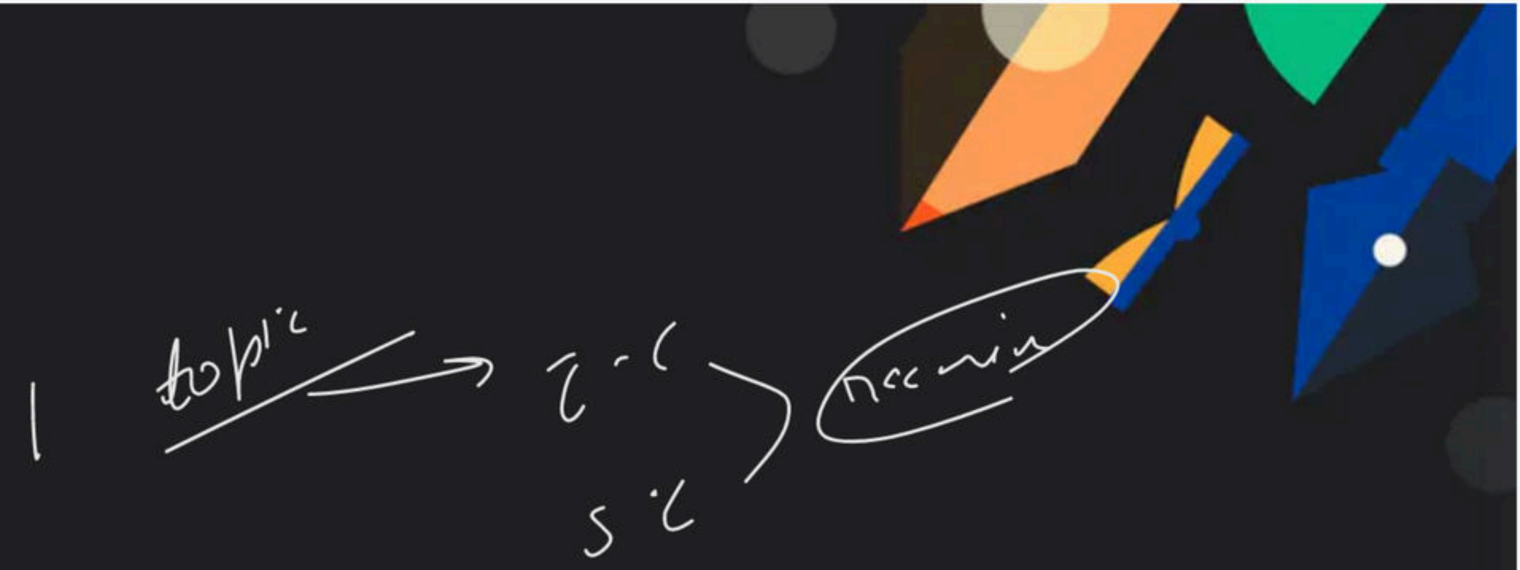


OOPs - I

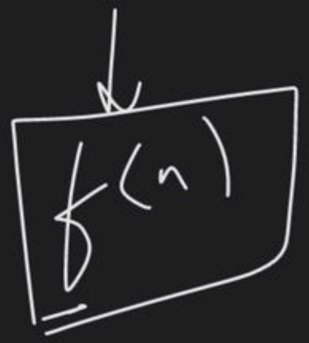
Foundation Course on Data Structures & Algorithm - Part I

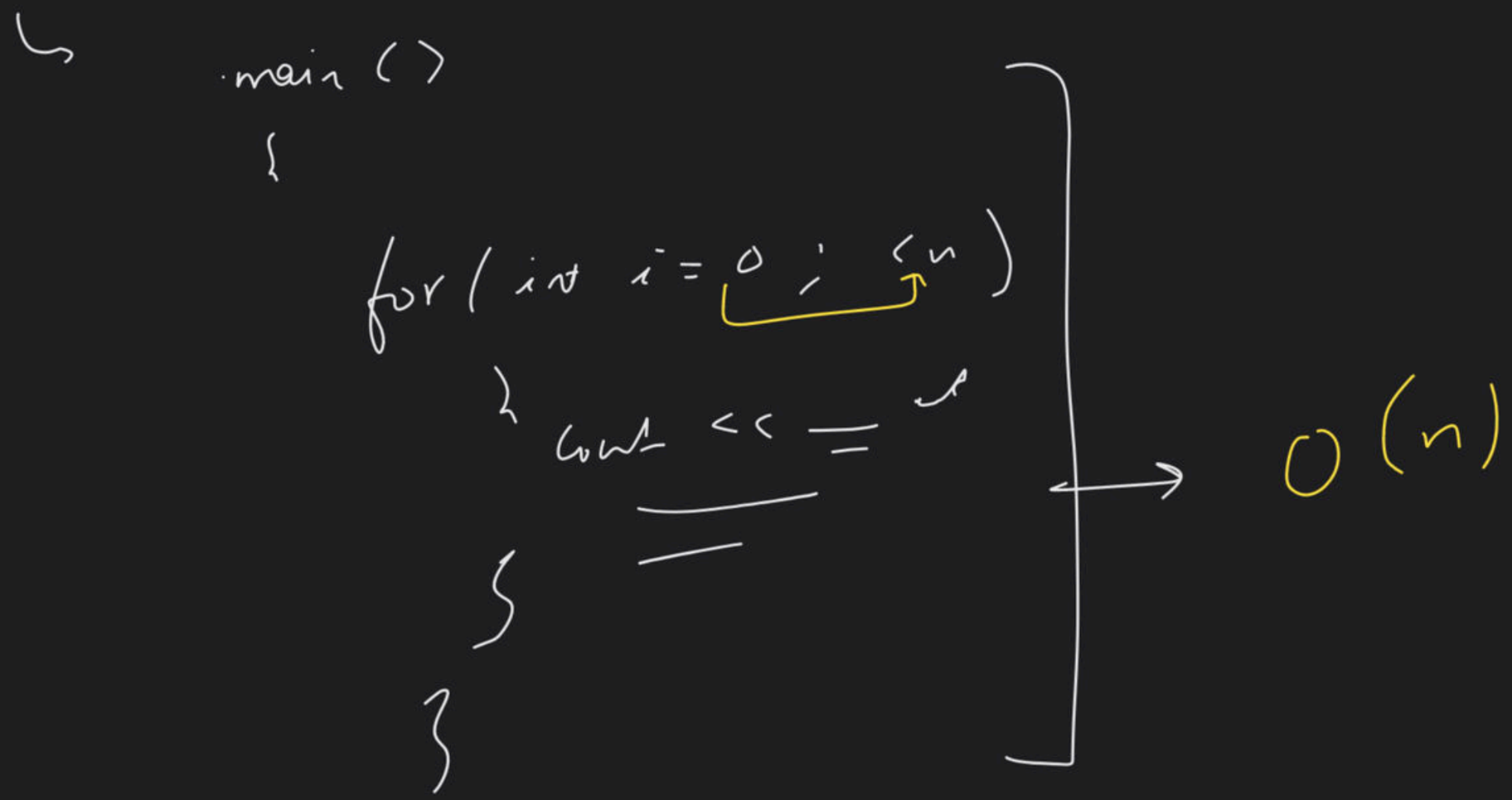
Love Babbar • Lesson 26 • June 9, 2022

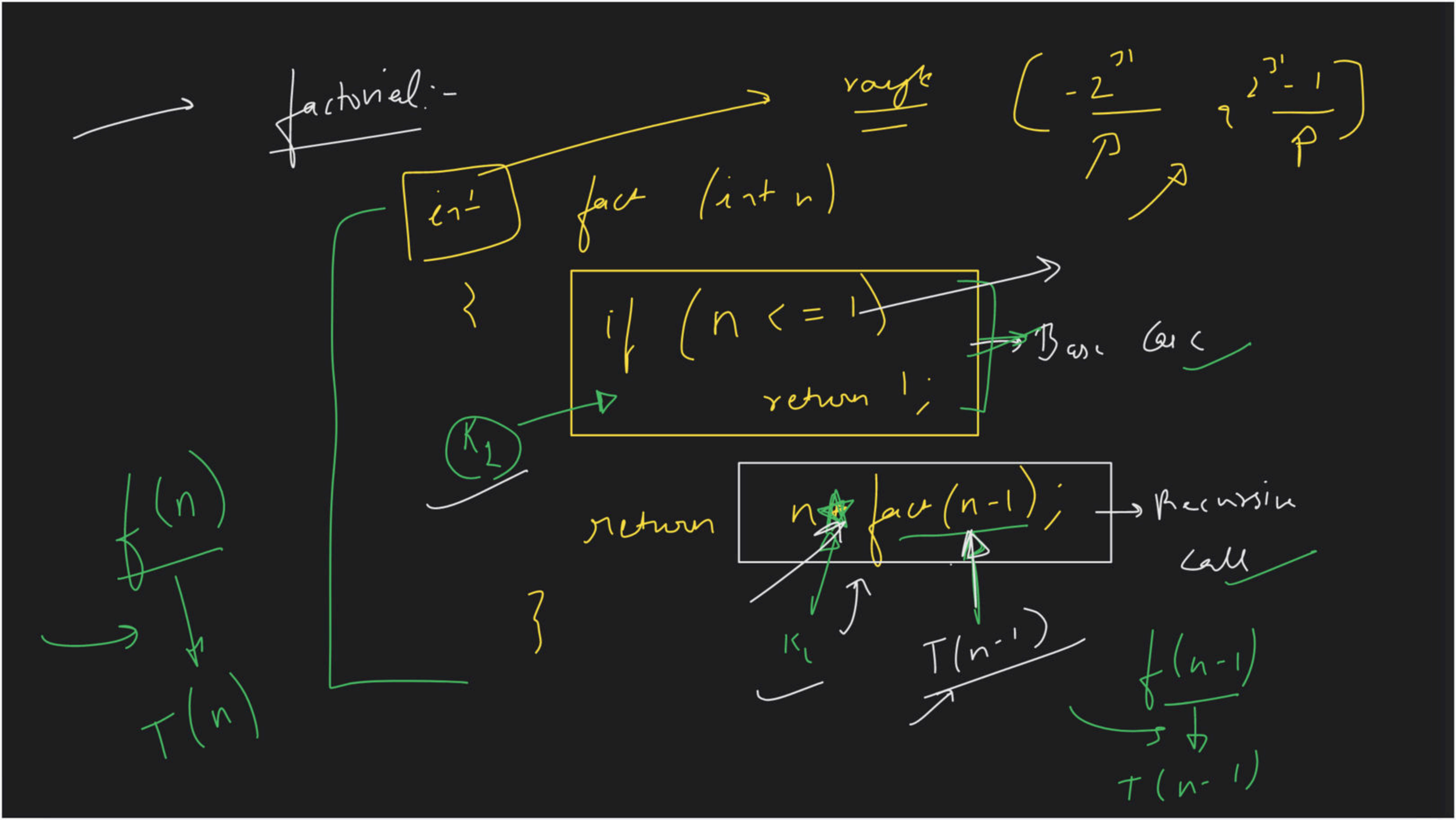
Recursion:-

factorial → T.C → ?

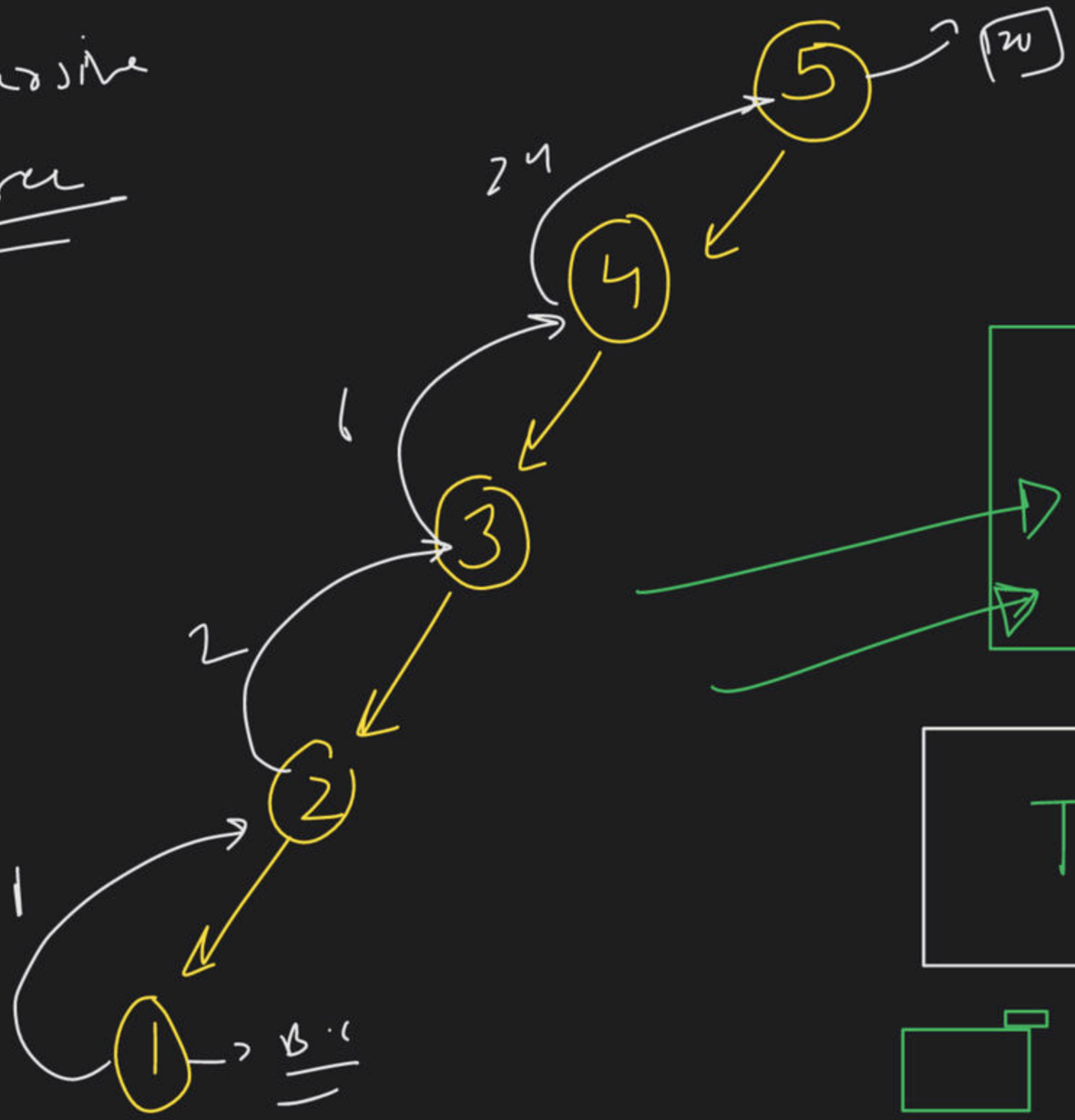
Time Complexity → [total time required as function of input] → repeat




main()
{
for(int i=0; i<n)
cout<<
}
O(n)


factorial :-
range
$\left[-\frac{2^{n}}{p}, \frac{2^{n}-1}{p}\right]$
int
fact (int n)
{
if (n <= 1)
return 1;
Base Case
K_1
return
n * fact(n-1);
Recursive call
}
f(n)
T(n)
K_1
T(n-1)
f(n-1)
T(n-1)

Recursive tree

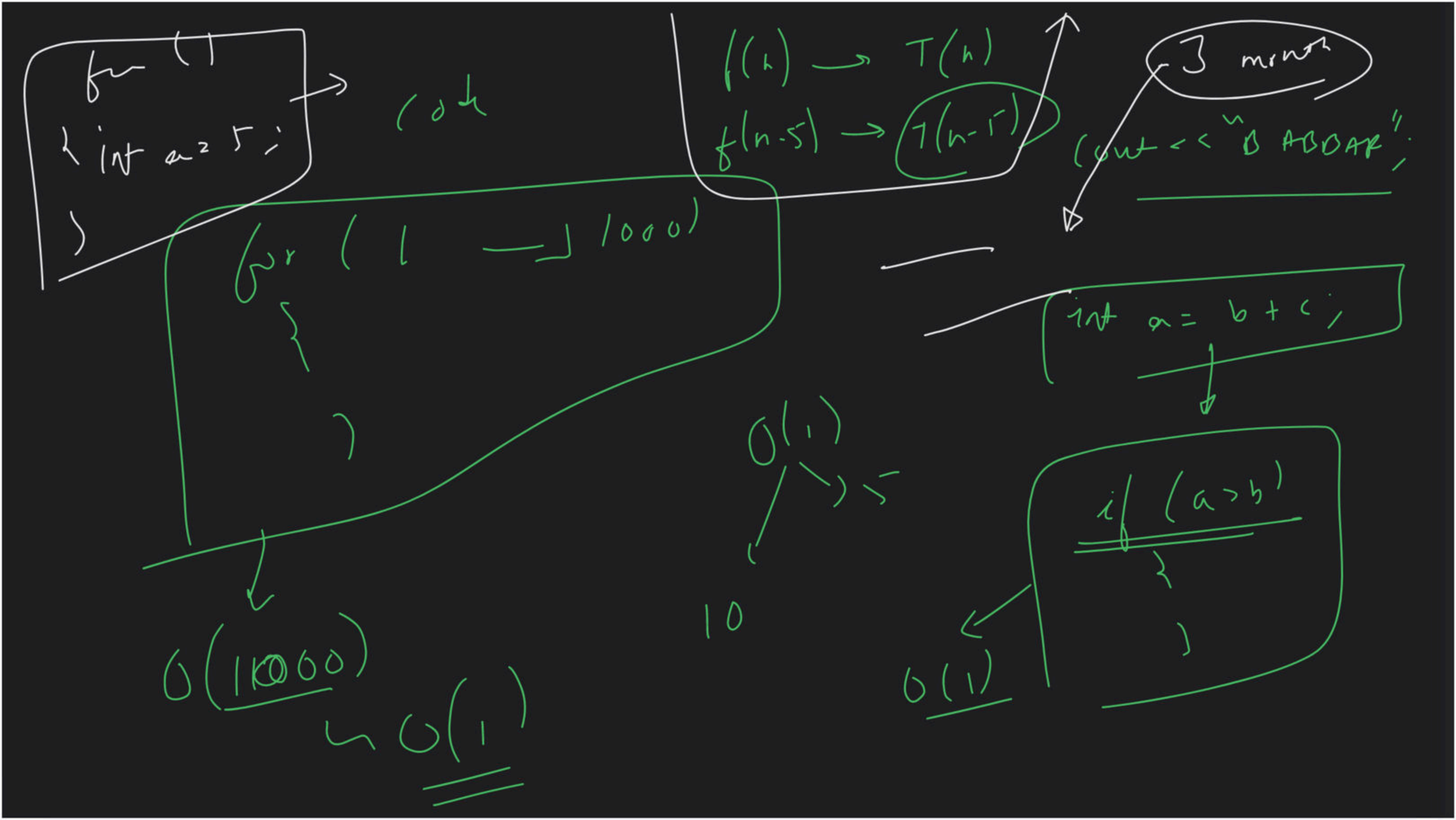


R.R:-

$$f(n) = n * f(n-1)$$

$$T(n) = K_1 + K_2 + T(n-1)$$

?


fun ()
{ int a = 5;
}
code
f(n) → T(n)
f(n-5) → T(n-1)
3 mins
cout << "I ABDAR";
for (i → 1000)
{
}
int a = b + c;
O(1)
O(1000)
O(1)
10
if (a > b)
{
}
O(1)

doubt

Index

$T(n) = K_1 + K_2 + T(n-1)$

K

$T(n) = K + 1 + T(n-1)$

$T(n-1) = K + 1 + T(n-2)$

$T(n-2) = K + 1 + T(n-3)$

$T(2) = K + 1 + T(1)$

$T(1) = K$ → worst

B.C

$T(n) = n * K$

n

(n-1)

$T(n) = (n-1)K + K_1$

$T(n) = nK - K + K_1$

$T(n) = nK + K_3$

O

nK

$O(n)$

or

$(n)^2 n^3 + \ldots$

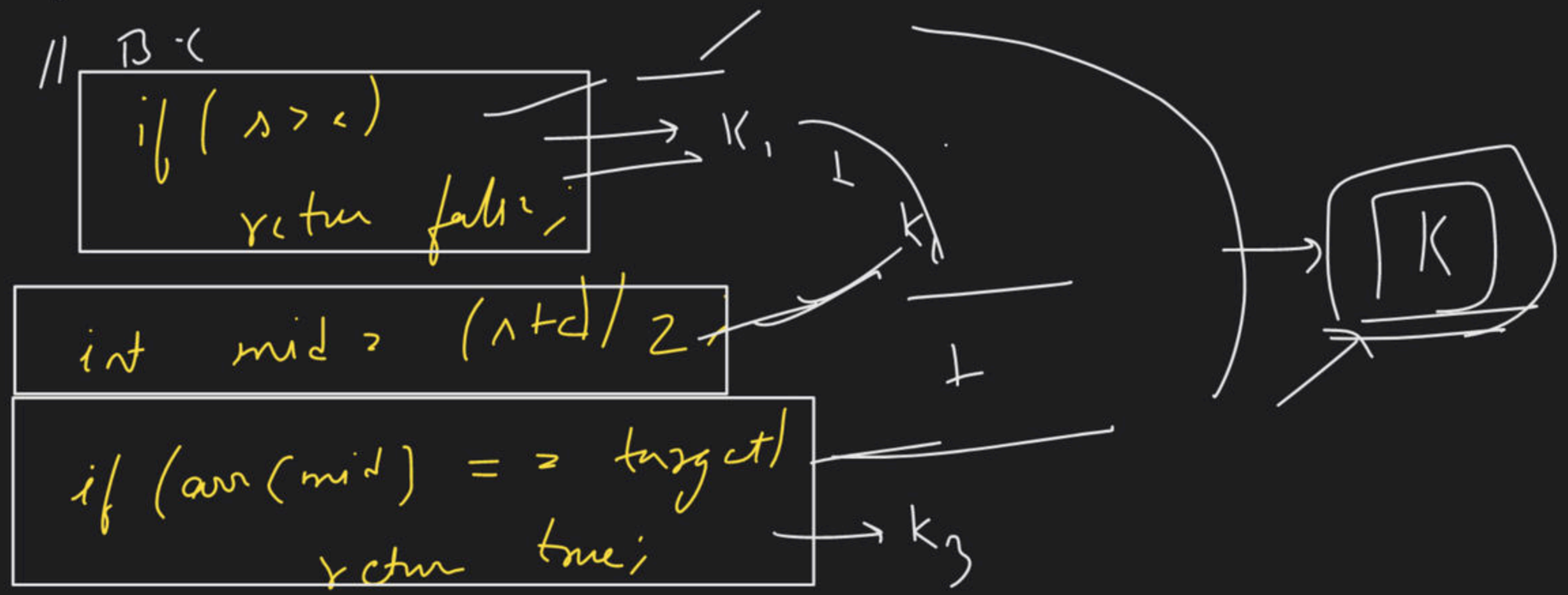
$O(n^3)$

$(n)^2 \beta n^2$

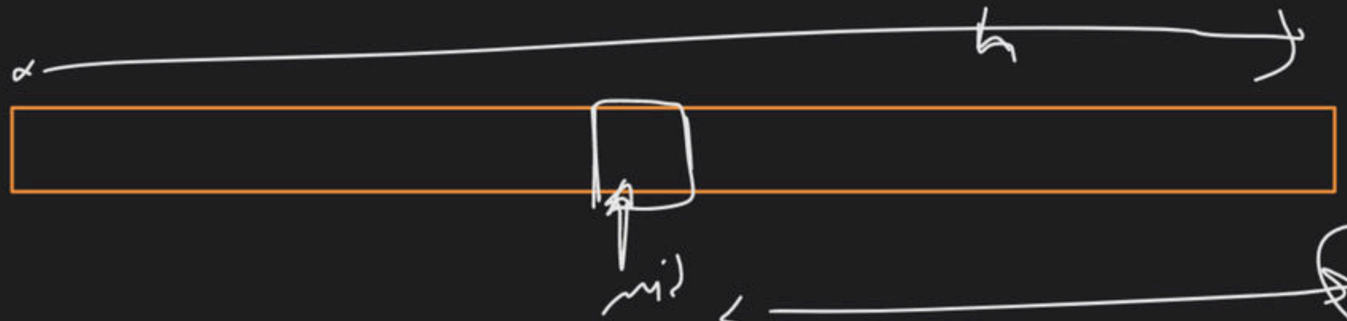
$O(n^2)$

Binary Search:-

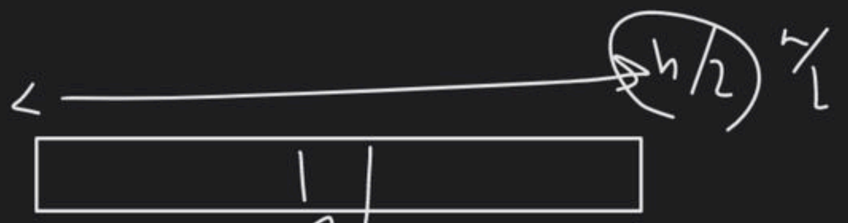
```
bool fun ( int arr[], int s, int e, int target )
{
// B.C
    if ( s > e )
        return false;

    int mid = (s+e)/2;

    if (arr[mid] == target)
        return true;
```




```
if ( )
    return fun ( arr, s, mid, target );
else
    return fun ( arr, mid+1, e, target );
}
```



n



$n/2$

$$f(n) = K + f\left(\frac{n}{2}\right)$$

$n/2$

$\frac{n}{2^1}$

$n/8$

$\frac{n}{2^2}$

2

1

$\frac{n}{2^K} = 1$

$n = 2^K$

$\log n = K$

$\frac{n}{2^K} = 1$

$K = \log n$

$T(n) = K_1 + K_2 + K_3 + T\left(\frac{n}{2}\right) \longrightarrow$ $\frac{n}{2}$, $\frac{n}{2}$, ... $\log n$

K

why $\rightarrow \log$

$T(n) = a * K$

$\doteq \log n * 1$

$O(\log n)$

n, $n-1$, $n-2$, $n-3$, ..., 1

$T(n) = K + T\left(\frac{n}{2}\right)$

$T\left(\frac{n}{2}\right) = K + T\left(\frac{n}{4}\right)$

$T\left(\frac{n}{4}\right) = K + T\left(\frac{n}{8}\right)$

$T(2) = K + T(1)$

$T(1) = K$

a

$a = b + c$

$b = c + d$

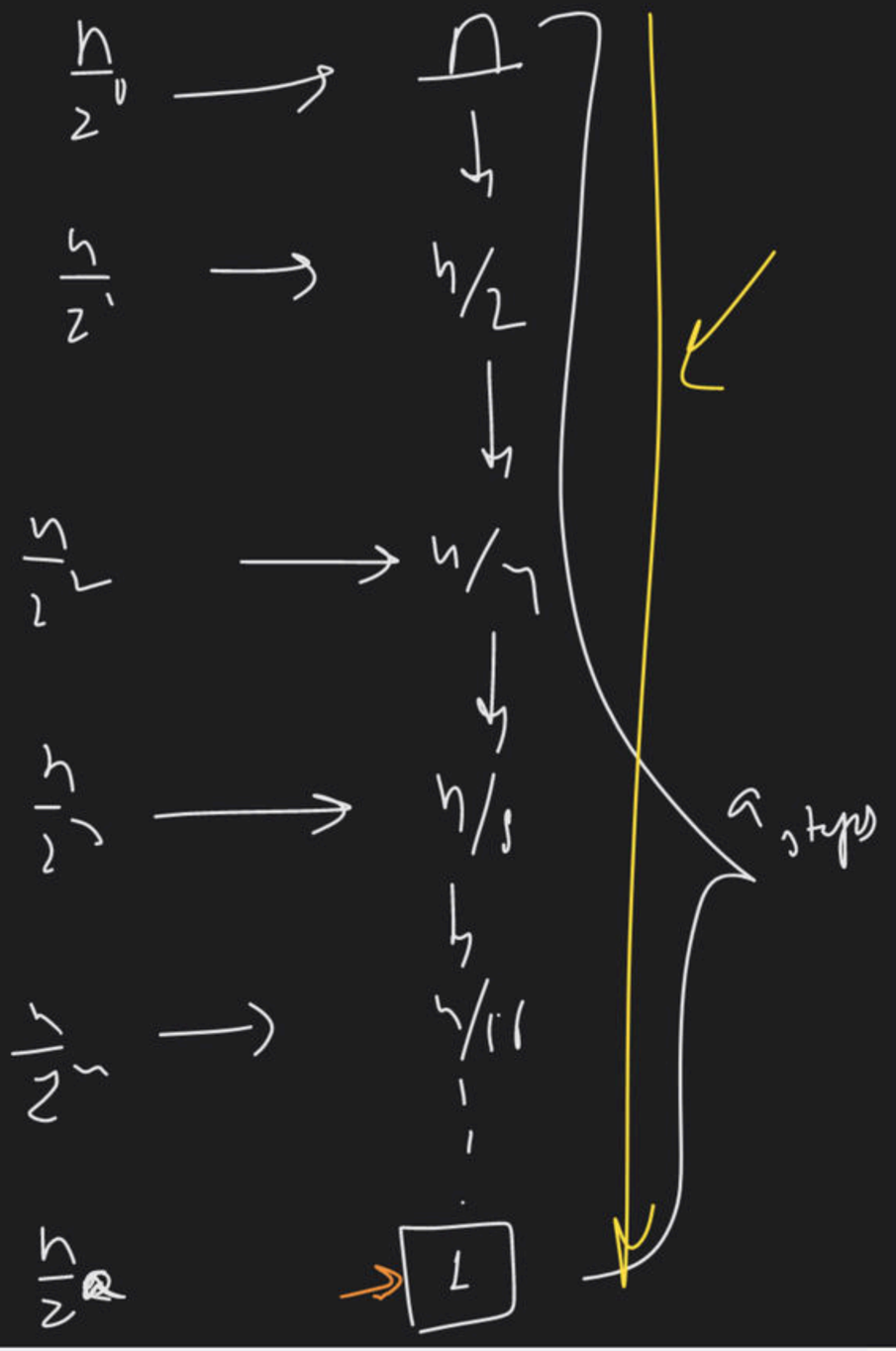
$a + b = b + c + c + d$

$\frac{n}{2^a} = 1$

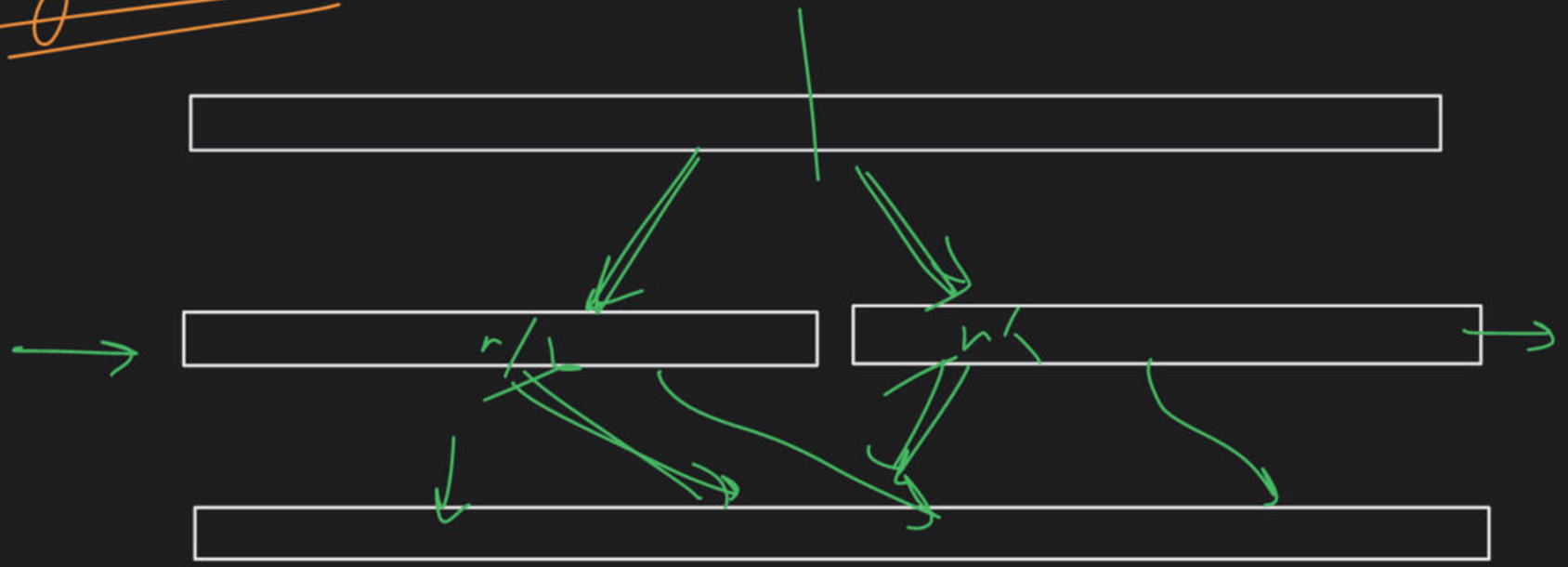
$n = 2^a$

$a = \log n$

harmonic



Merge Sort →



Steps:-

→ left / Right

→ Recursion → left
→ right

→ merge

mergeSort( ) ← T(n)

{

// B.C → K_1

// mid → K_2

→ K

leftSort( ) → $T(n/2)$

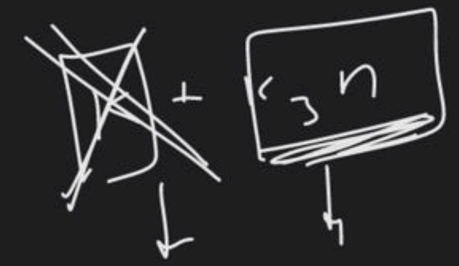
rightSort( ) → $T(n/2)$

merge( ) → $K_3 n$

}

$$T(n) = K + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + K_3 n$$

$$T(n) = 2T\left(\frac{n}{2}\right) + K_5 n$$



$2^1 \times \left[T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + K_5 \frac{n}{2} \right]$

$4 \times \left[T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + K_5 \frac{n}{4} \right]$

δ

$T(1) = K$

$f(n) = n^2 + n$

$O(n^2)$

$$2 \times \left[T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + k_1 \frac{n}{2} \right]$$

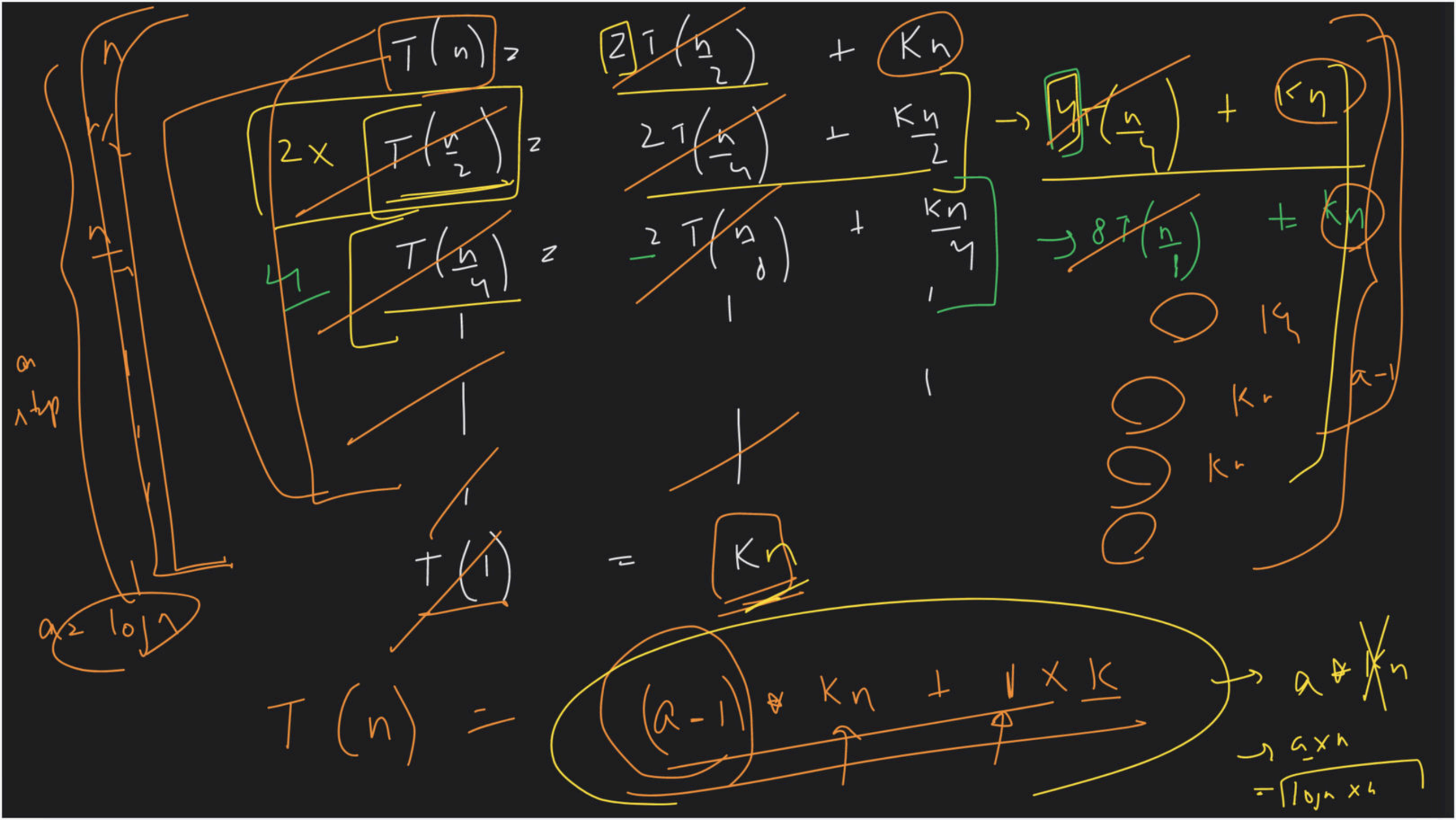
$$2T\left(\frac{n}{2}\right) = 4T\left(\frac{n}{4}\right) + k_1 n$$

$$T(n) = K_1 + K_2 + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + K_3 n$$

$K_1 + K_2 \rightarrow K_4$

$$T(n) = K_4 + 2T\left(\frac{n}{2}\right) + K_3 n$$

$$T(n) = 2T\left(\frac{n}{2}\right) + K_3 n$$


T(n) = 2T(n/2) + Kn
2x T(n/2) = 2T(n/2) + Kn
T(n/4) = 2T(n/8) + Kn/4
→ 4T(n/4) + 2Kn
→ 8T(n/8) + 3Kn
T(1) = Kn
a
height
a = log n
Kn
Kn
Kn
a-1
T(n) = (a-1) * Kn + 1 x Kn
→ a * n
→ a x n
= log n x n

$T(n) = a * k * n \rightarrow kn + k$

$a = \log_2 n$

$= a kn - k(n+1)$

$= k * n \log_2 n - k(n+1)$

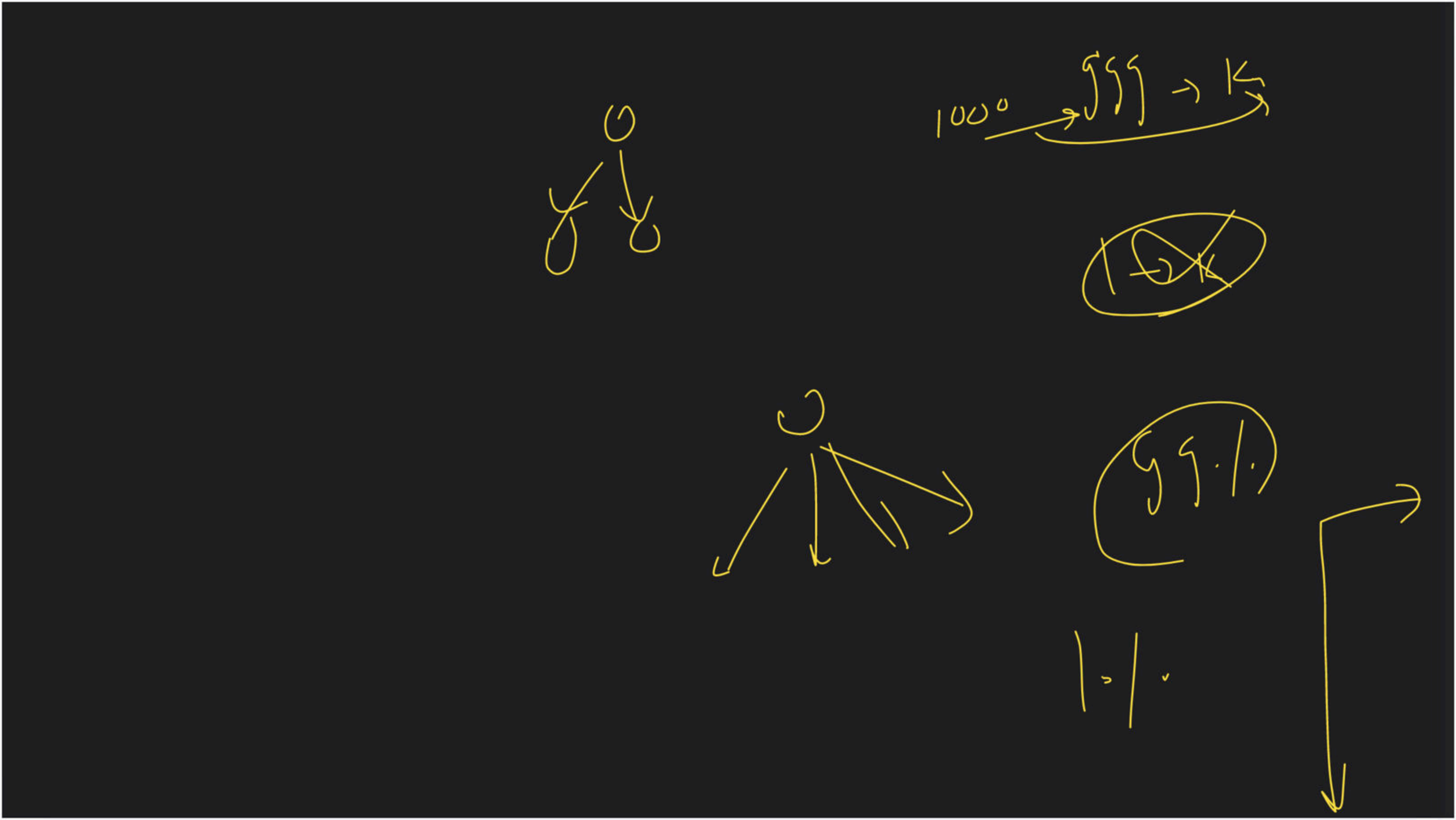
$O(n \log_2 n)$

$n^2 + n$

$O(n^2$

$n \log_2 n + n$

$O(n$



Fibonacci Series

```
int fib(int n)
{
  // Base case
  if (n == 0 || n == 1)
    return n;

  return fib(n-1) + fib(n-2);
}
```

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

Relation

$T(n) = T(n-1) + T(n-2)$ How

$T(n-1) = T(n-2) + T(n-3)$

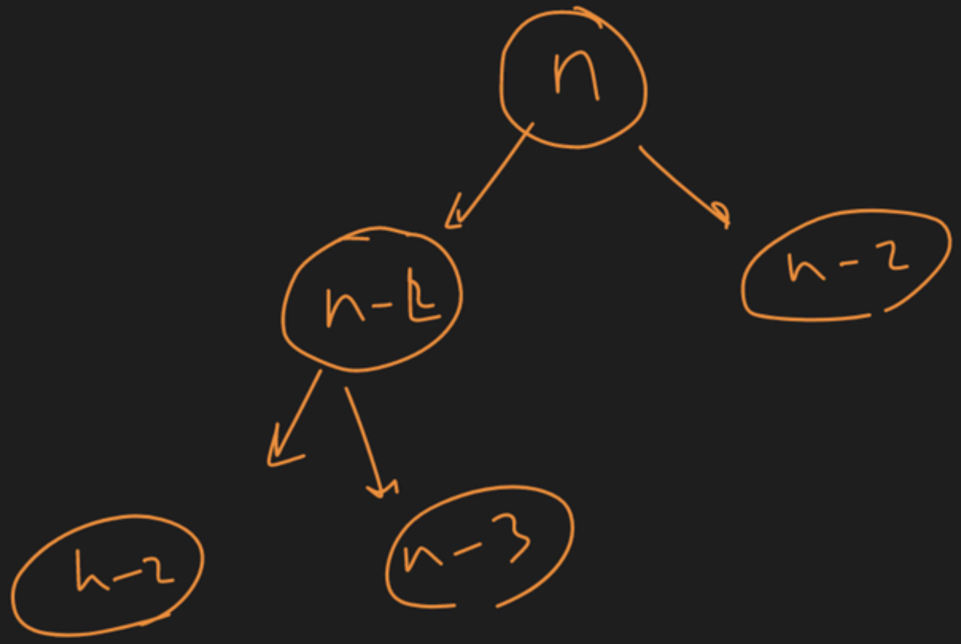
$T(n-2) = T(n-3) + T(n-4)$

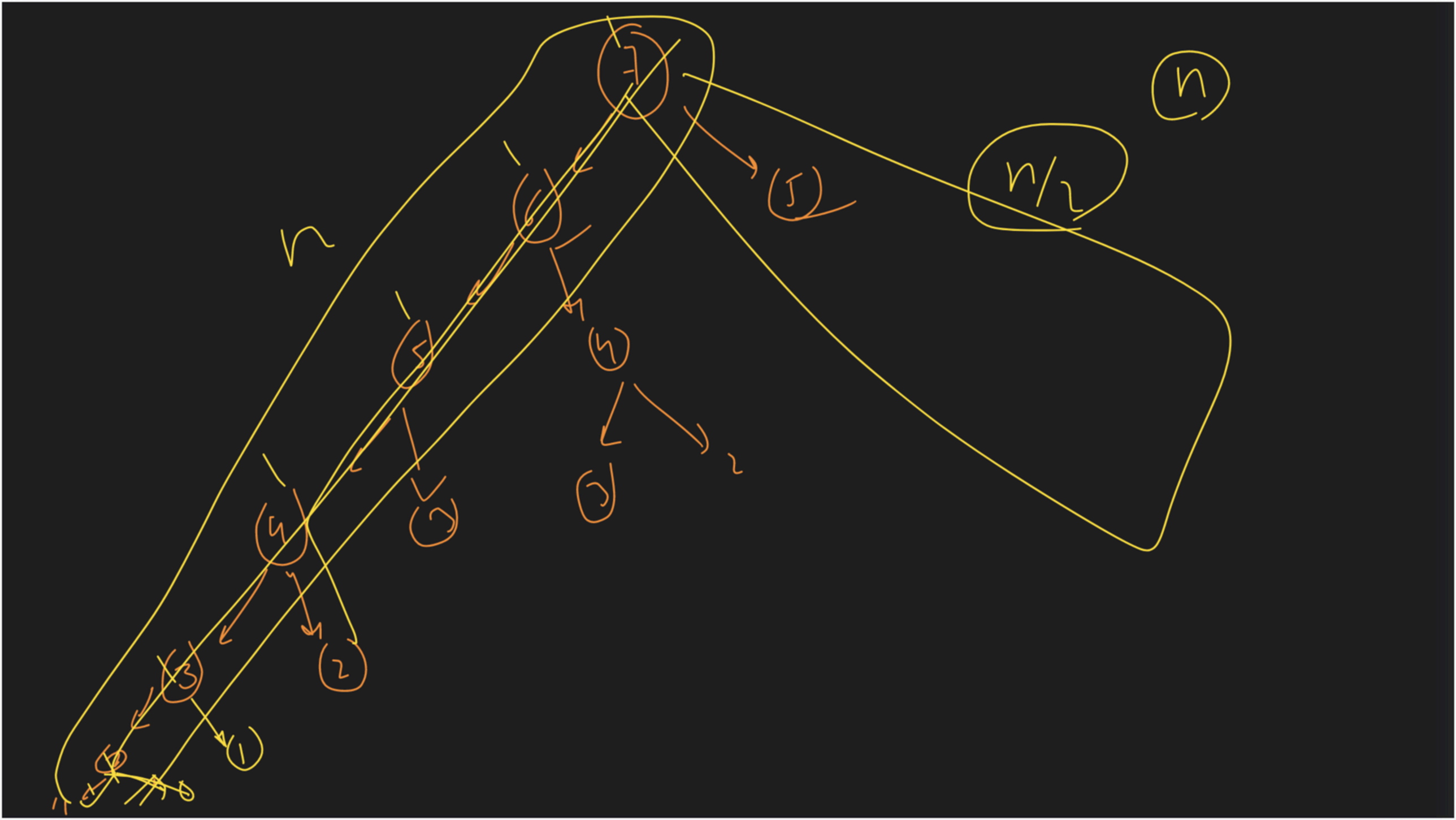
$\vdots$

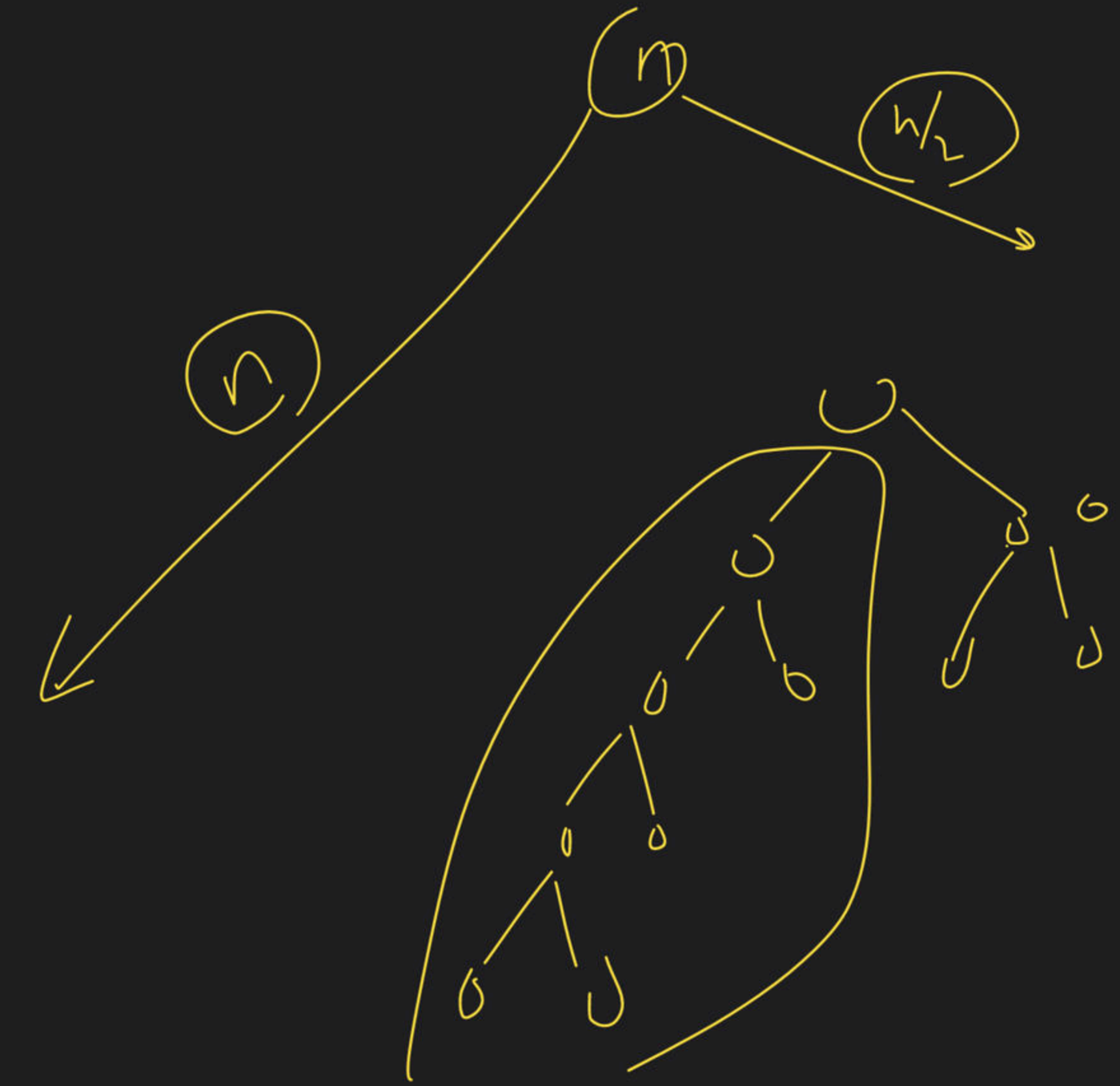
$T(2) = T(1) + T(0)$

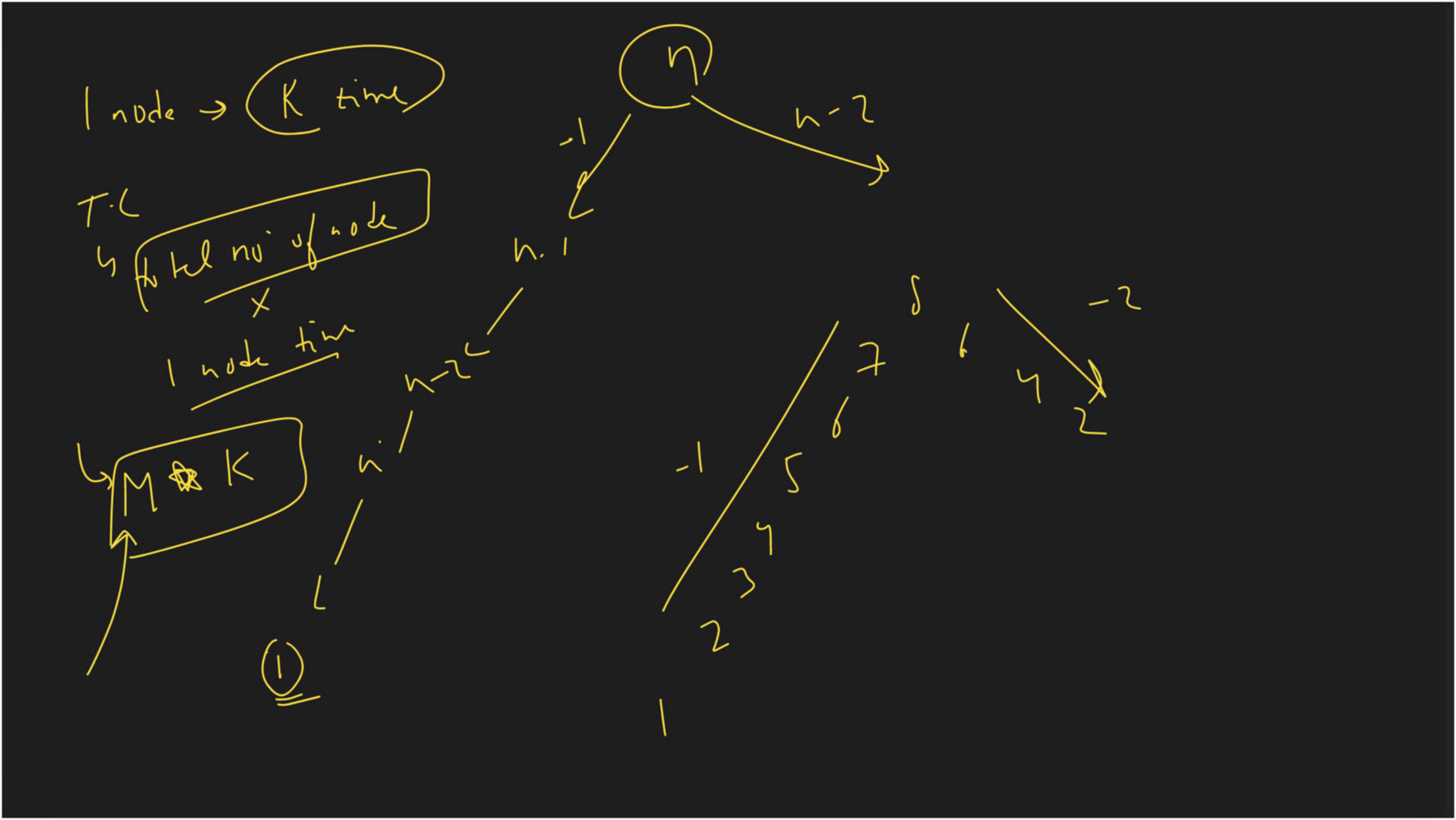
$T(1) = K$

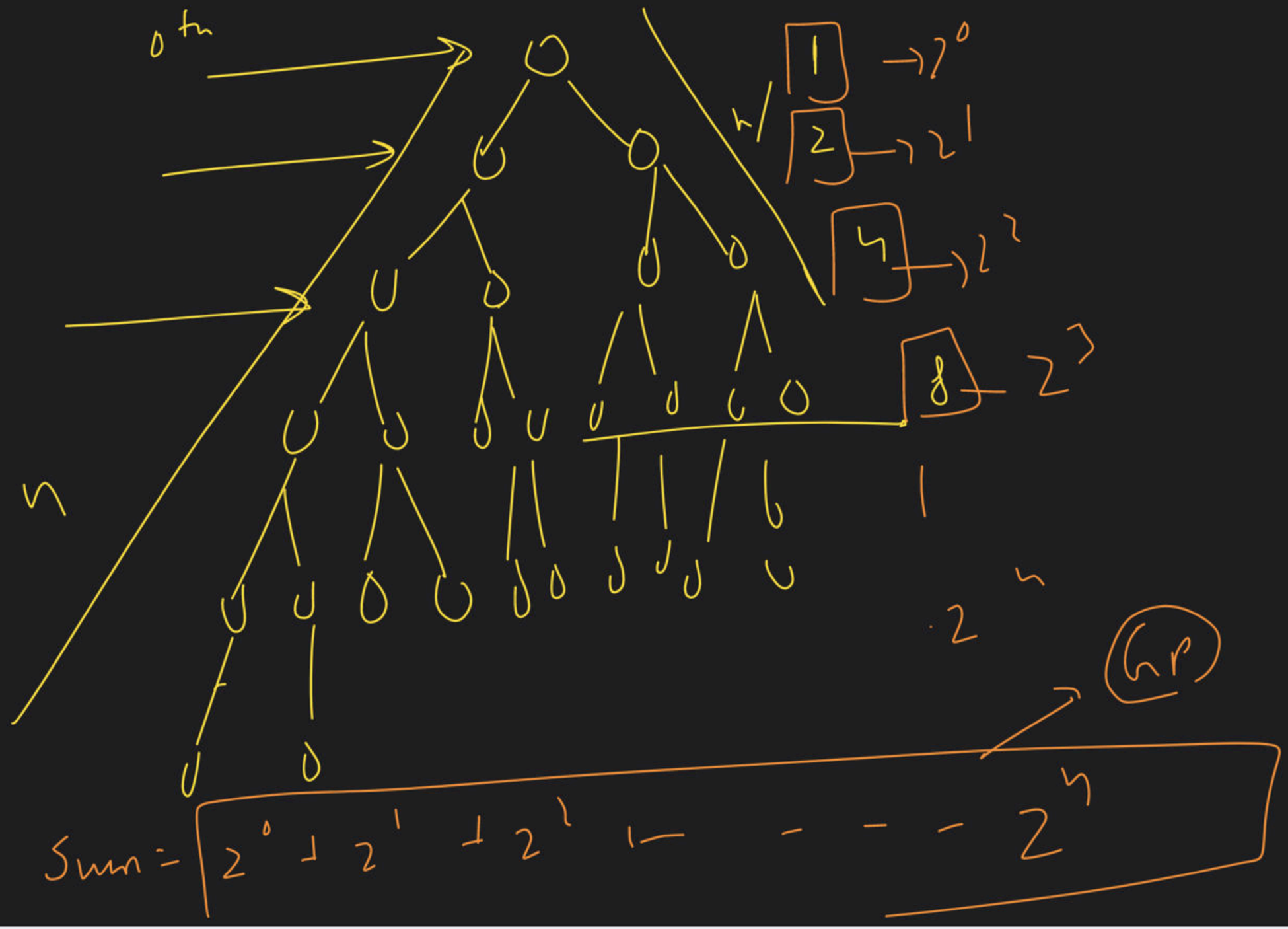
$T(0) = K$


n
n-1
n-2
n-2
n-3


n
n/2
n


m
w/2
n


1 node → K time
T.C
total no of node
X
1 node time
M * K
n
n-1
n-2
1
2
3
4
5
6
7
5
6
4
2
-1
-2


0th
h
1
2
4
8
2^0
2^1
2^2
2^3
1
2^h
GP
Sum = $2^0 + 2^1 + 2^2 + \cdots - - - 2^h$

Total node = $2^{n+1} - 1$ → 2^{n+1} → $2^n \times 2$

(2^n)

T.C = 2^n

= $O(2^n)$

$O(3^n)$ → TLE

S.C →

factorial

```
int f(int n)
{
    if ( )
    return n;
    return n*f(n-1);
}
```

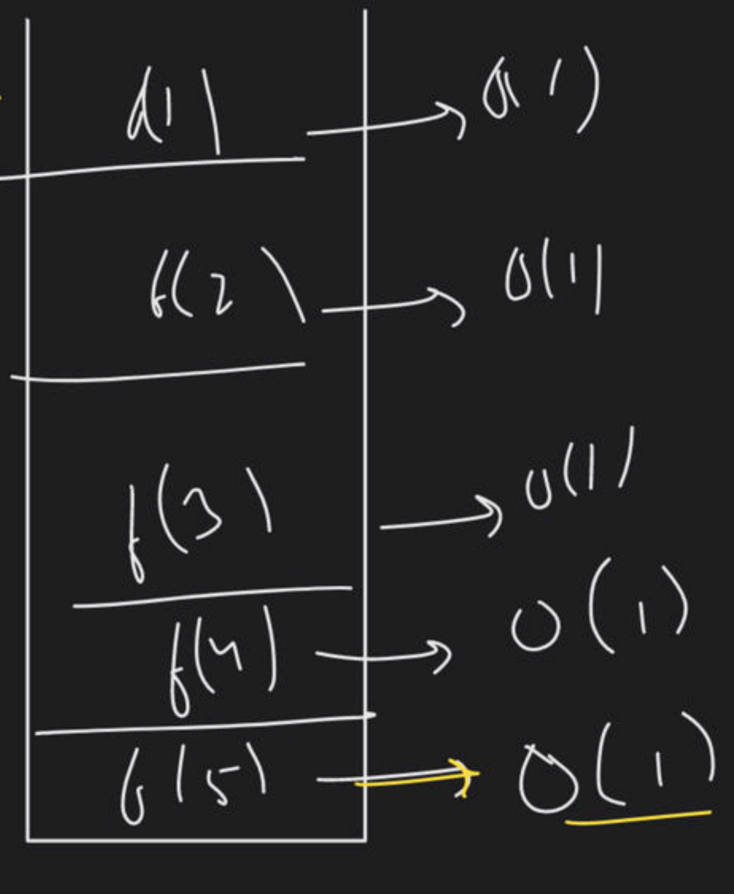
1 call → 10ms

f(5) n = 5

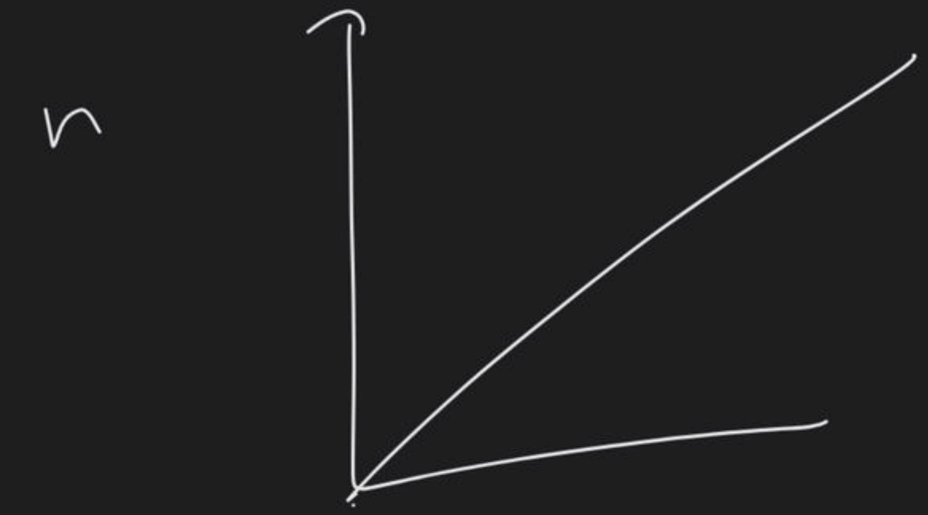
1 call → O(1)

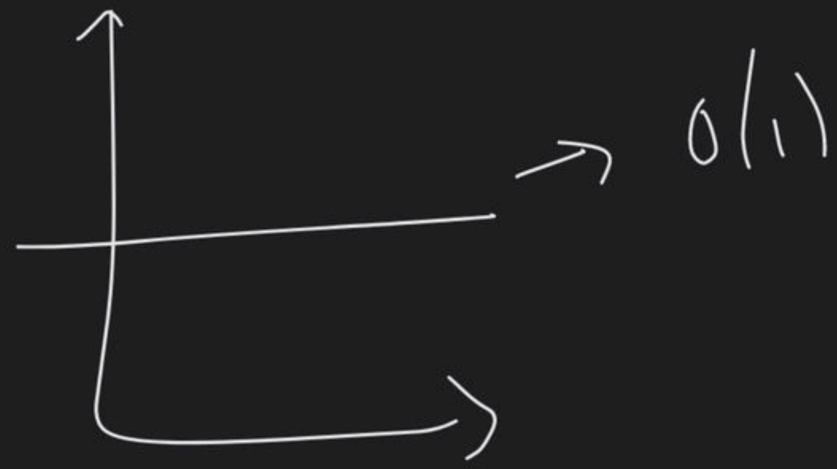
n call → n = O(1)

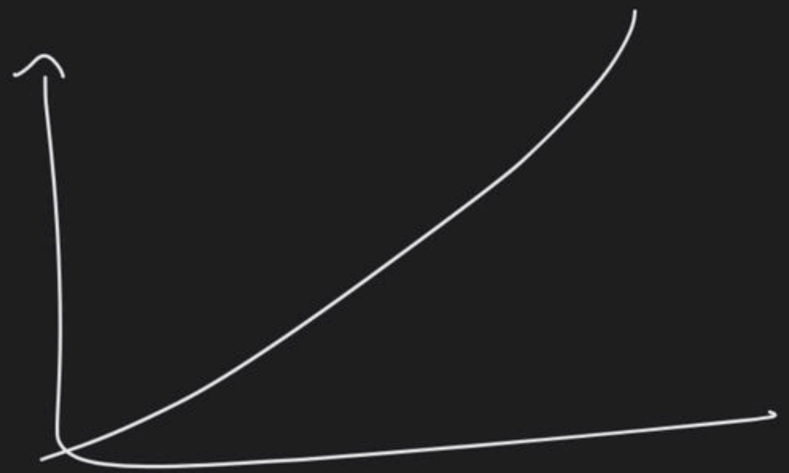
n calls

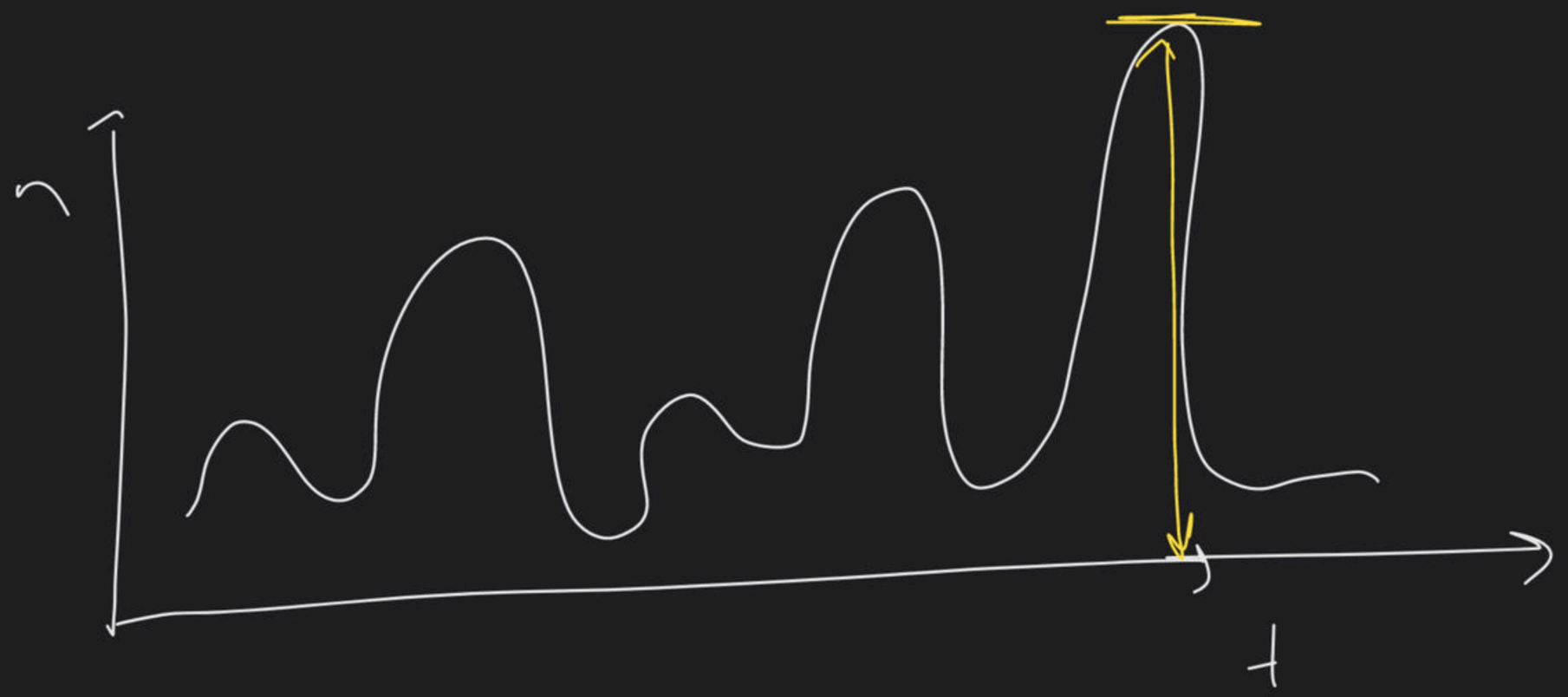


f() → O(n)


n

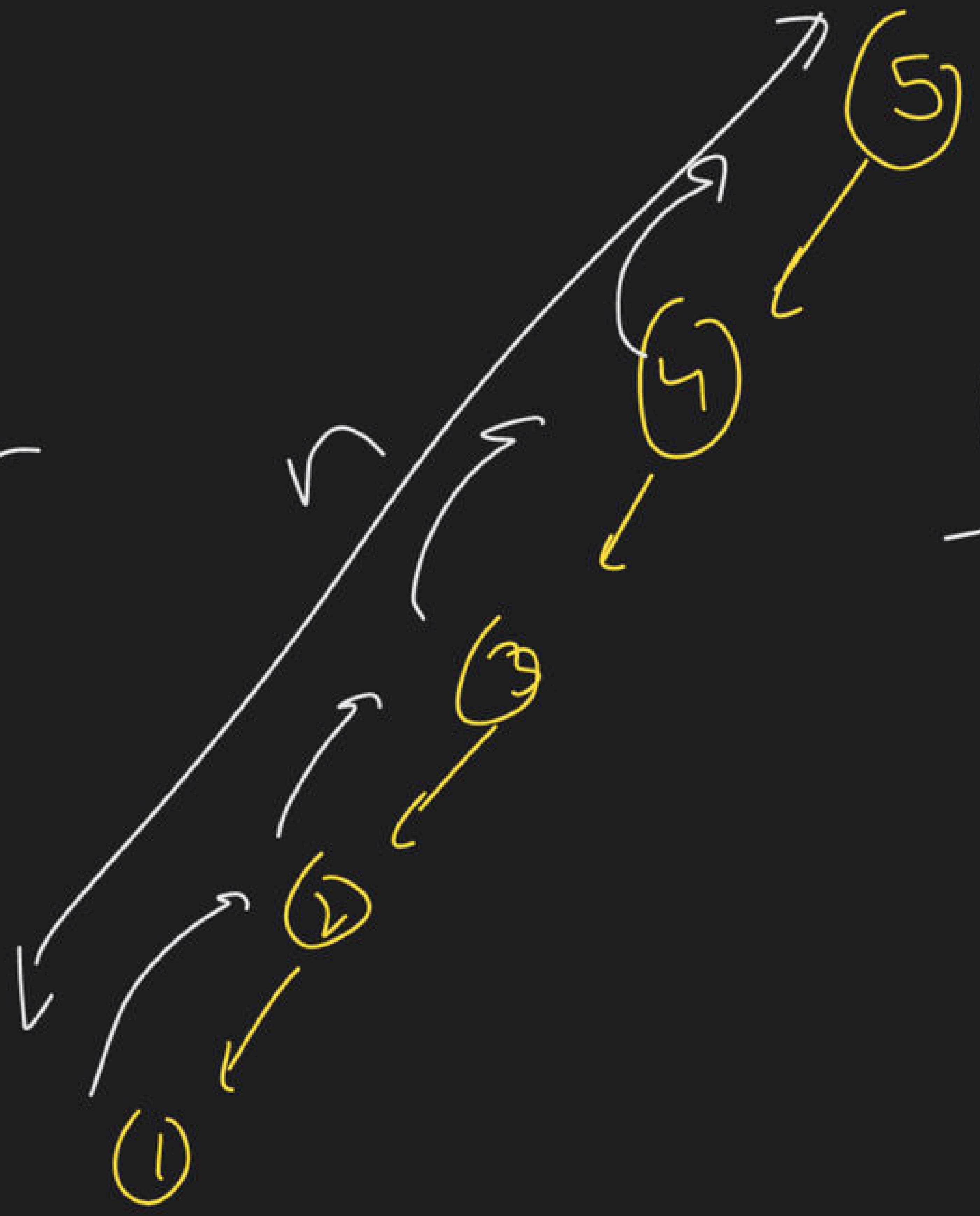

O(1)





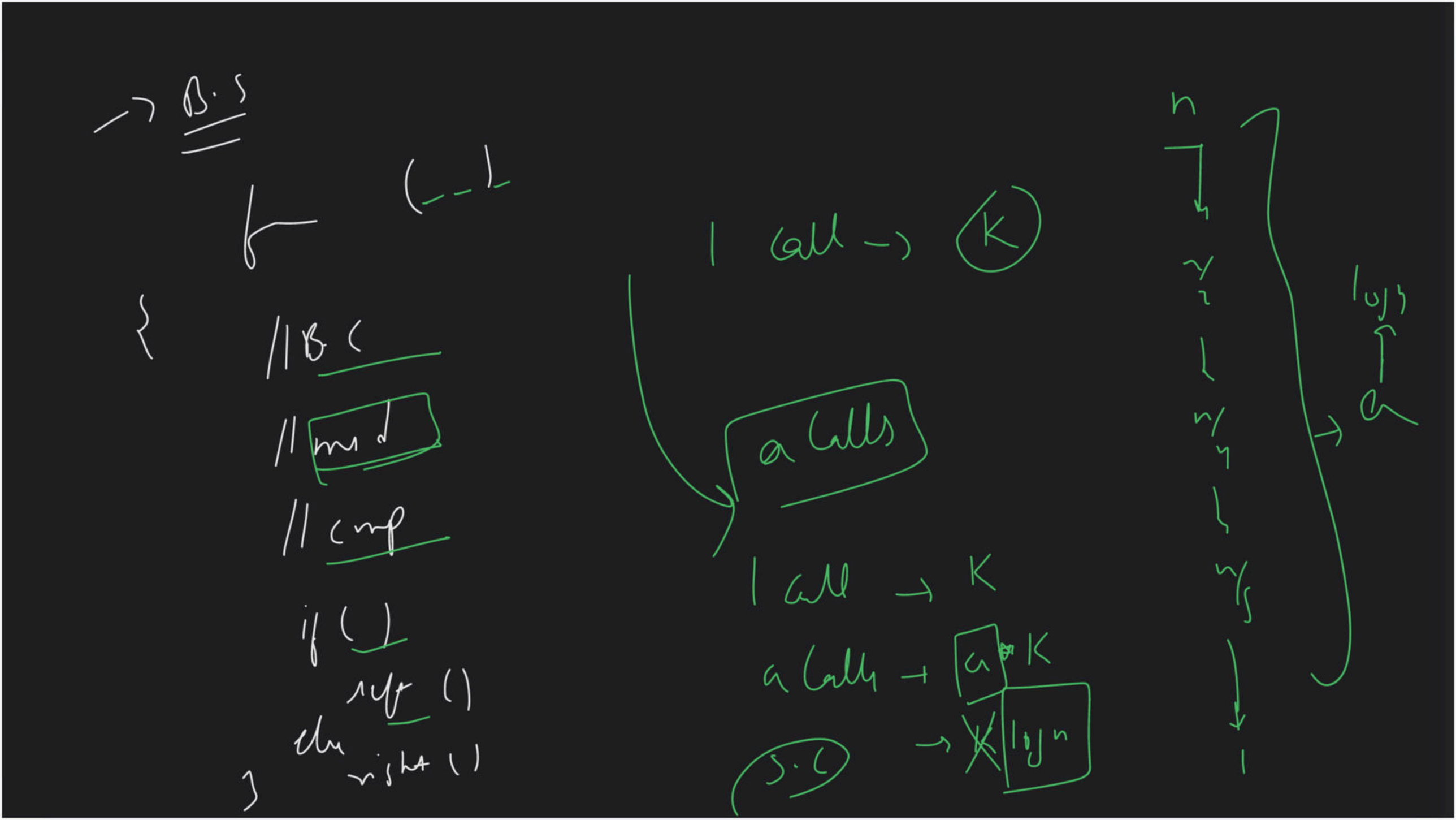
space → max space used
at any instant
of time

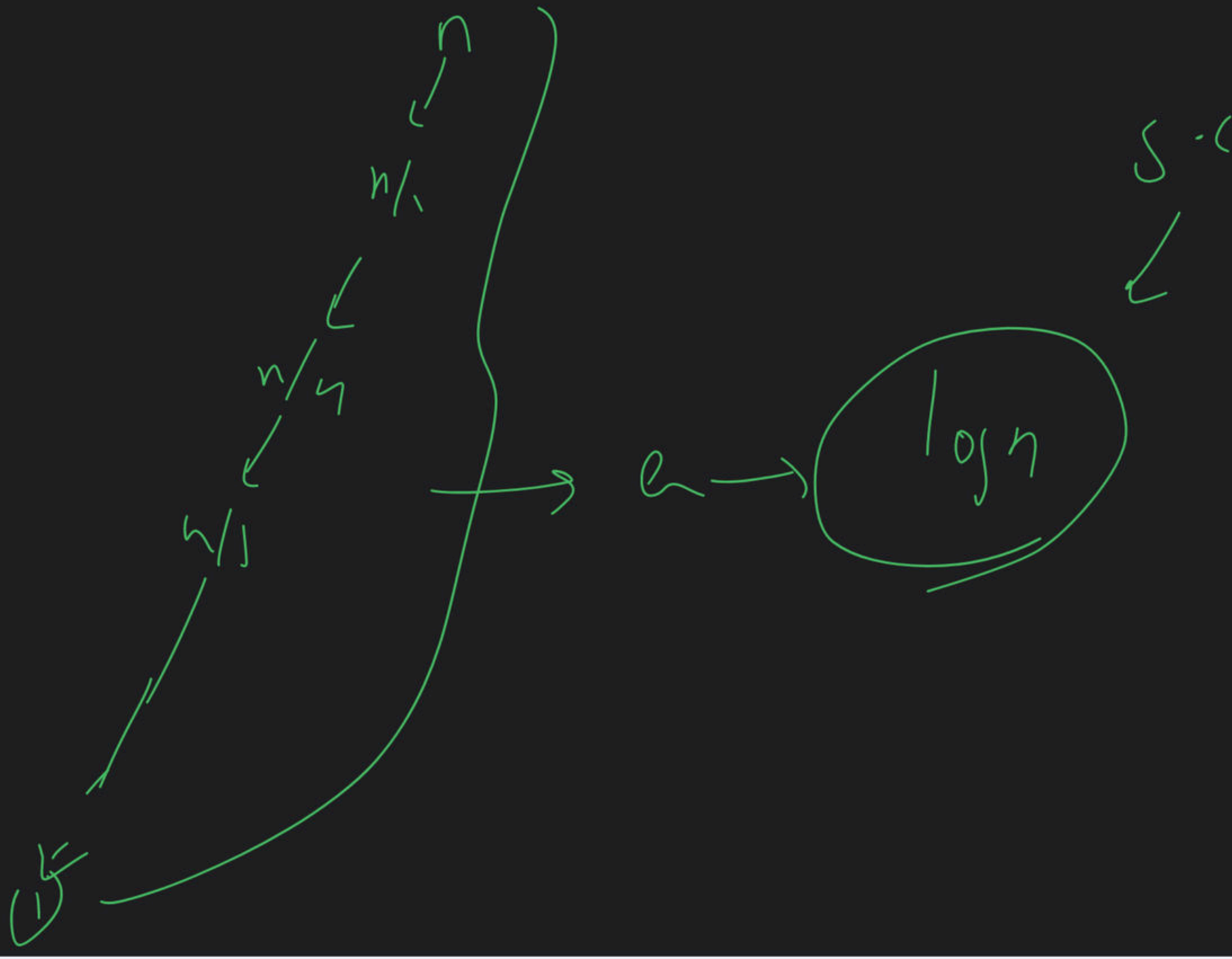
rec
depth

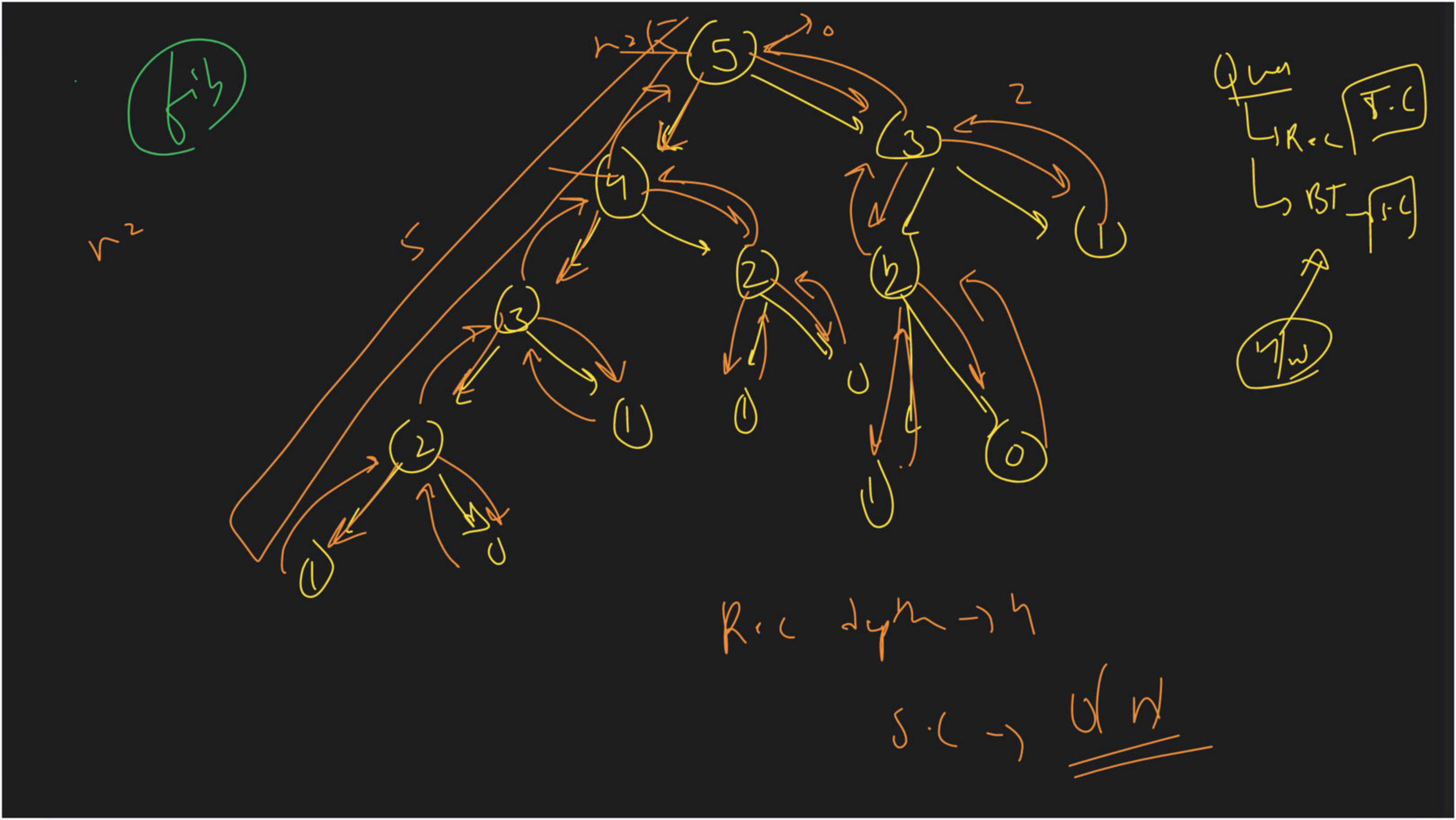


O(n)

factorial


B.S
{
//B C
//mid
//cmp
if ( )
left ()
else right ()
}
1 call → K
a calls
1 call → K
a calls → a * K
S.C → X log n
n
n-1
n-2
n-3
n-4
n-5
1
log n


n
n/2
n/4
n/8
1
log n
S.C


Rec dyn

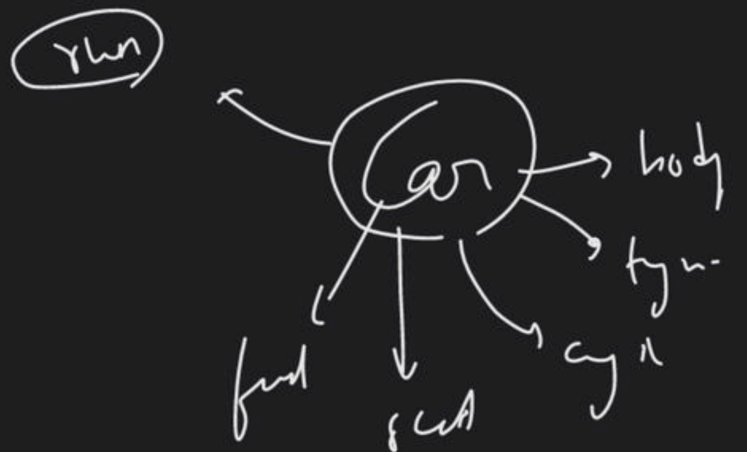
OOPS:-

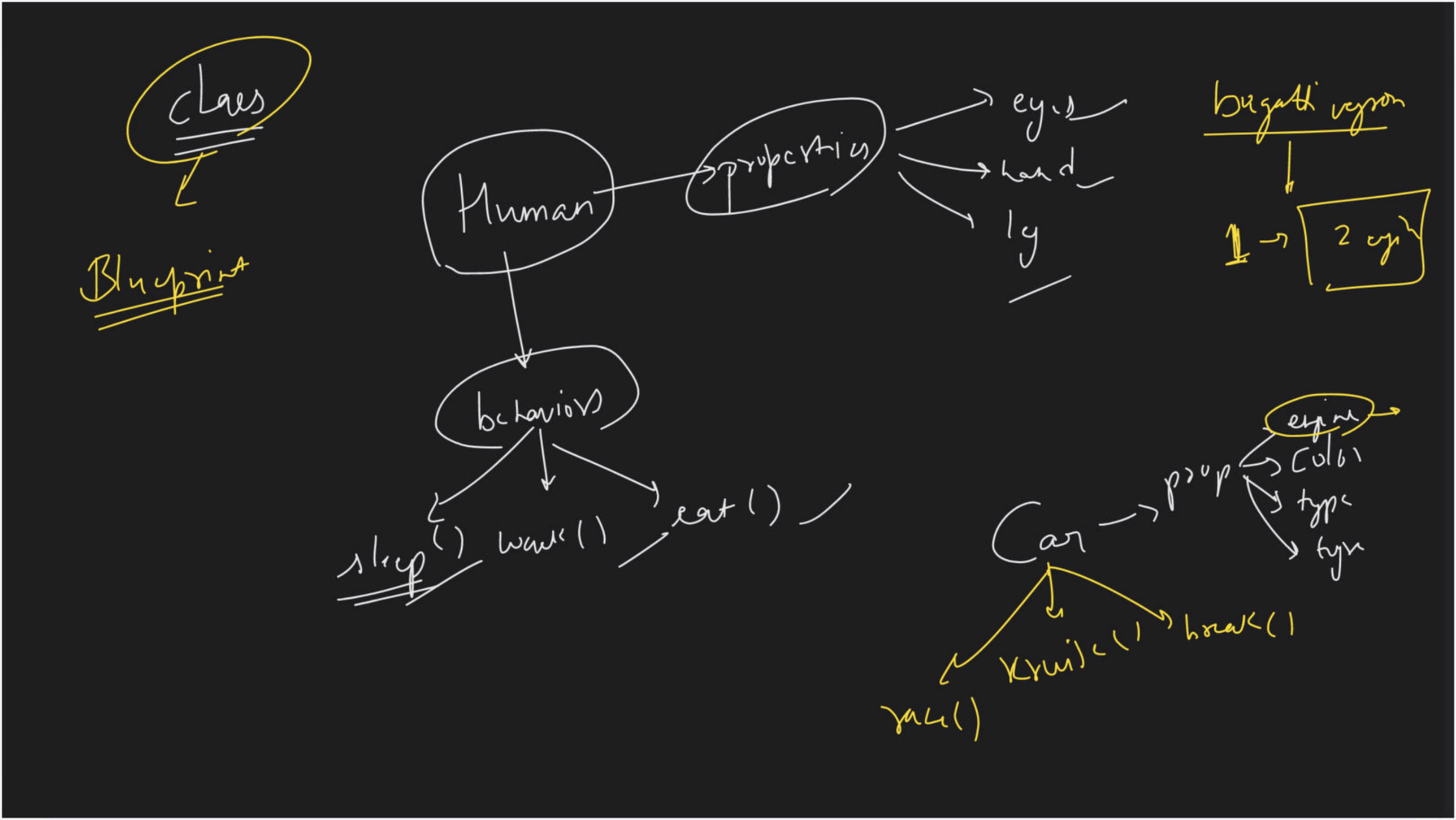
?

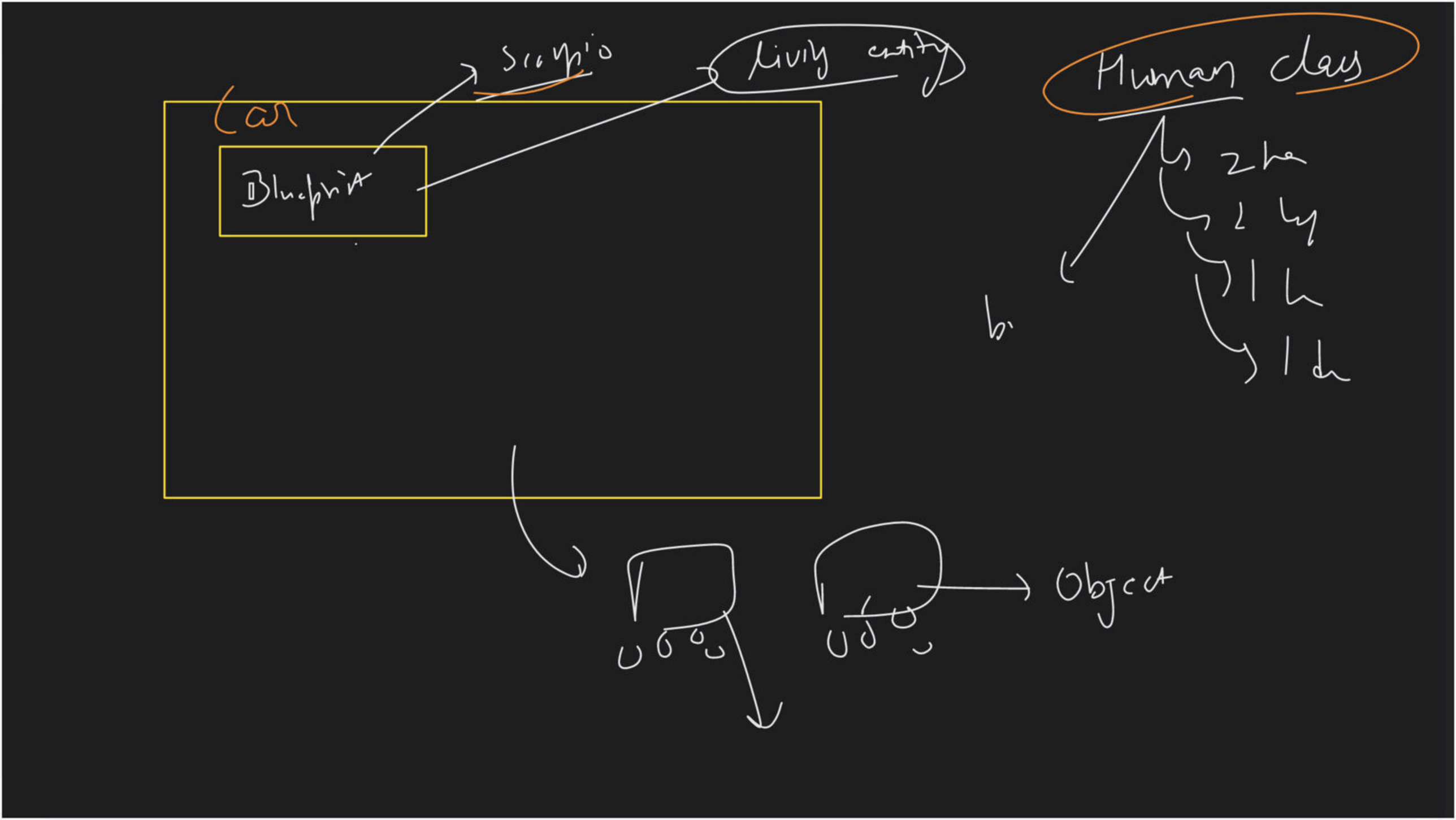
what → ?

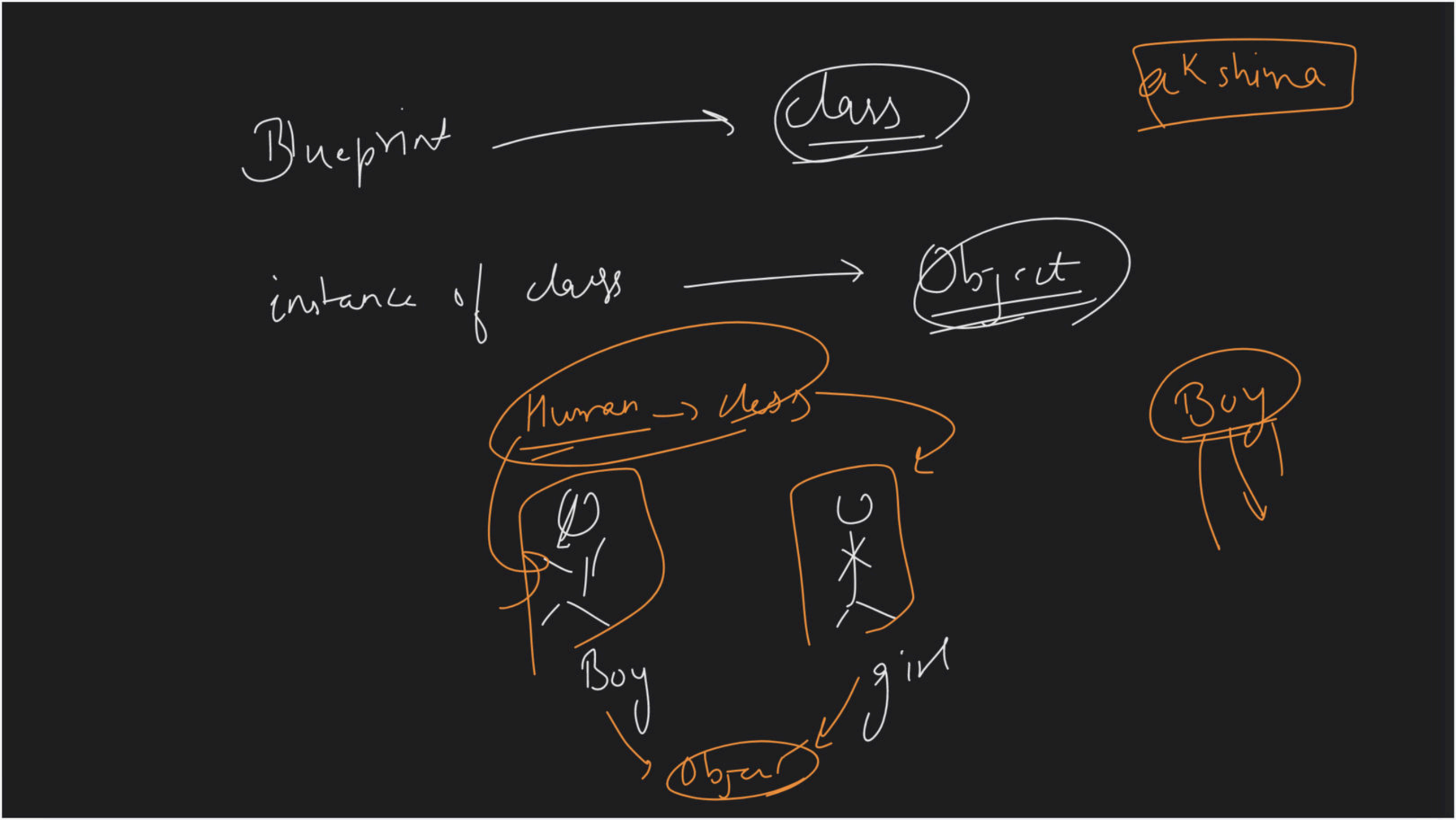
Class & objects

plan → fly




class
Blueprint
Human
properties
eyes
hands
legs
behaviors
sleep()
walk()
eat()
bugatti veyron
1
2 eyes
engine
Car
prop
color
type
fuel
jack()
Kruise()
break()


Scorpio
Living entity
Car
Blueprint
Human class
Object


Blueprint
class
Akshina
instance of class
Object
Human → Class
Boy
Boy
girl
Object

Human class

↳ String gender → male
→ female

class

Object

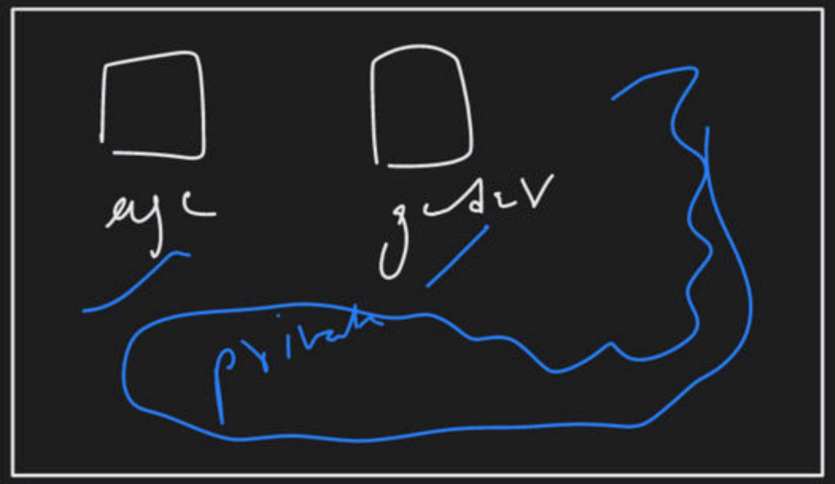
Boy Girl

↳ limit

Boy class

↳ gender → M

Human



main()

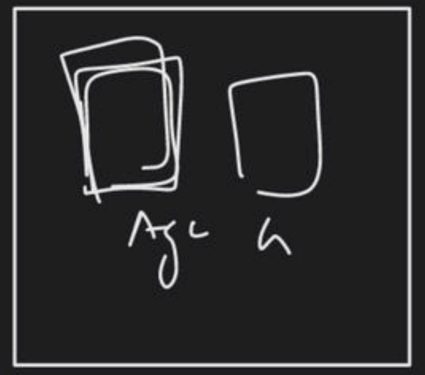
Human obj

obj.age;

obj.gender

error

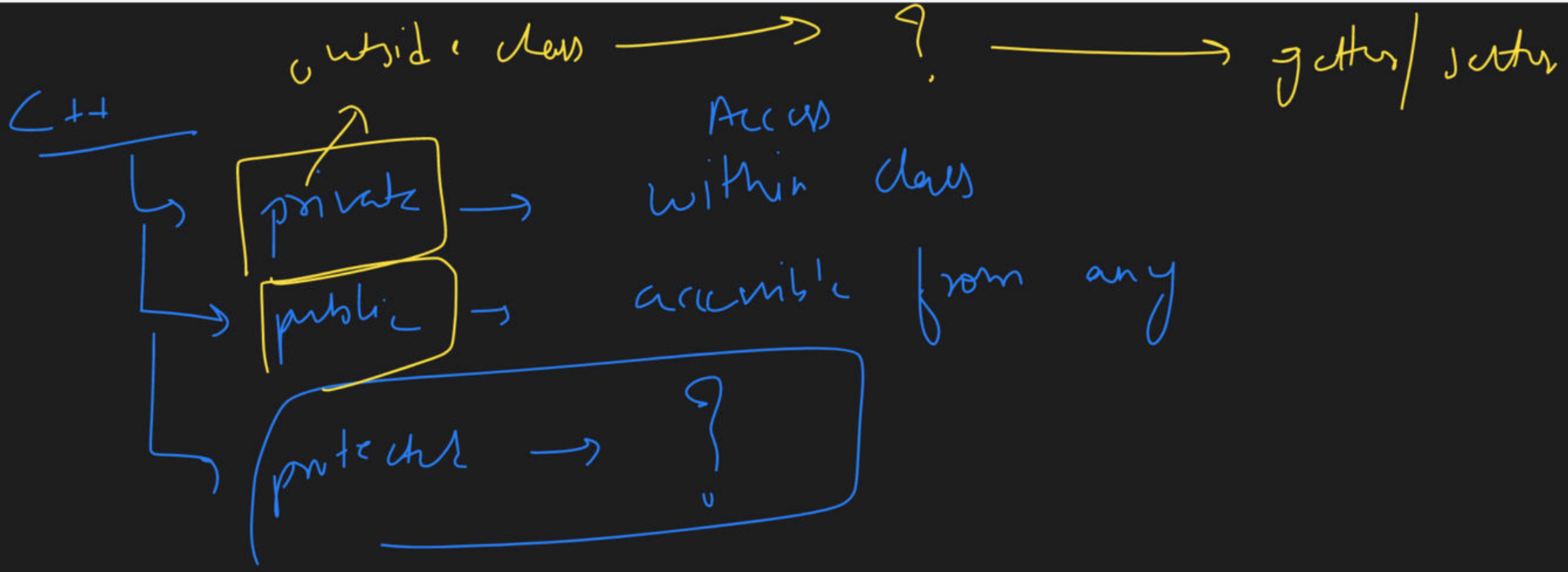
Access modifier / Specifier

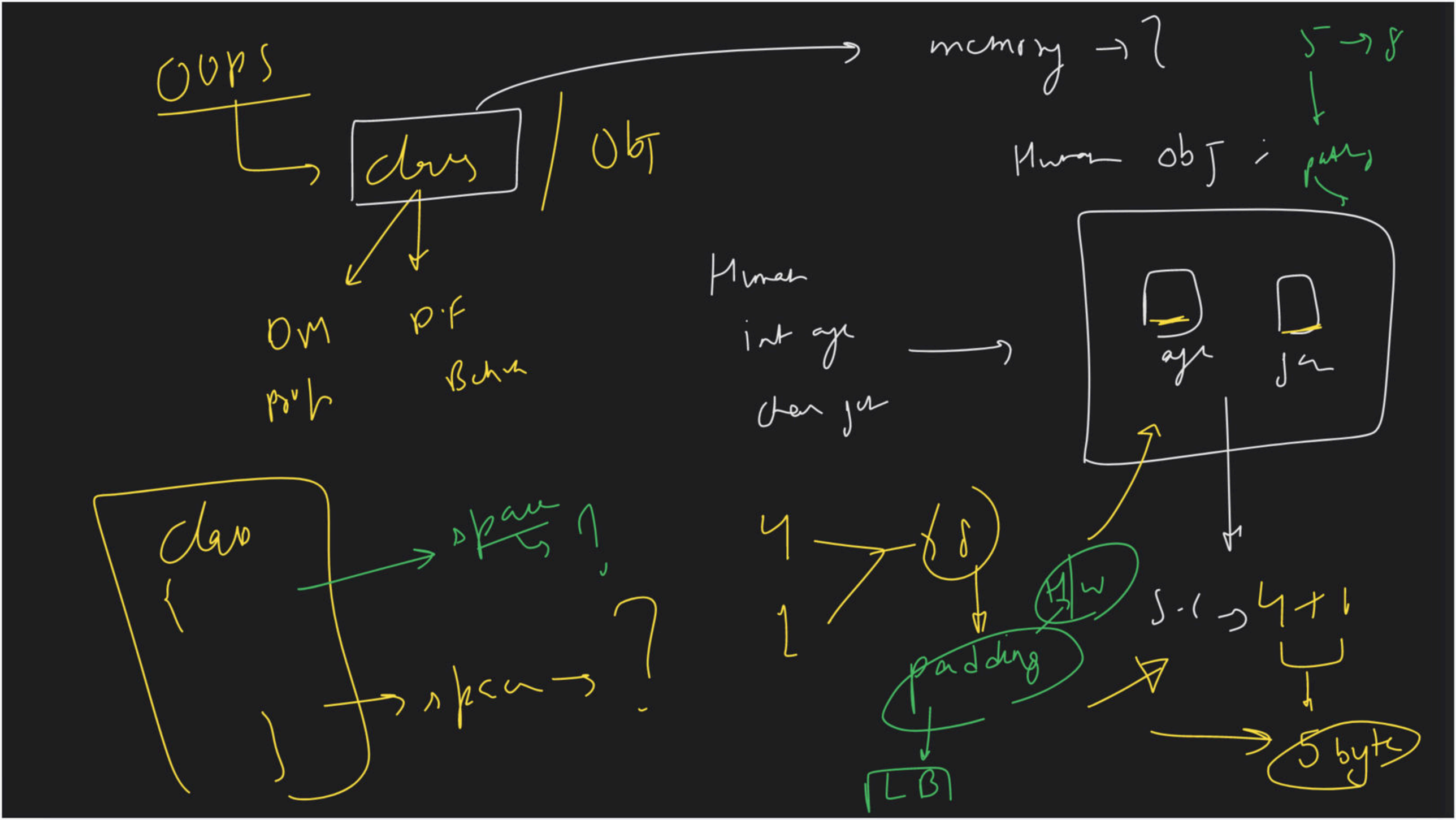


default → private → Access within class

→ public

→ protected → inheritance


outside class → ? → getters/setters
C++
Access
private → within class
public → accessible from any
protected → ?


OOPS
class
Obj
memory
Human obj
OM
P.F
Human
int age
class
age
class
5 byte
padding
LB

class
{
//D.M → Priv
int ___
char ___
str ___
arr [ ]
vector

//method → Pub
int fun()
{
}

void fun()
{
}
};

access

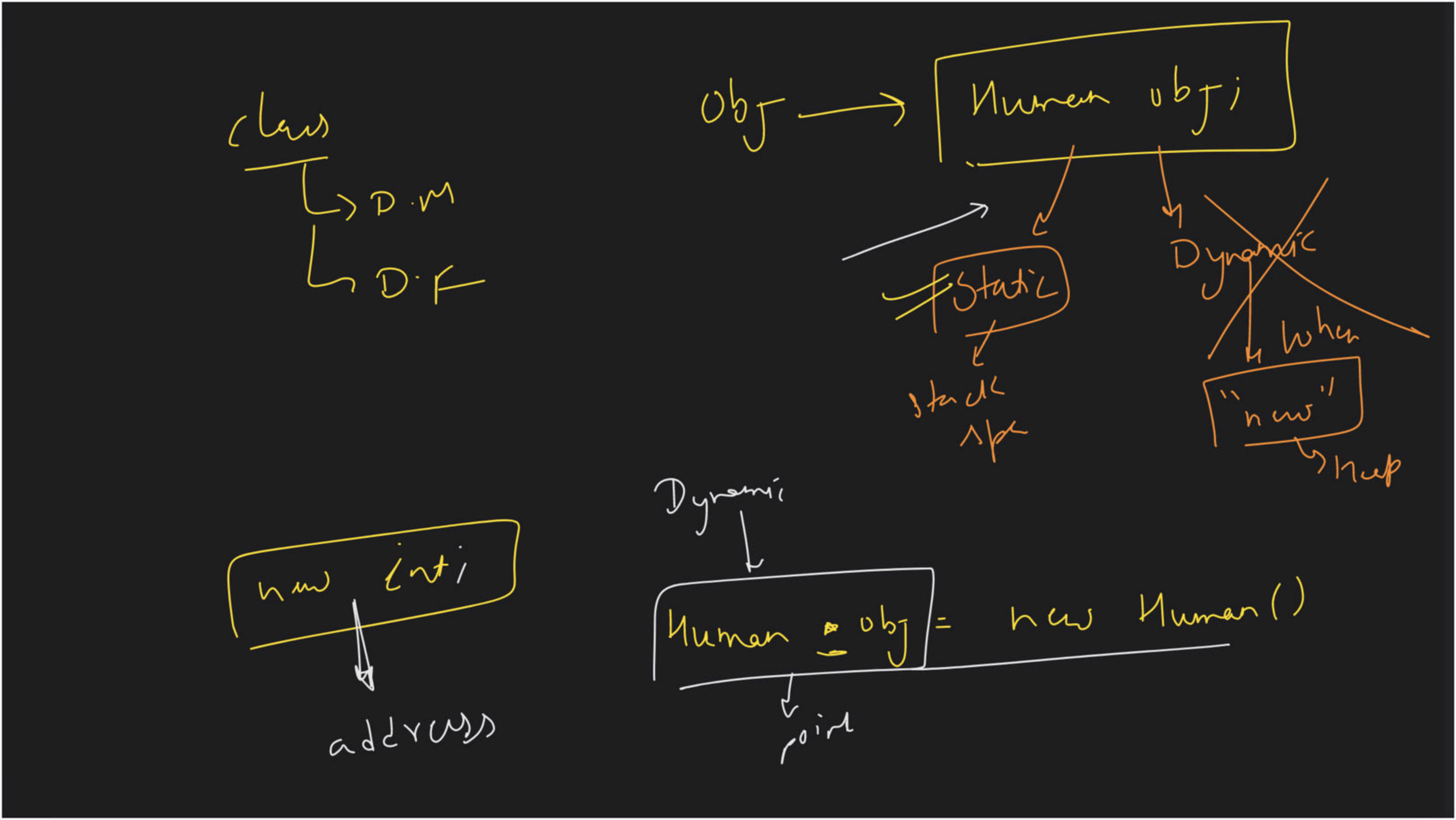
private

→ getter / setter

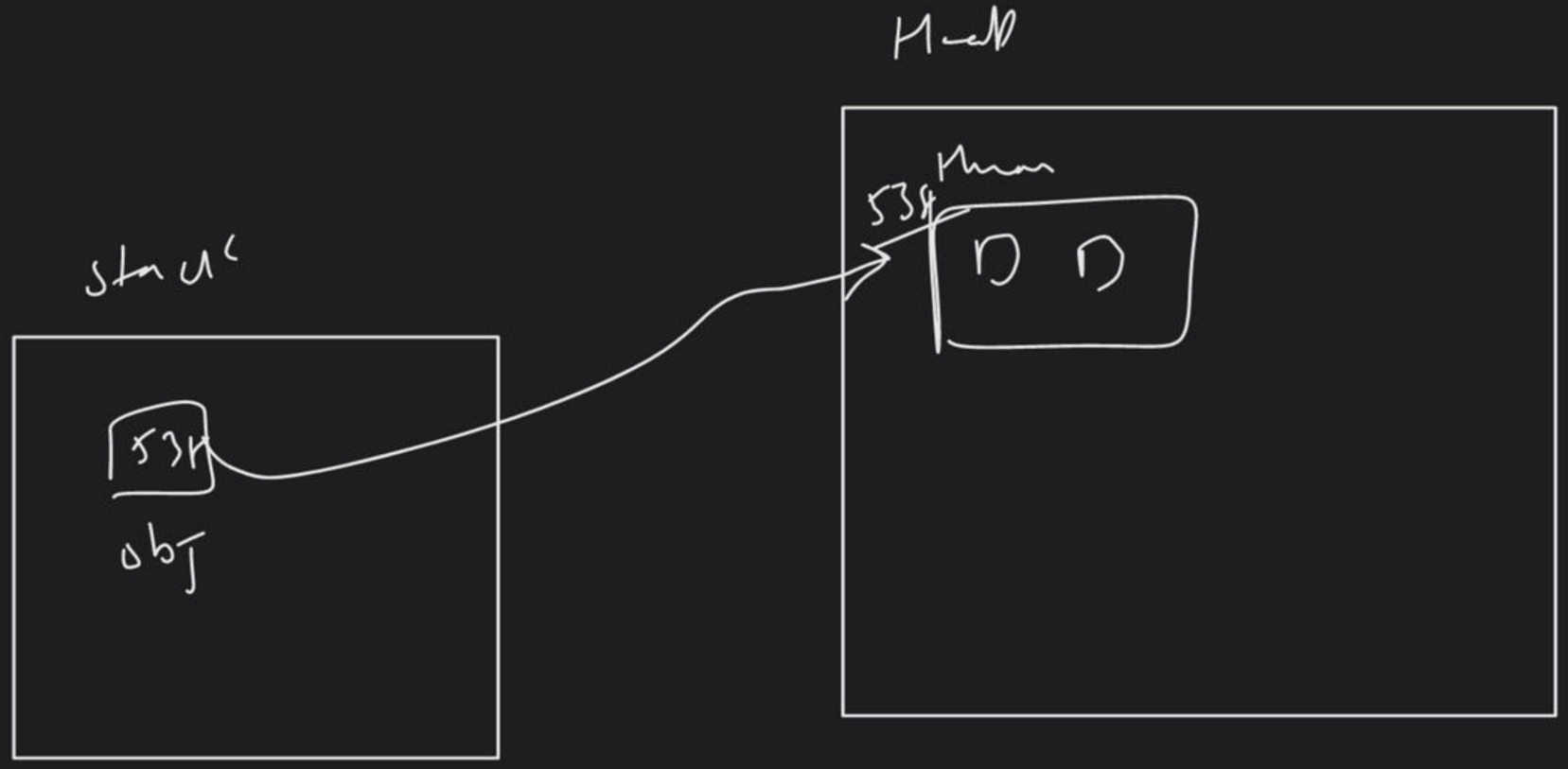
structure → C

int getAge()
{ return age;
}

void setAge(int n)
{ age = n;
}


class
D.M
D.F
Obj
Human obj
Static
Dynamic
when
"new"
heap
new int
address
Dynamic
Human obj = new Human()
point

Human * obj = new Human()



Constructor :-

Obj create

Human ( )

constructor called

types

→ default constructor

(constructor) → name → same class → parameterized constructor

→ no return type

→ obj initialisation

obj

Human (int age, char gender)

{

age = age;

gen = gender

}

this → age = age

(curr ob

this → pointer to your current obj

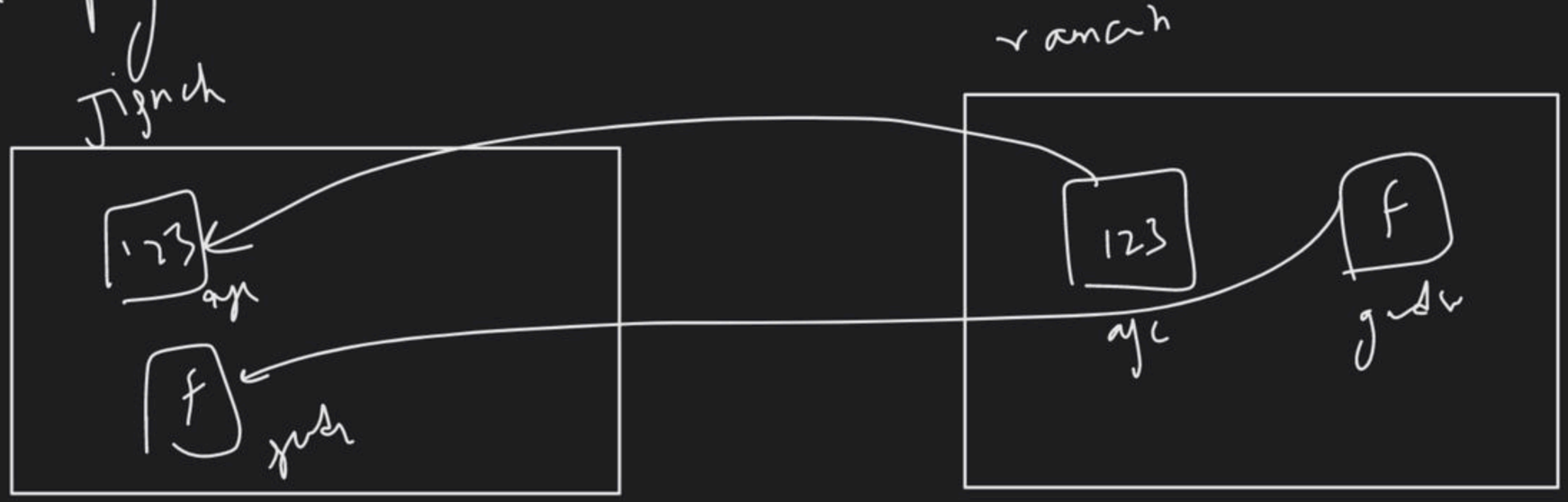
= &

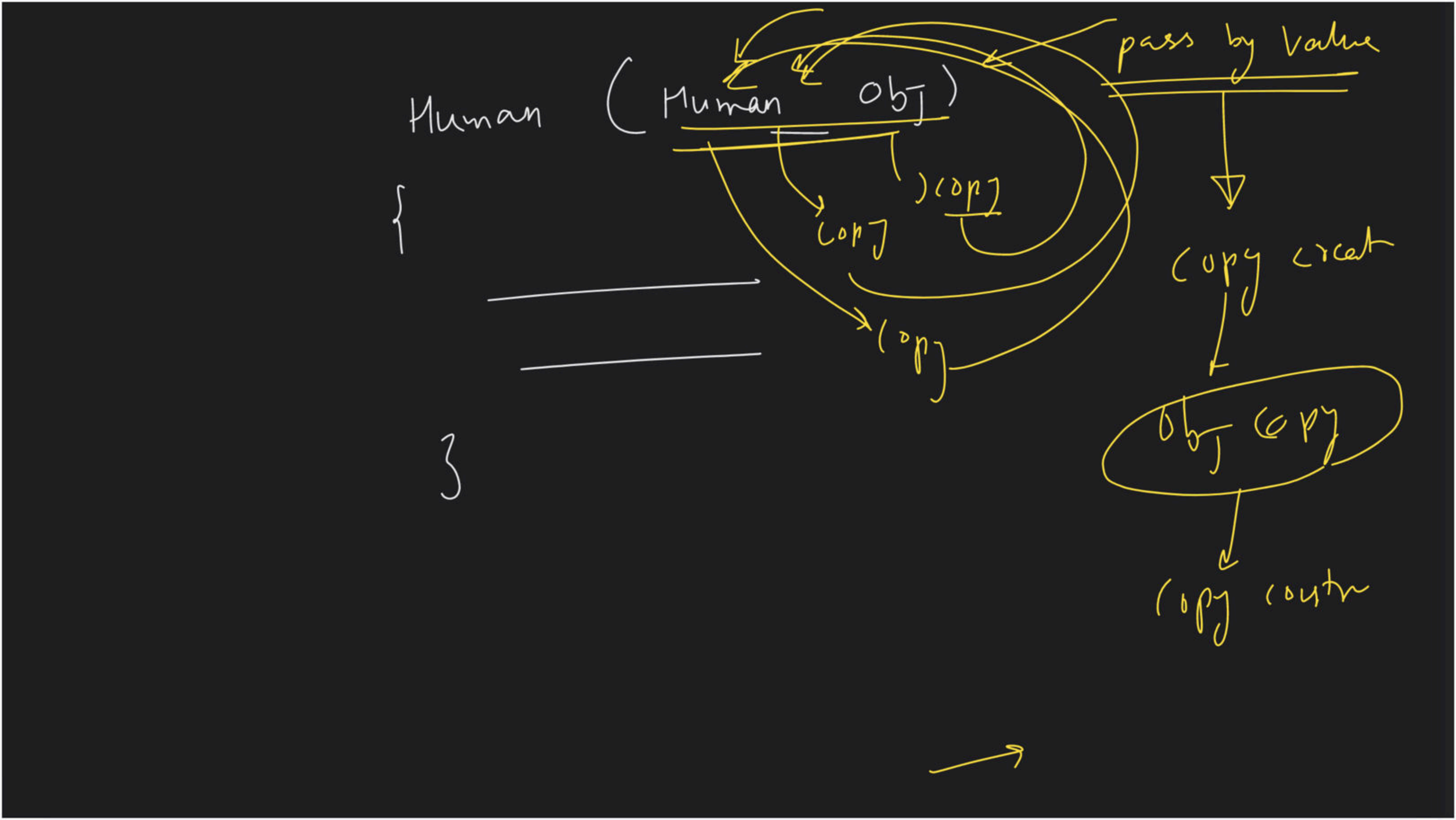
→

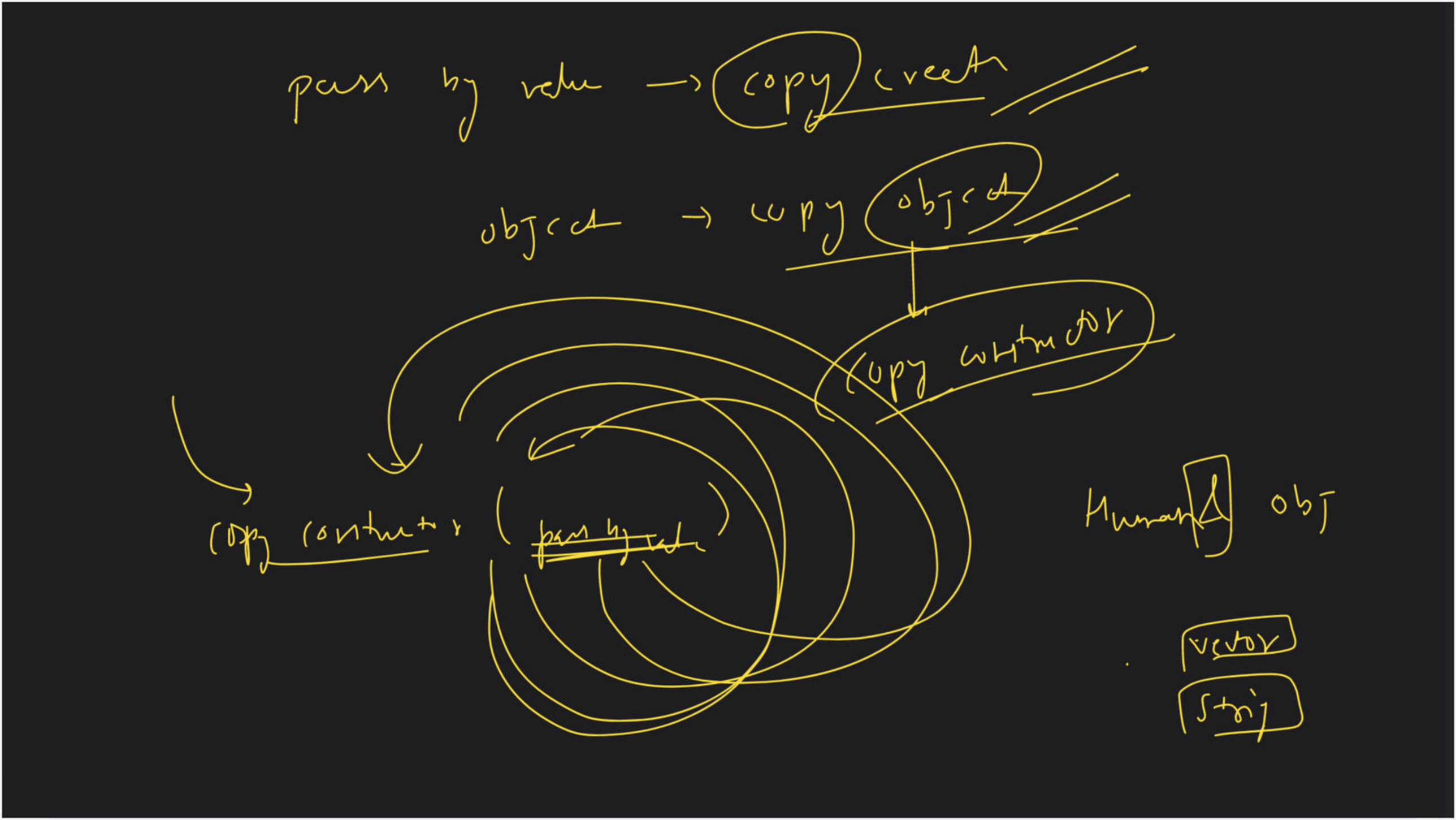
→ default constructor

→ parameterized constructor

→ Copy Constructor



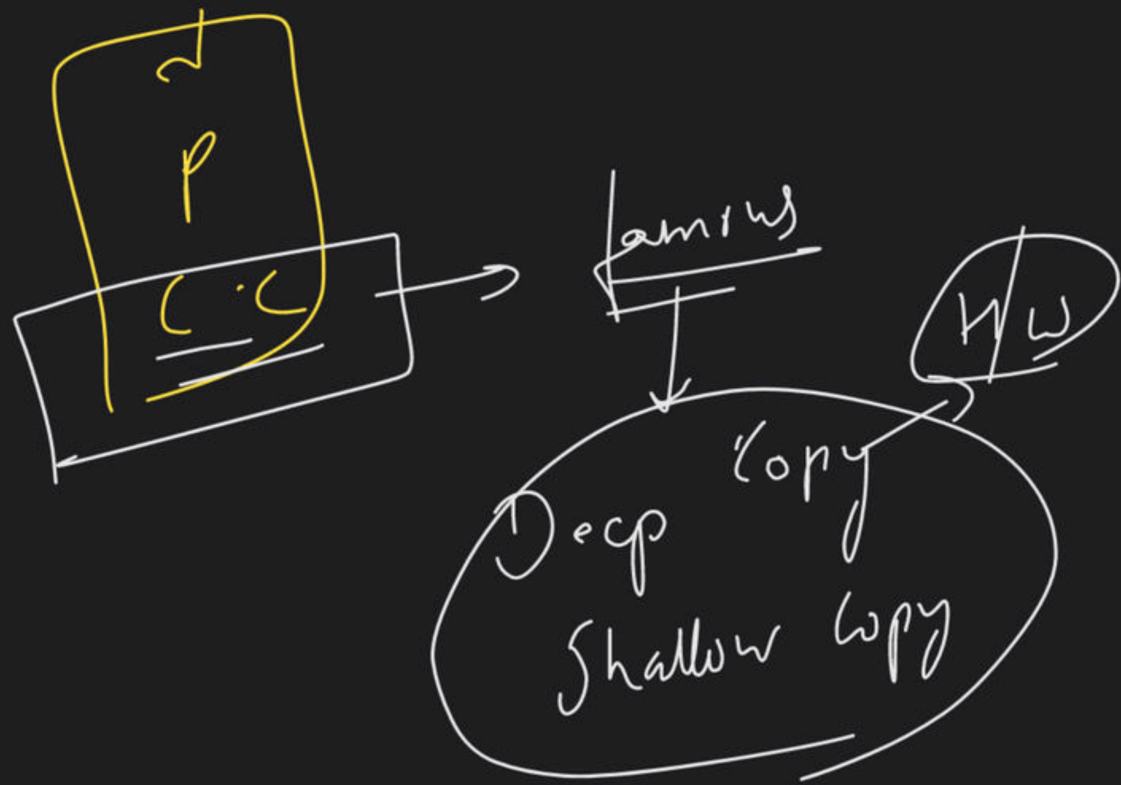

Human
(Human obj)
{
}
copy
copy
copy
pass by value
copy creat
obj copy
copy constr

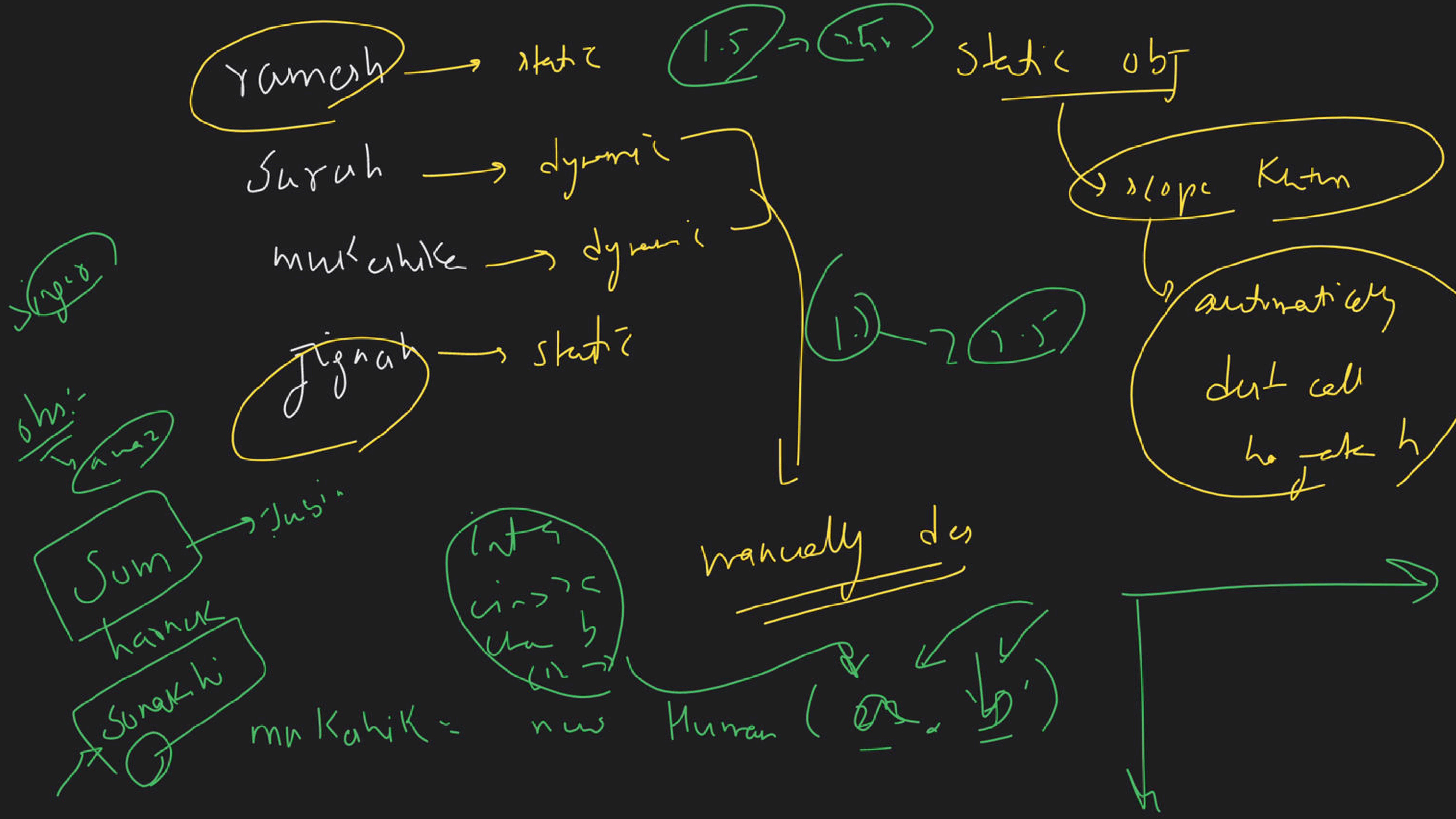

pass by value → copy creation
object → copy object
copy constructor
copy constructor (pass by value)
Human obj
vector
string

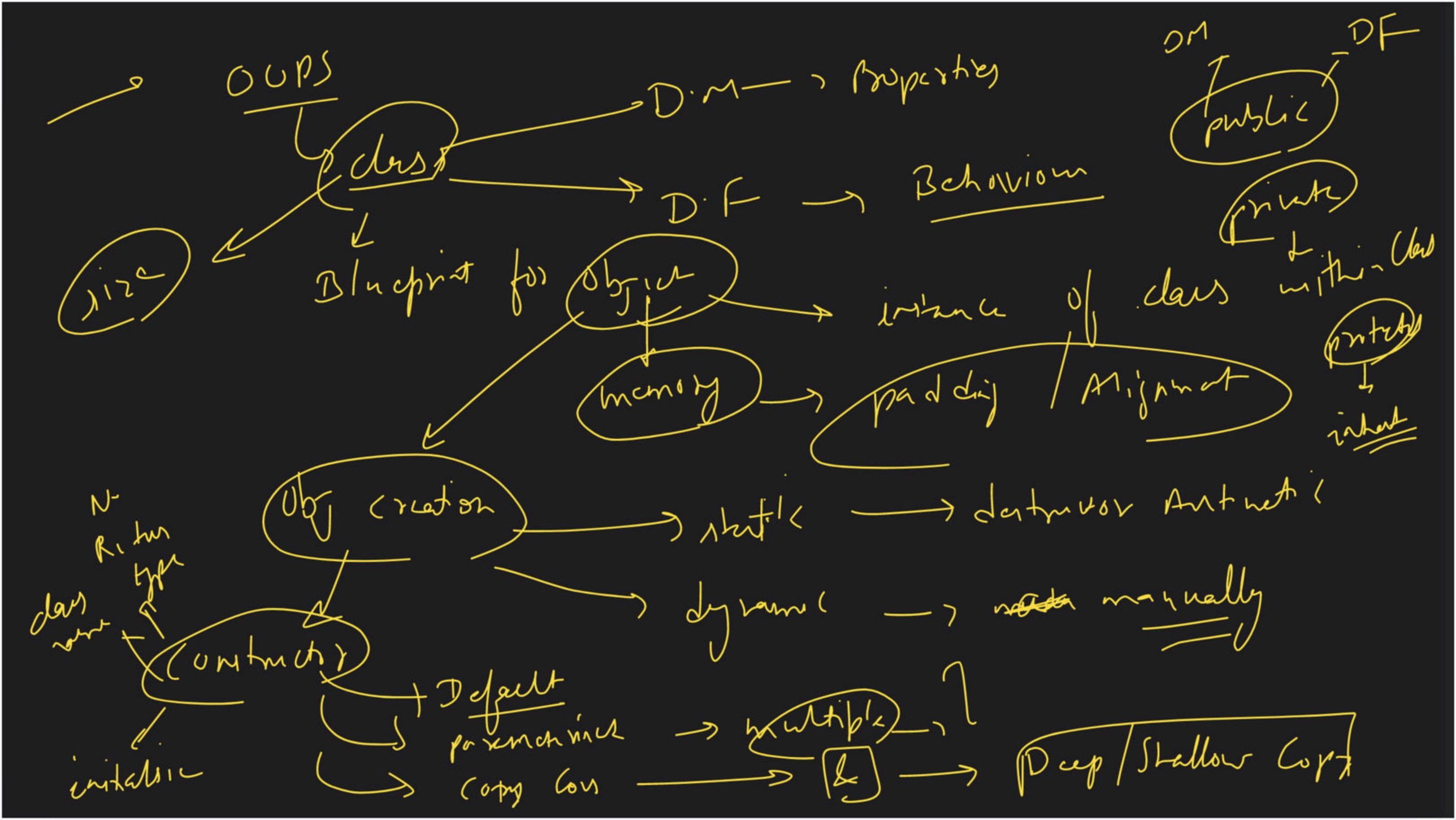
→ Destructor

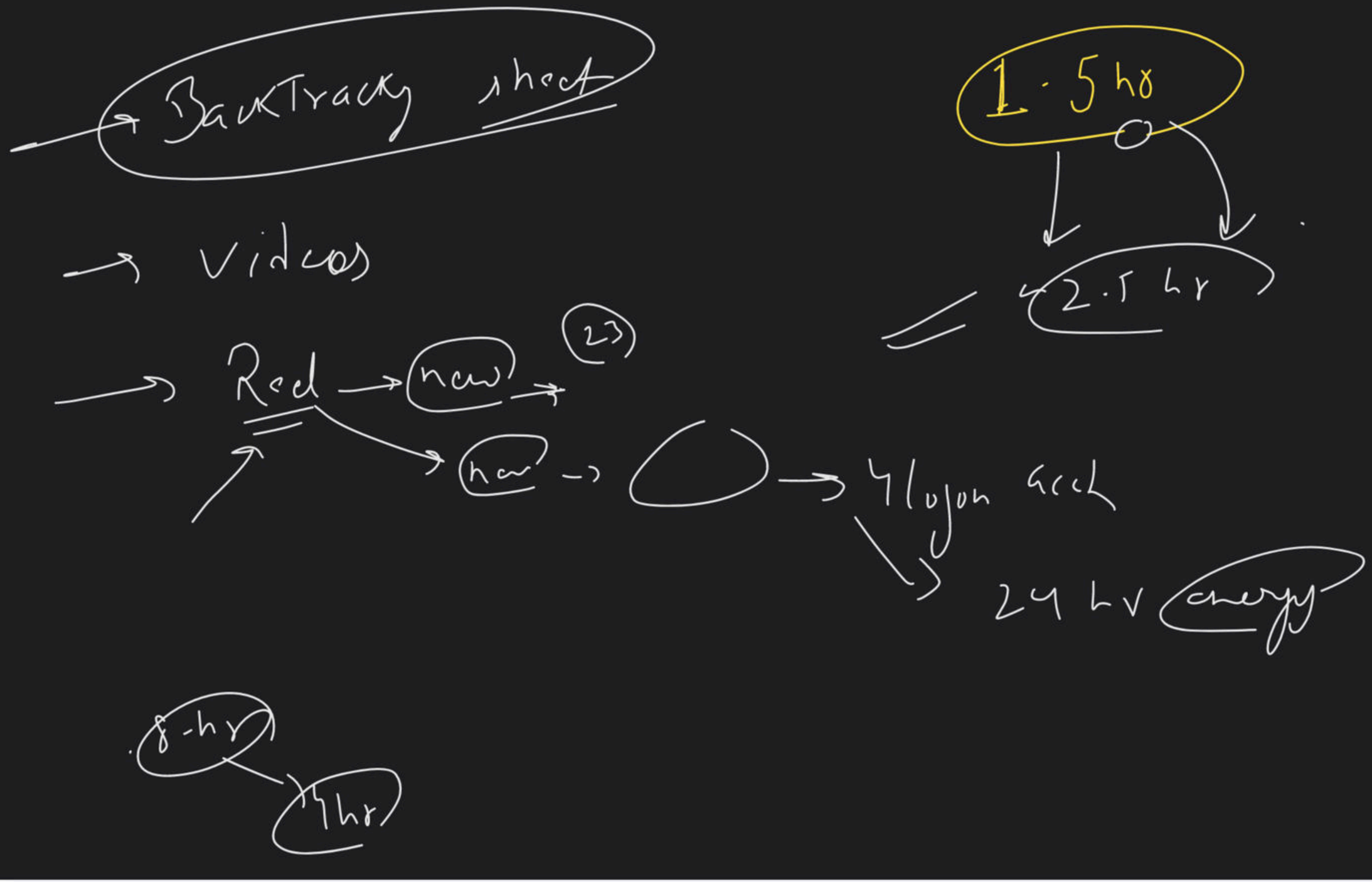
~Human ()
{

}




ramesh
static
Suresh
dynamic
dynamic
static
Static obj
manually del
new Human (


OOPS
class
D.M → Property
D.F → Behaviour
Blueprint for
Object
instance of class
memory
padding / Alignment
DM
DF
public
private
protected
within class
inherit
Obj creation
static
dynamic
destructor Automatic
manually
N
Return type
class name
Constructor
Default
parameterized
copy Cons
multiple
&
Deep / Shallow Copy
initialize


Backtrack sheet
1.5 hr
2.5 hr
Videos
Read
new
23
new
Yojon Geek
24 hr energy
8-hr
7hr