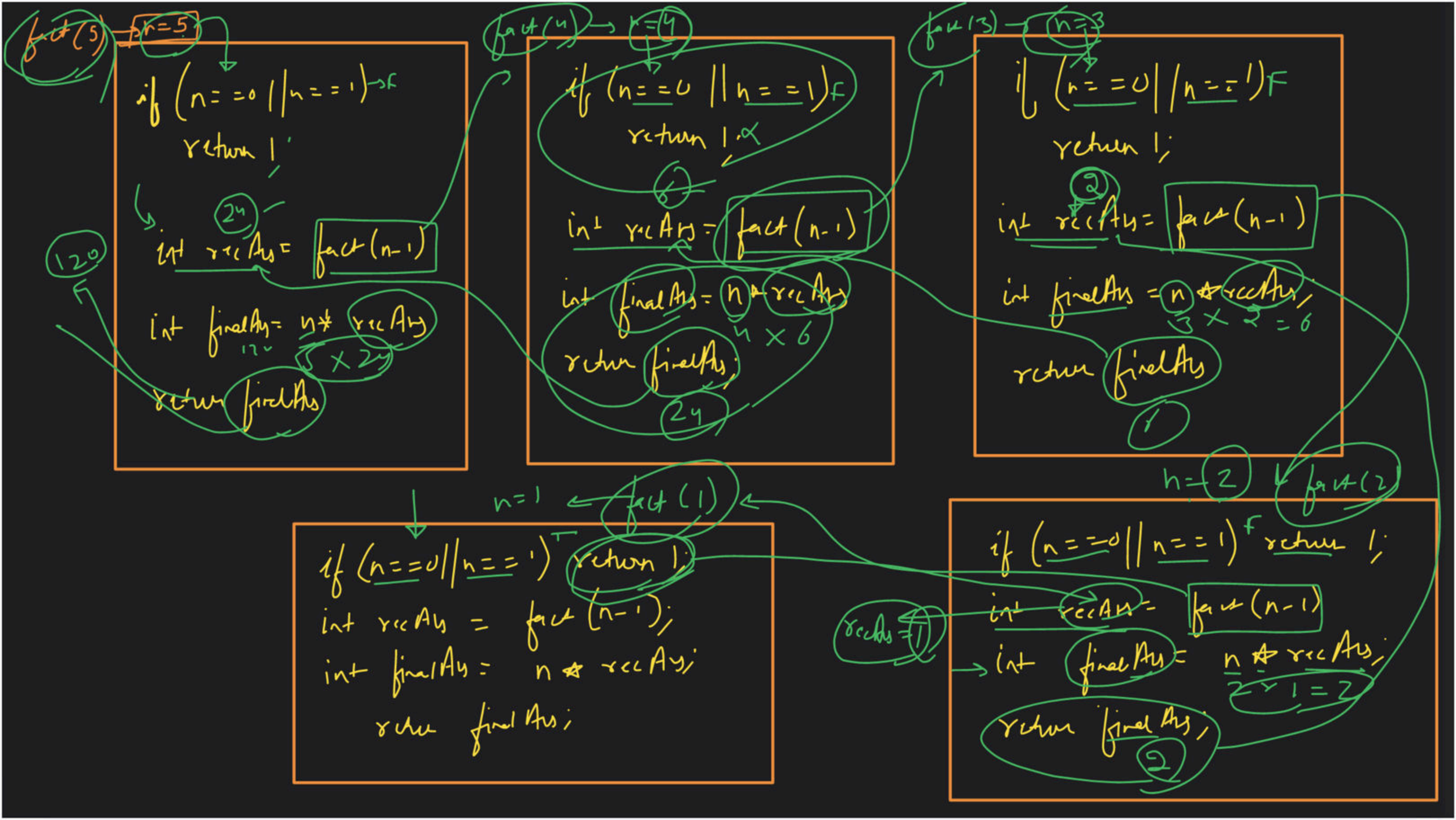
fact(7) = 7 × 6!

fact(7) = 7 × fact(6)

fact(n) = n × fact(n-1)


Recursion
Basic Case → mandatory
Recursive Call → mandatory
Processing → optional


solu ( )
{
solu ( )
}
Recursion
Call
relati

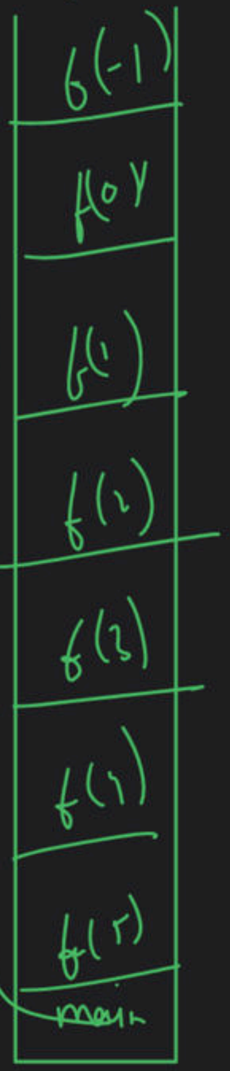

fact(5) → n=5
if (n==0 || n==1) → F
return 1;
int recAns = fact(n-1)
int finalAns = n * recAns
return finalAns
120
24
fact(4) → n=4
if (n==0 || n==1) F
return 1;
int recAns = fact(n-1)
int finalAns = n * recAns
4 × 6
return finalAns;
24
fact(3) → n=3
if (n==0 || n==1) F
return 1;
int recAns = fact(n-1)
int finalAns = n * recAns
3 × 2 = 6
return finalAns
2
n=1 ← fact(1)
if (n==0 || n==1) T return 1;
int recAns = fact(n-1);
int finalAns = n * recAns;
return finalAns;
recAns = 1
n=2 ← fact(2)
if (n==0 || n==1) F return 1;
int recAns = fact(n-1)
int finalAns = n * recAns;
2 × 1 = 2
return finalAns;
2

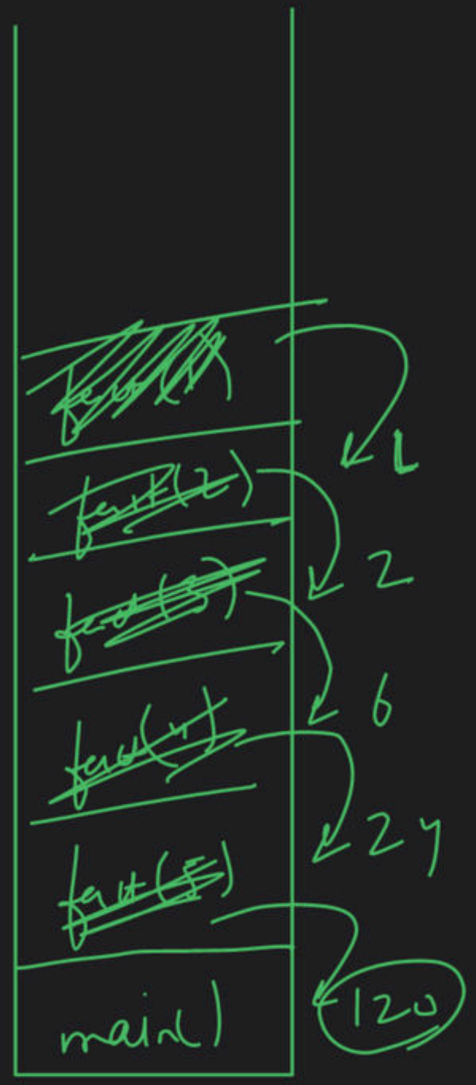
Base Case !

f(-4)
f(-3)
f(-2)

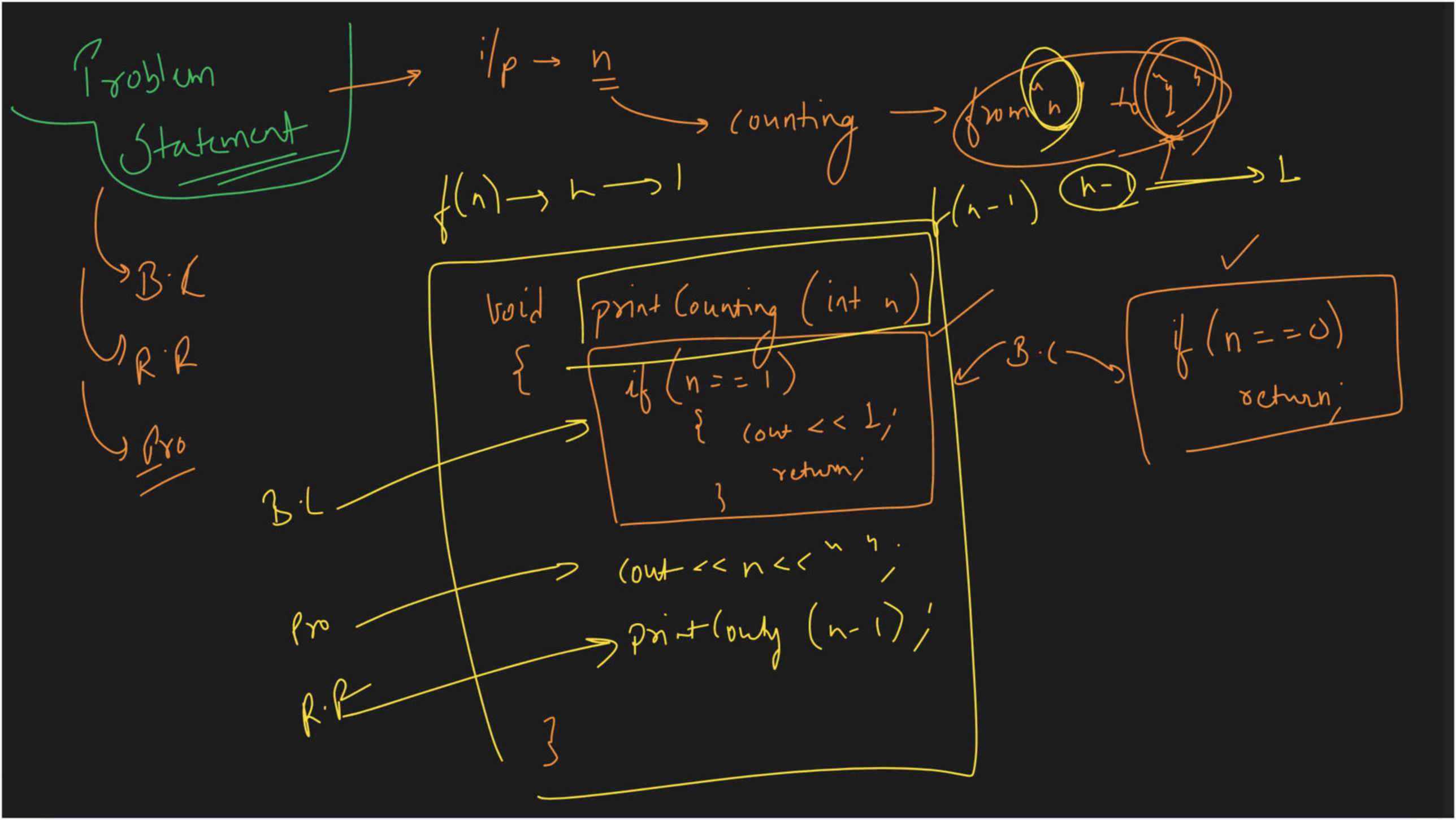
H/W

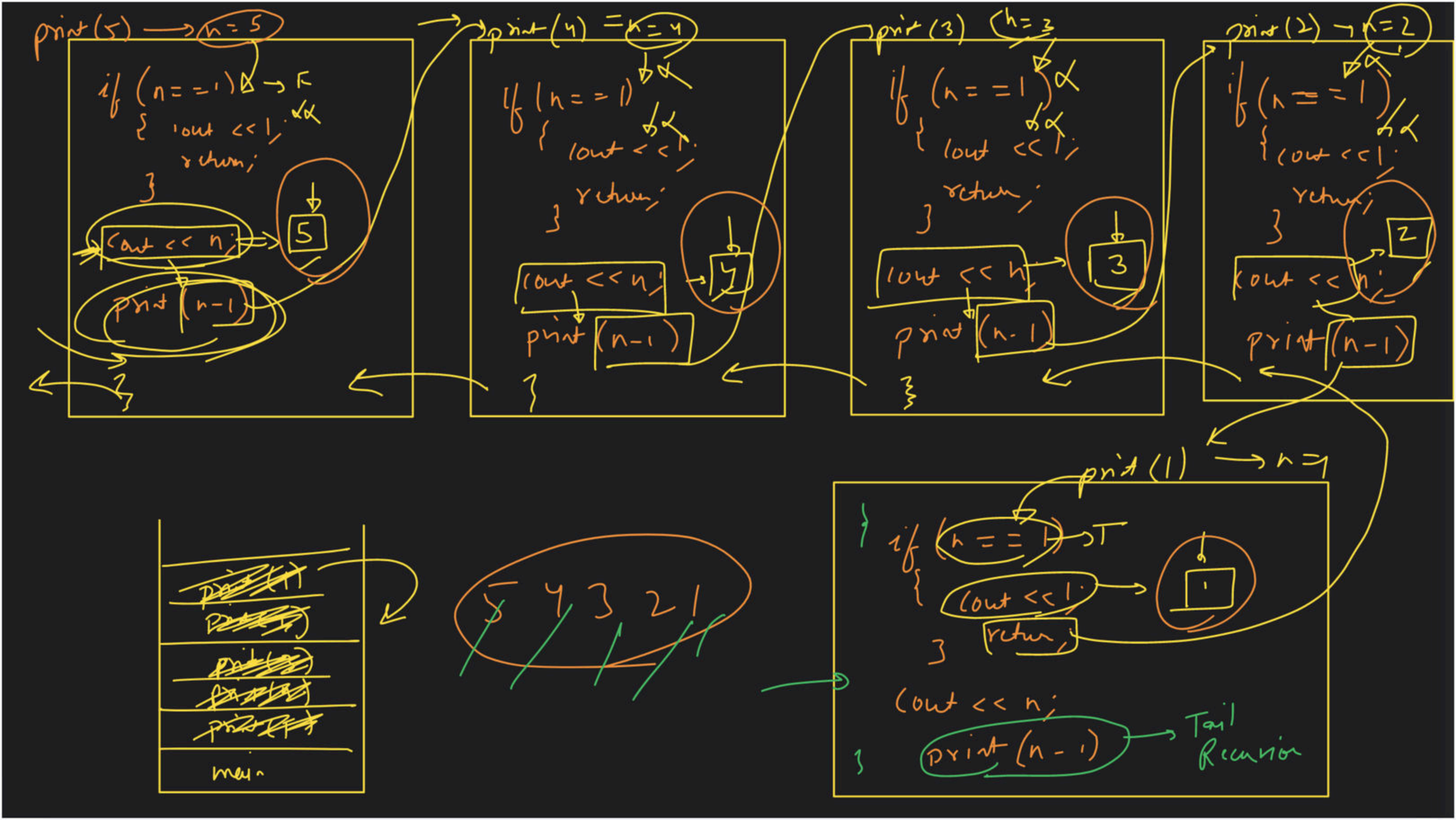
Stack Overflow

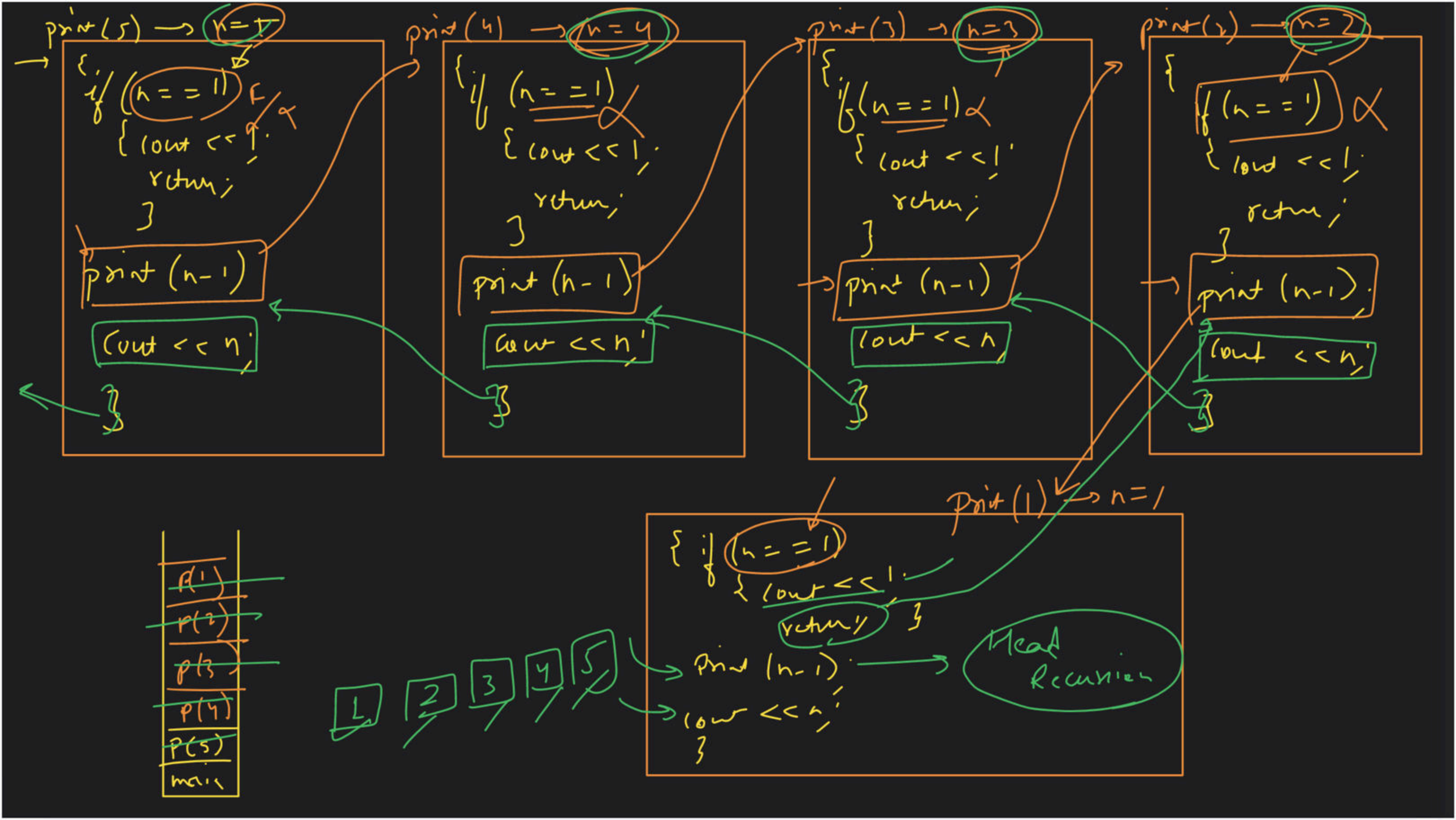


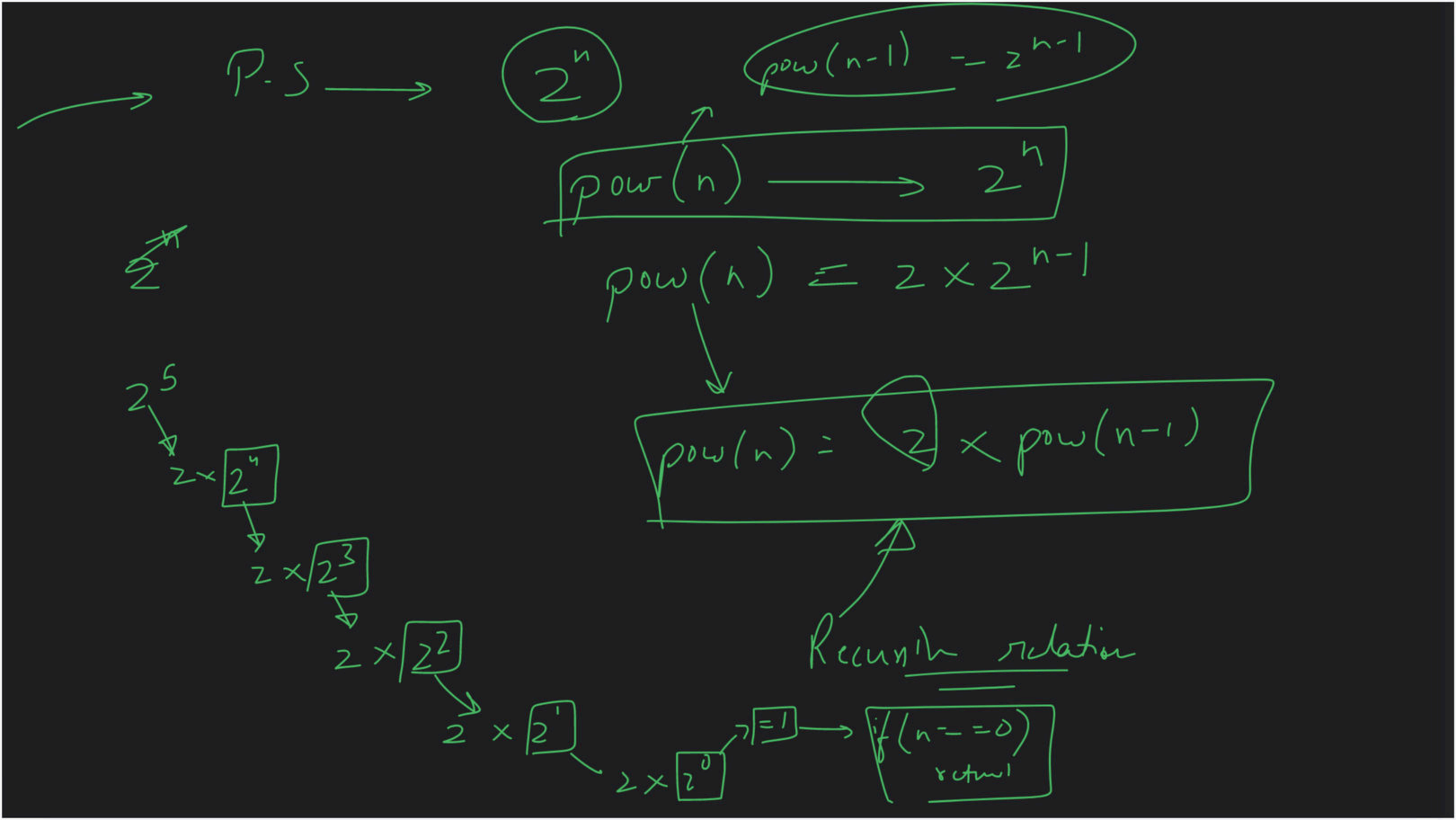


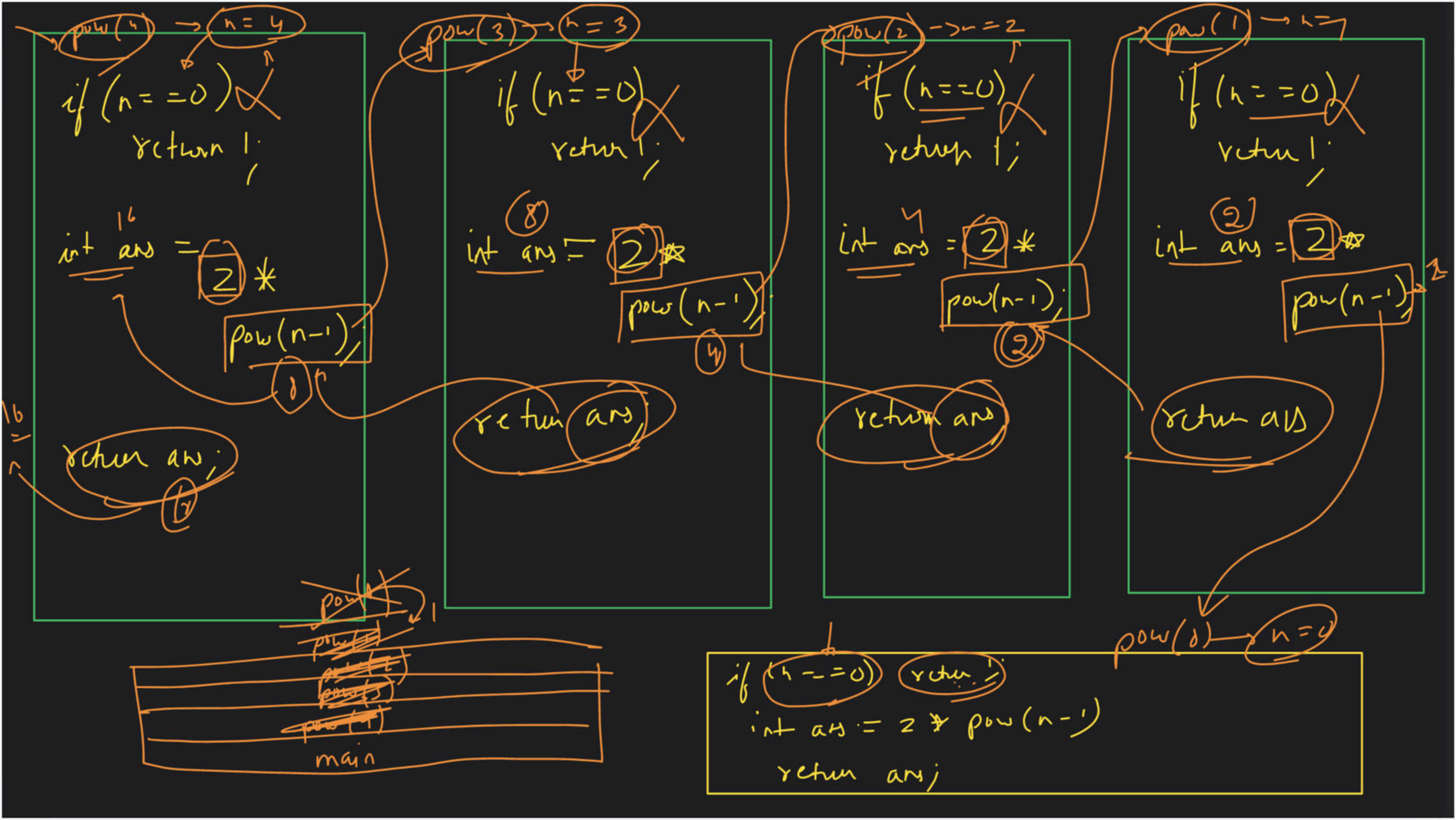
call Stack


Problem Statement
i/p → n
counting → from n to 1
f(n) → n → 1
f(n-1)
n-1 → 1
B.C
R.R
Pro
Void printCounting (int n)
{
if (n == 1)
{ cout << 1;
return;
}
B.C
B.C → if (n == 0) return;
B.C
cout << n << " ";
Pro
printCounting (n-1);
R.R
}

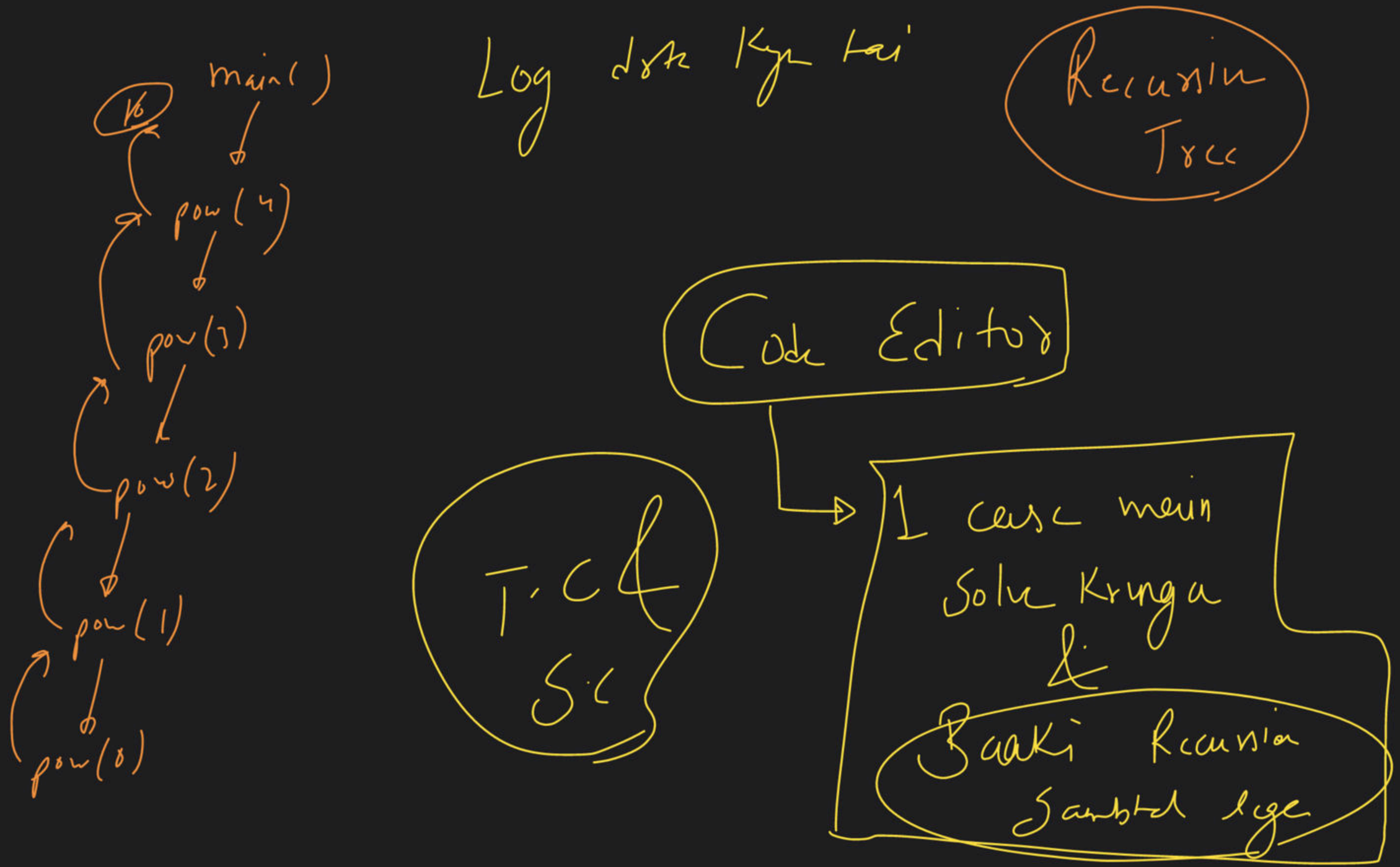

print(5) → n=5
if (n==1) → T
{ cout << 1;
return;
}
cout << n;
print (n-1)
print(4) → n=4
print(3) → n=3
print(2) → n=2
print(1) → n=1
5 4 3 2 1
Tail Recursion
main


print(5) → n=5
print(4) → n=4
print(3) → n=3
print(2) → n=2
print(1) → n=1
if (n==1)
cout << 1;
return;
print(n-1)
cout << n;
F
T
P(1)
P(2)
P(3)
P(4)
P(5)
main
1 2 3 4 5
Head Recursion


P.S
2^n
pow(n-1) = 2^{n-1}
pow(n) → 2^n
pow(n) = 2 × 2^{n-1}
2^5
2 × 2^4
2 × 2^3
2 × 2^2
2 × 2^1
2 × 2^0
=1
if(n==0)
return
Recursive relation

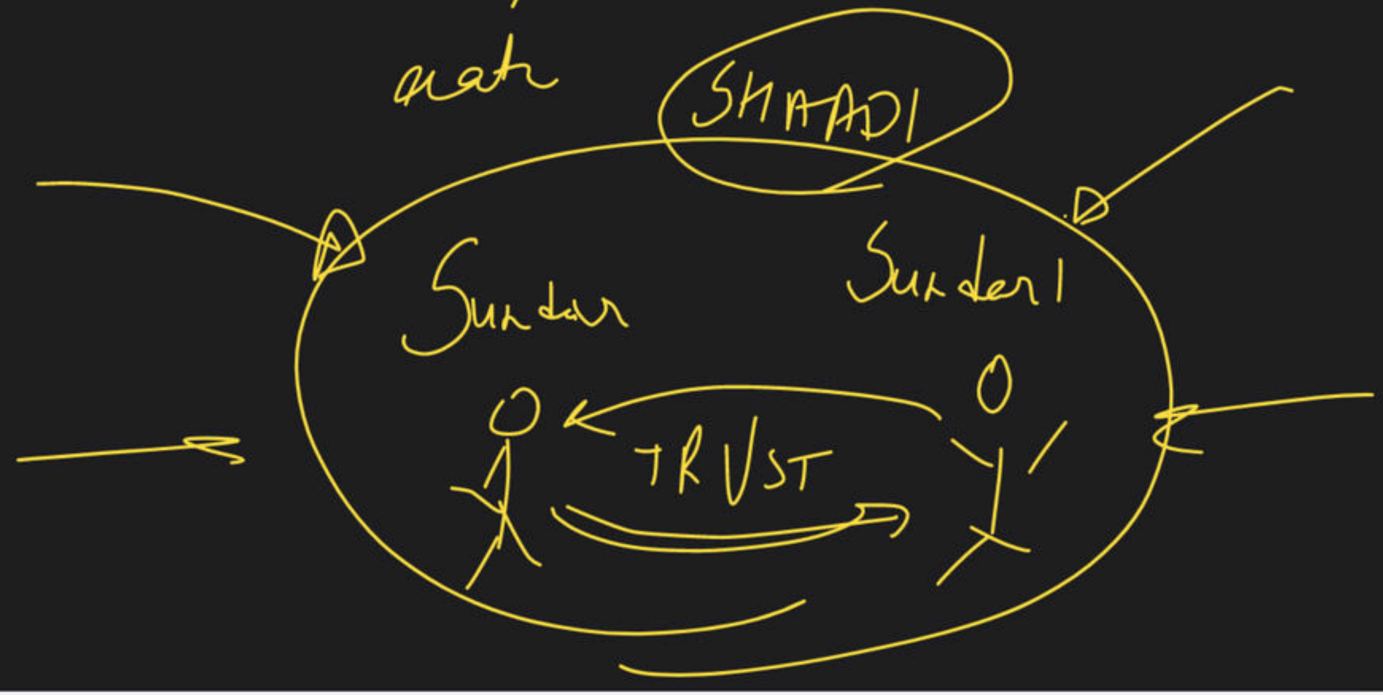

pow(4) → n = 4
if (n == 0)
return 1;
int ans = 2 *
pow(n-1);
return ans;
pow(3) → n = 3
if (n == 0)
return 1;
int ans = 2 *
pow(n-1);
return ans;
pow(2) → n = 2
if (n == 0)
return 1;
int ans = 2 *
pow(n-1);
return ans;
pow(1) → n = 1
if (n == 0)
return 1;
int ans = 2 *
pow(n-1);
return ans;
pow(0) → n = 0
main
if (n == 0) return 1;
int ans = 2 * pow(n-1)
return ans;

Call Shell

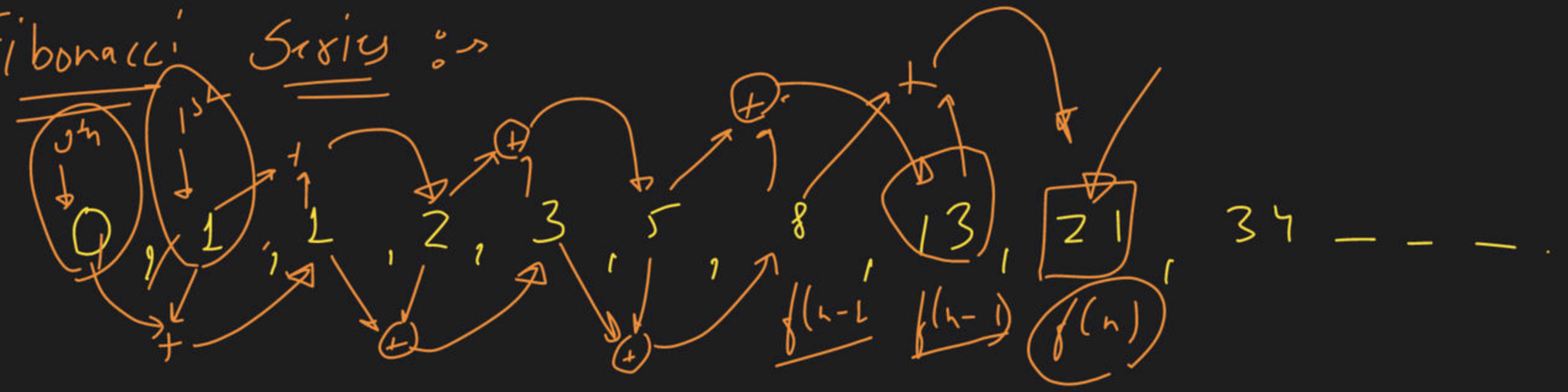

main()
pow(4)
pow(3)
pow(2)
pow(1)
pow(0)
Log dekh kya hai
Recursion Tree
Code Editor
T.C & S.C
1 case mein Solve Krunga &
Baaki Recursion Sambhal lega

To understand Recursion, you need to

first understand Recursion



→ Fibonacci Series :-



find n^{th} term

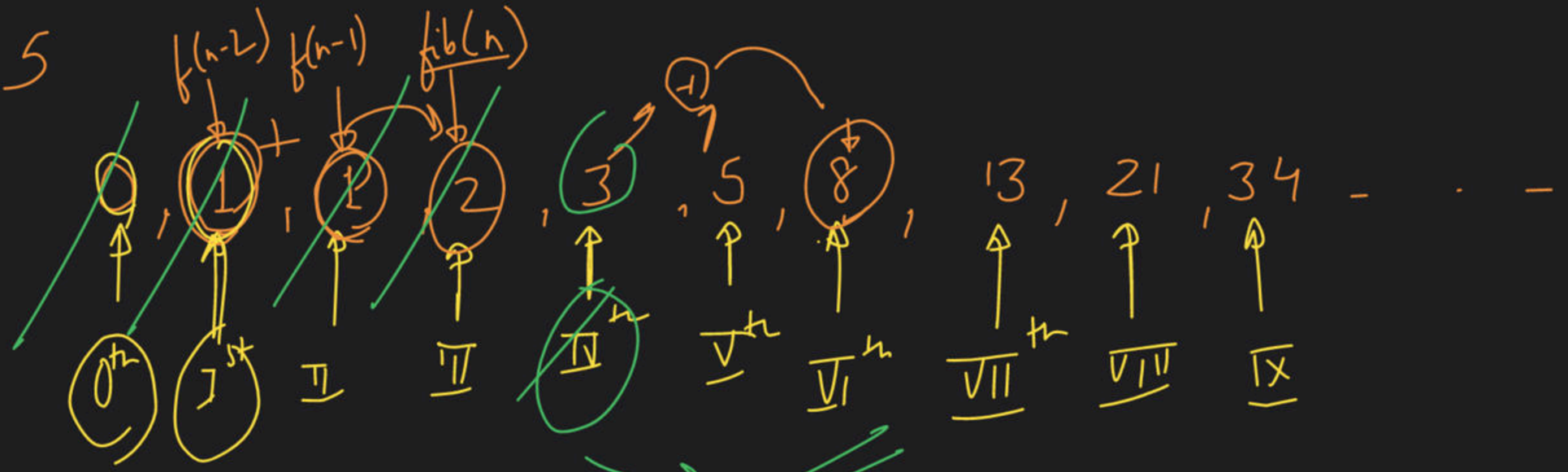
$$f(n) = f(n-1) + f(n-2)$$

Rec. relation

$$f(n) = f(n-1) + f(n-2)$$

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

E.5



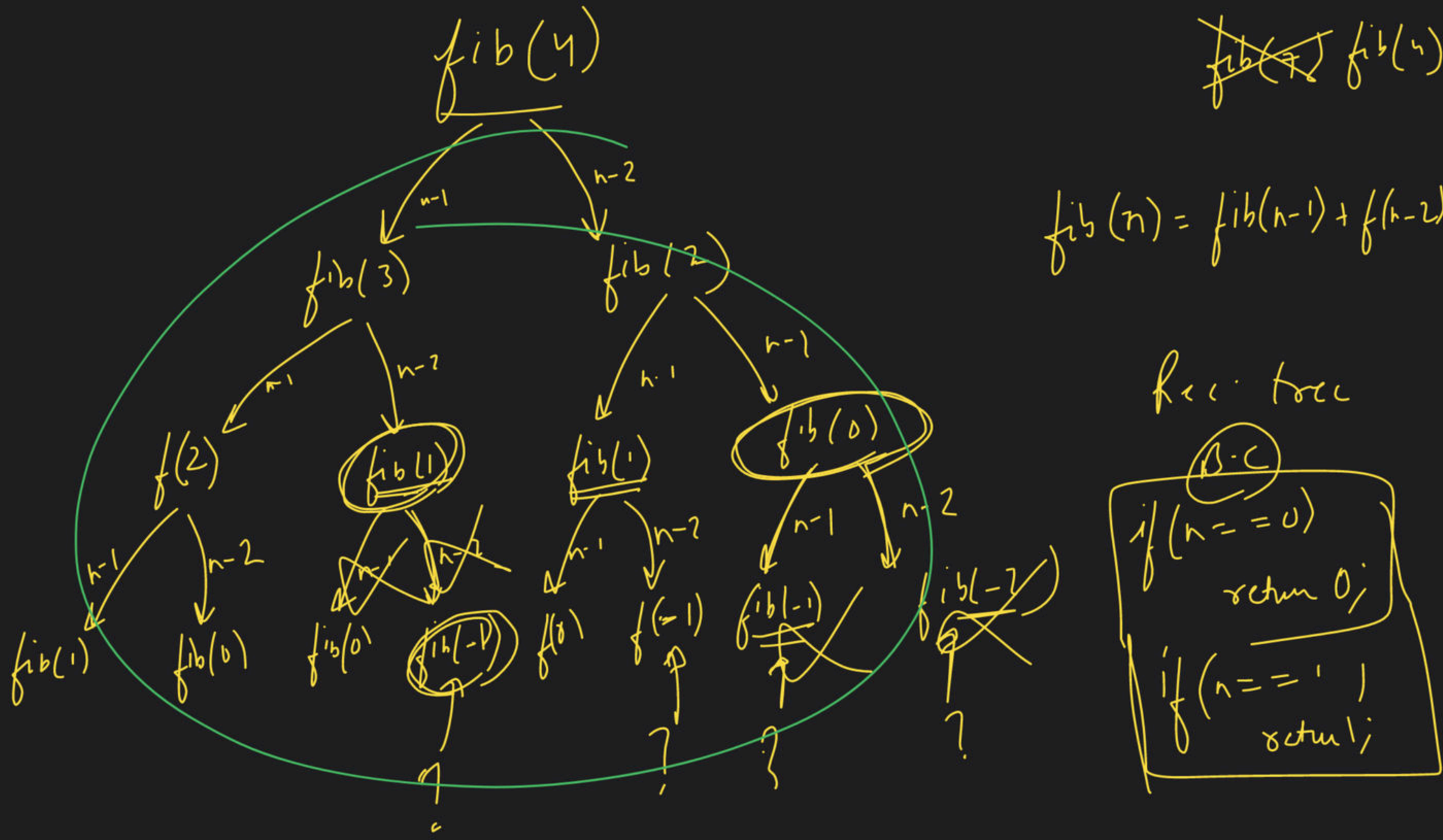
find n^{th} term → $f(n)$

B.C

$$fib(n) = fib(n-1) + fib(n-2)$$

$fib(7)$

f

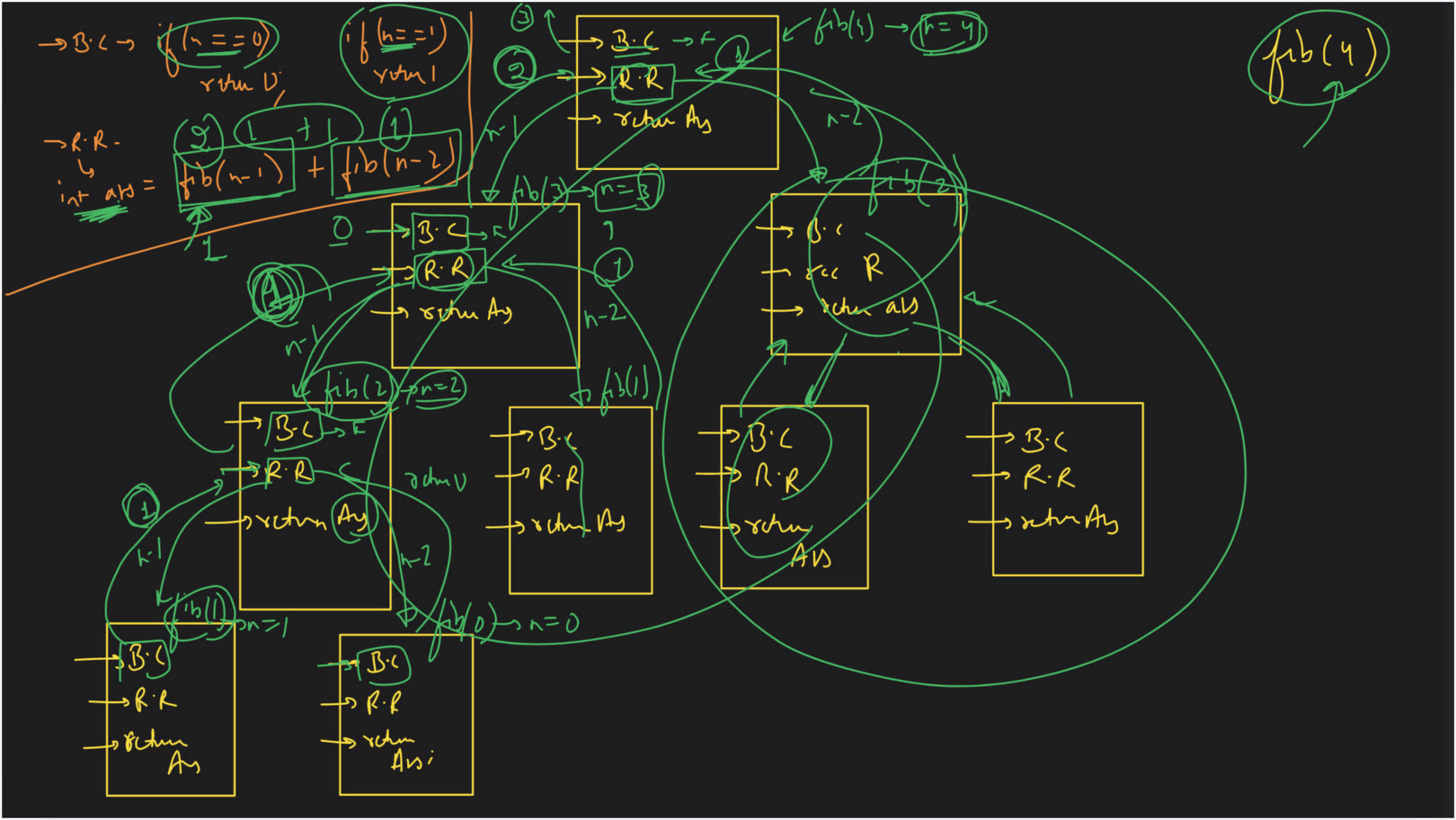


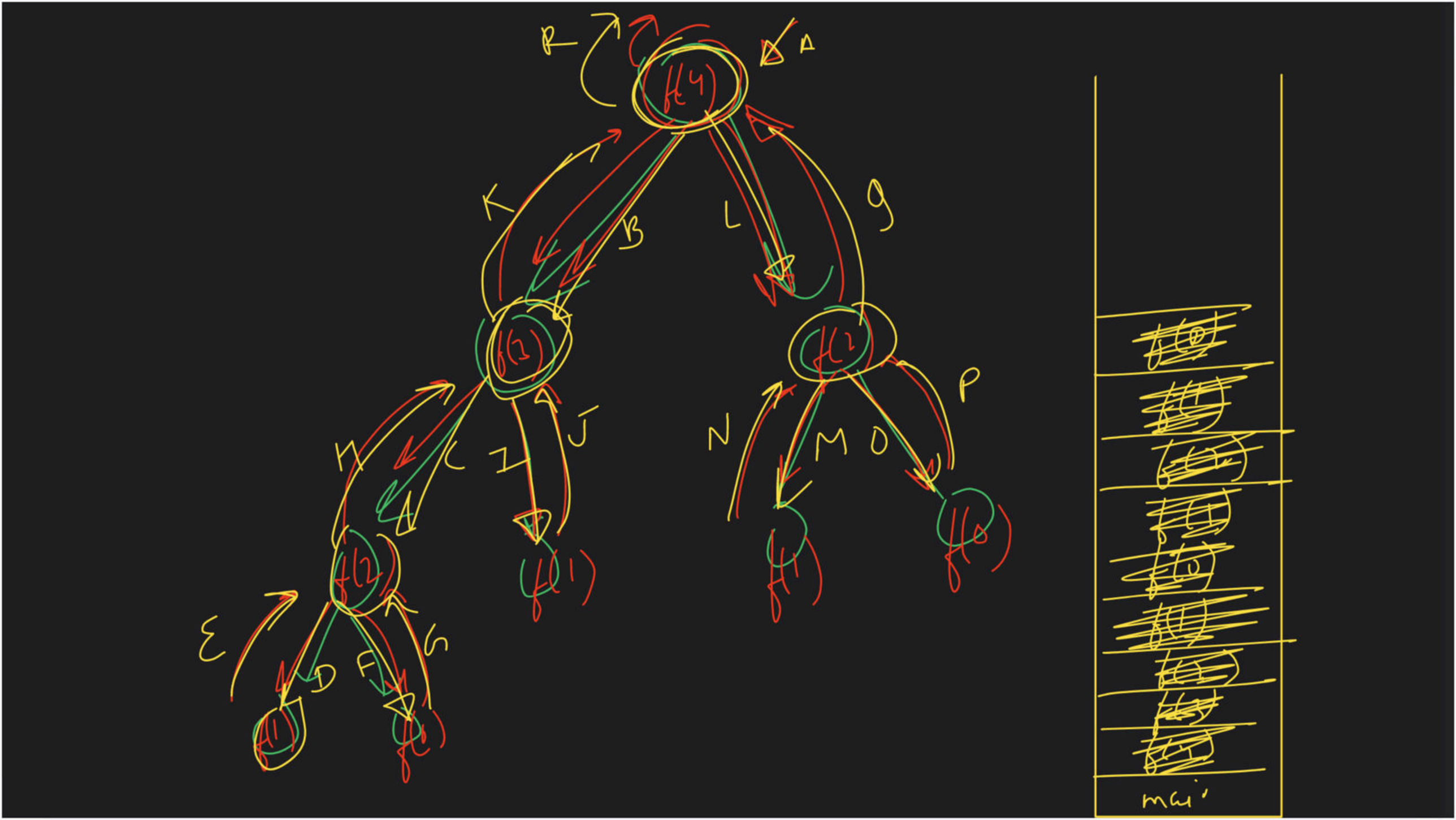
~~fib(-1)~~ fib(4)

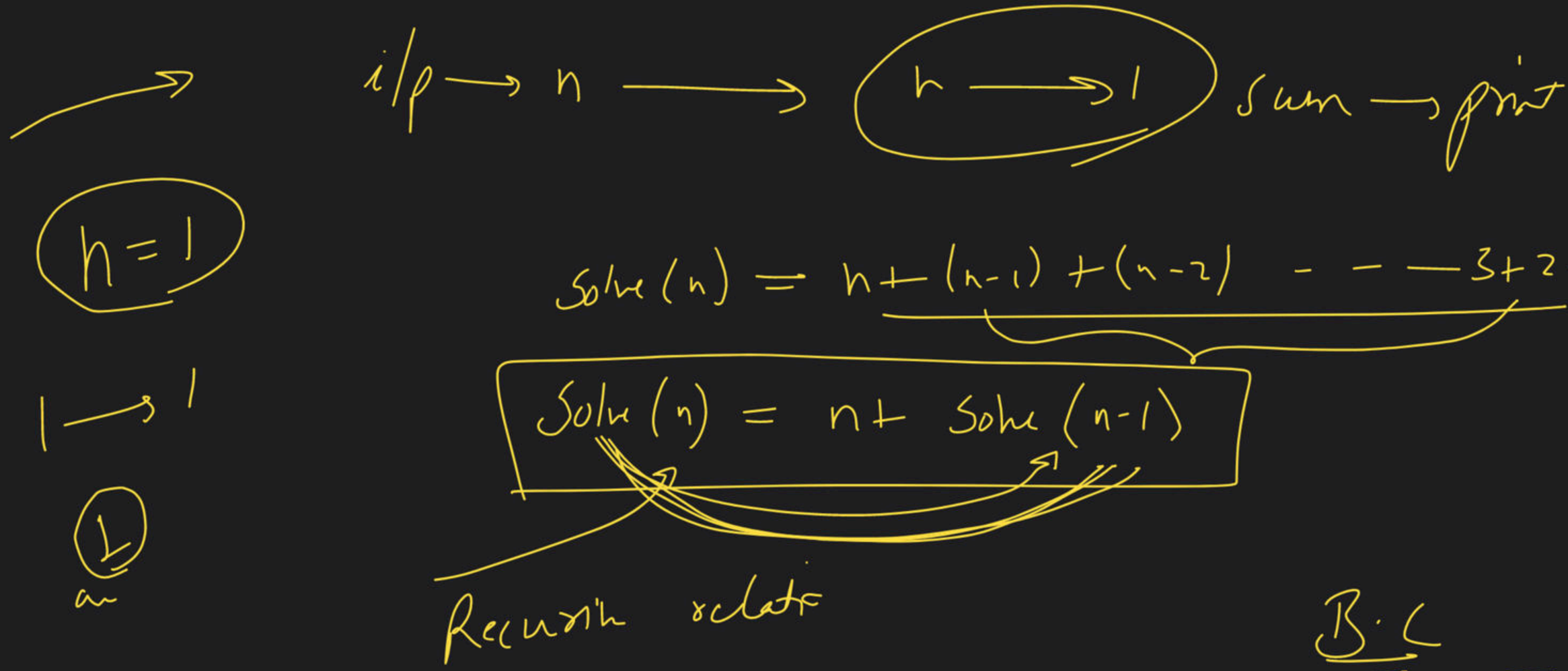
$$fib(n) = fib(n-1) + f(n-2)$$

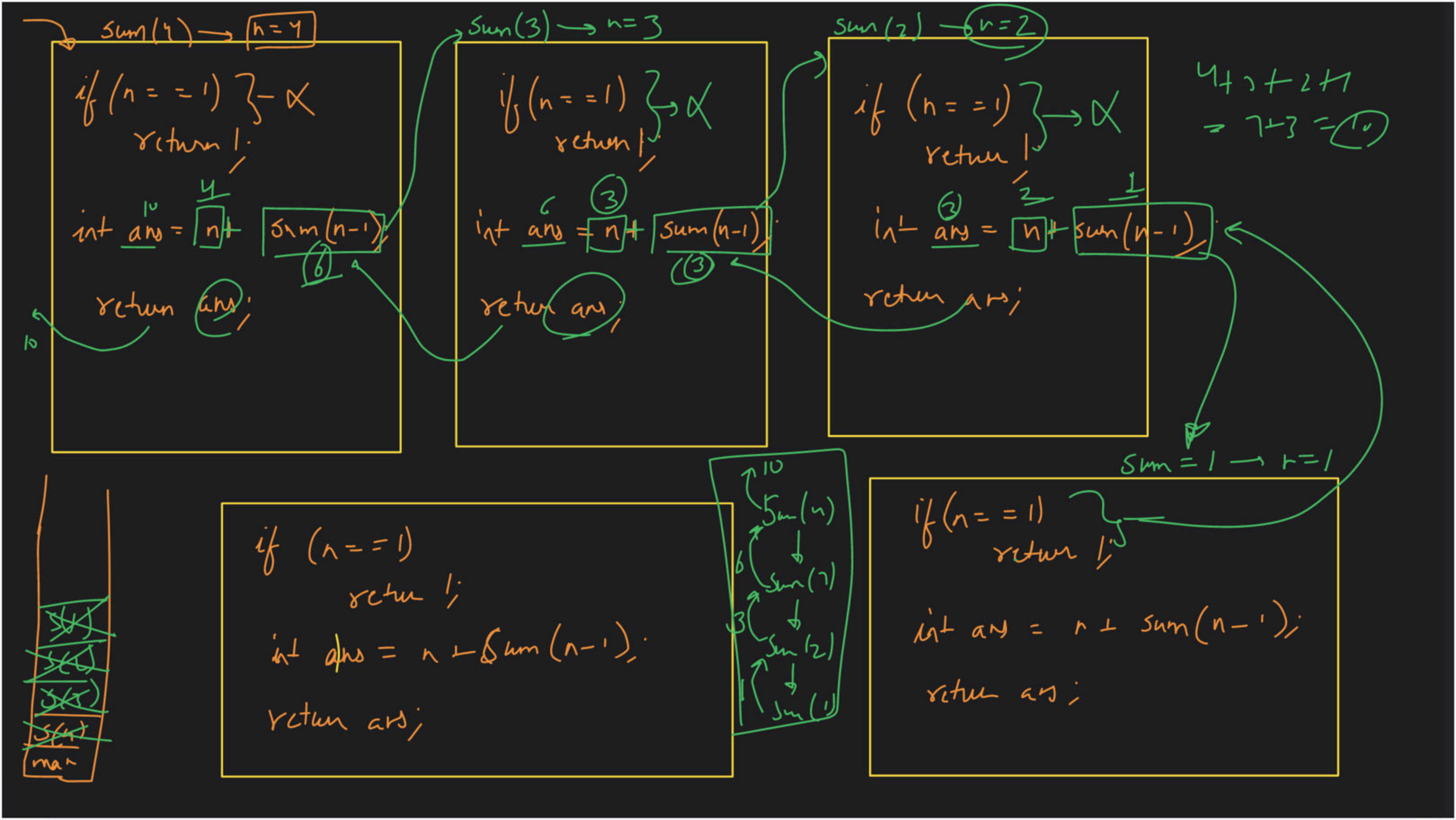
Recursion

```
B.C
if(n==0)
    return 0;
if(n==1)
    return 1;
```


B.C
if (n == 0)
return 0
if (n == 1)
return 1
R.R
int ans =
fib(n-1) + fib(n-2)
return ans
fib(4)
fib(3)
fib(2)
fib(1)
fib(0)
n=4
n=3
n=2
n=1
n=0
n-1
n-2
return 0
return 1
return ans


R
A
K
B
L
g
H
I
J
N
M
O
P
E
D
S
main


i/p → n
n → 1
sum → print
n=1
1 → 1
1
Solve(n) = n + (n-1) + (n-2) - - - 3+2+1
Solve(n) = n + Solve(n-1)
Recursive relate
B.C
if (n==1)
return 1


sum(4) → n=4
if (n==1) → X
return 1;
int ans = n + sum(n-1);
return ans;
sum(3) → n=3
if (n==1) → X
return 1;
int ans = n + sum(n-1);
return ans;
sum(2) → n=2
if (n==1) → X
return 1;
int ans = n + sum(n-1);
return ans;
4+3+2+1
= 7+3 = 10
sum=1 → n=1
if (n==1)
return 1;
int ans = n + sum(n-1);
return ans;
if (n==1)
return 1;
int ans = n + sum(n-1);
return ans;
sum(4)
sum(3)
sum(2)
sum(1)
main
10
6
3
1