Time & Space Complexity of Recursive Algorithms [Part - 2] - LIVE

Special class

Love Babbar • Sept 18, 2022

① **Merge Sort**

$$T(n) = \underbrace{K_1 + K_2}_{\downarrow} + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + K_3 n + K_4 n$$

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

$$= K + 2 * T\left(\frac{n}{2}\right) + nK_5$$

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

$$1^{st} \to 2^{0} \qquad \boxed{T(n)} = 2\cancel{T\left(\frac{n}{2}\right)} + n * K_s$$

$$II^{nd} \to 2^{1} \qquad 2\cancel{T\left(\frac{n}{2}\right)} = 4\cancel{T\left(\frac{n}{4}\right)} + \frac{n}{\cancel{2}} * K_s$$

$$III^{rd} \to 2^{2} \qquad 4\cancel{T\left(\frac{n}{4}\right)} = 8\cancel{T\left(\frac{n}{8}\right)} + \frac{n}{\cancel{4}} * K_s$$

$$\vdots \qquad \qquad \vdots \qquad \qquad \vdots$$

$$\to 2^{a-1} \qquad 2^{a-1}\cancel{T(1)} = 2^{a} T(0)$$

$$\left.\right\} a \text{ times}$$

$$2^{a-1} \times 2 = 2^{a}$$

$$a = \log n$$

$$T(n) = 2^{a} * T(0) + a * n * K_s$$

$$T(n) = 2^{a}\boxed{T(1)} + \underline{a * n * \underset{\alpha}{k}}$$

$$= 2^{a} + a * n$$

$$= 2^{\log n} + n * a$$

$$= \underbrace{2^{\log n}}_{n} + n * \log n$$

$$\underline{n} + n\log n = n\log n$$

$$T.C \rightarrow O(\underline{n\log n})$$

$$2^{\log n} + n * \log n$$

$$\boxed{n = 2^{10}}$$

$$\boxed{2^{\log 2^{10}}} + 2^{10} \times \log(2^{10})$$

$$= 2^{10 \times \boxed{\log 2}} + 2^{10} \times \log(2^{10})$$

$$= 2^{10} + 2^{10} * \log(2^{10})$$

$$= \underline{\underline{n}} + n \times \log(n)$$

$$\longrightarrow = n(\log n + 1)$$

$$\begin{tikzcd} \boxed{T.C} \arrow[d] \\ \boxed{n \log n} \end{tikzcd}$$

H/W → Master's theoram

T.C

Counting → $O(n)$ → exp sol

factorial → $O(n$

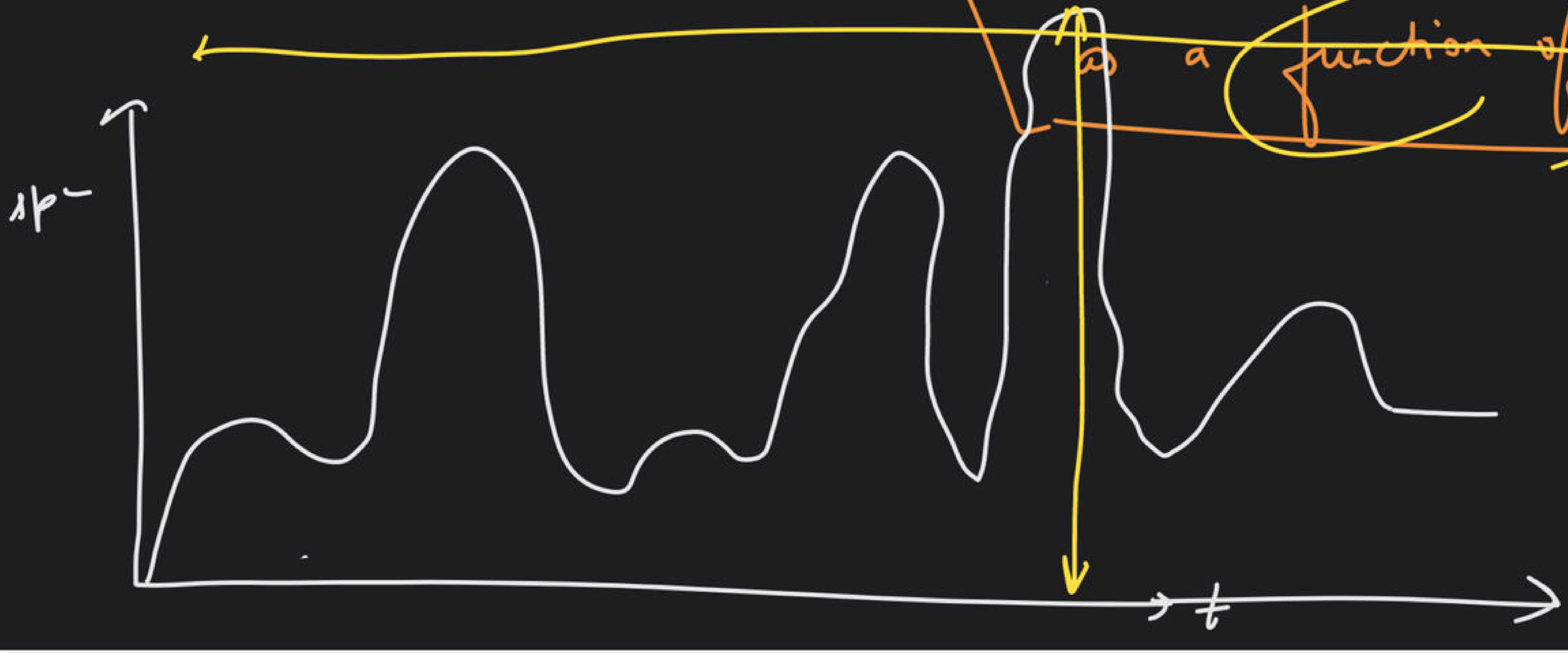
B.S → $O(\log n)$ → $\frac{n}{2^k} = 1$

Merge Sort → $O(n \log n)$ →

fibonacci → $O(2^n)$ → Rec tree

Space complexity → what ?

→ Space required to run an algo as a function of input ✓

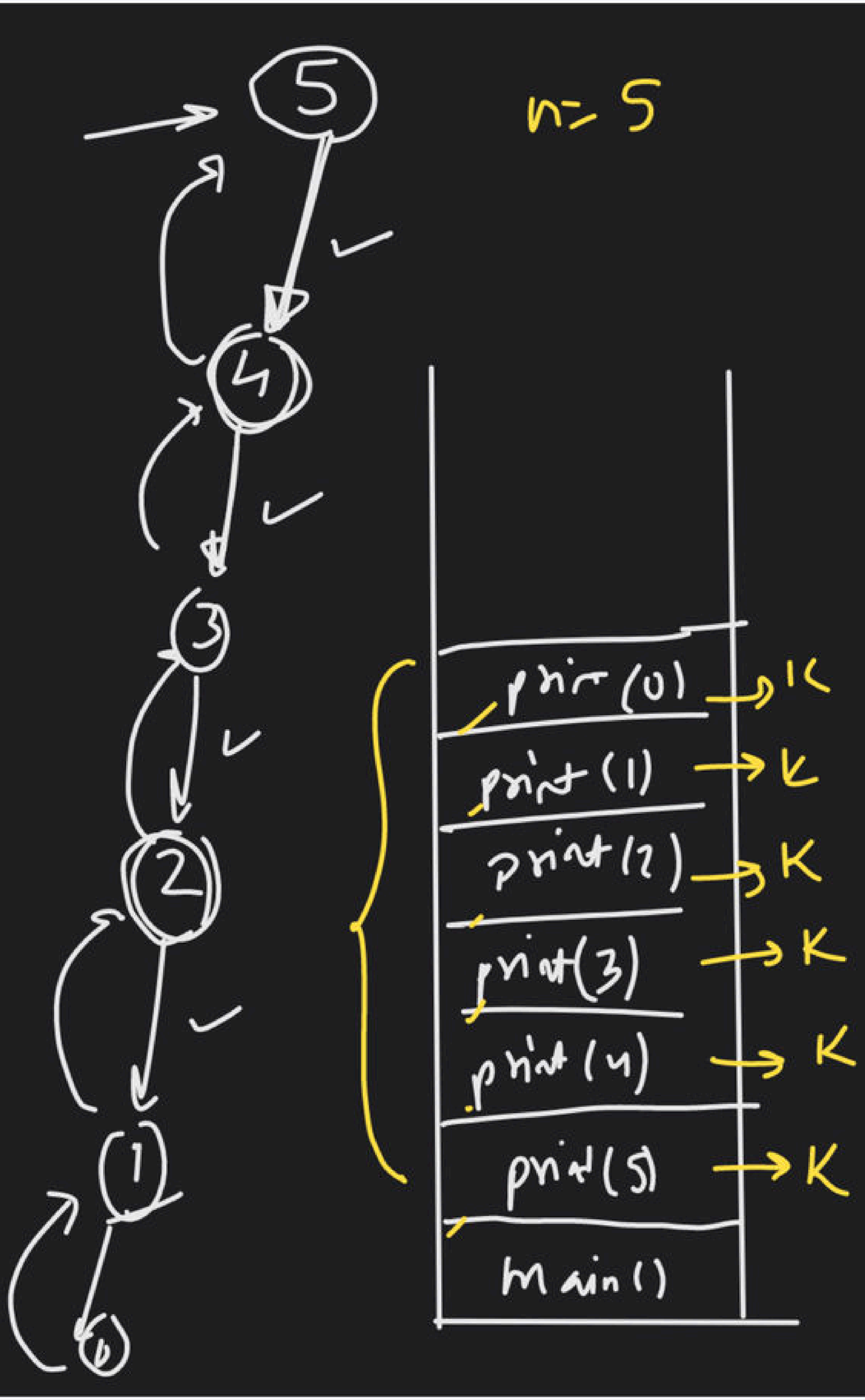


① Counting

```
void print (n)
{
    //B.C
    
    cout <<
    
    //R.R
    print(n-1);
}
```

O(1)

space



print(0) → K
print(1) → K
print(2) → K
print(3) → K
print(4) → K
print(5) → K
main()

1 call $\rightarrow$ K

n+1 call $\rightarrow$ $(n+1) * K$

$$S.C \rightarrow nK + \underset{\alpha}{K}$$

$$\rightarrow n\underset{\alpha}{K}$$

$$\rightarrow n$$

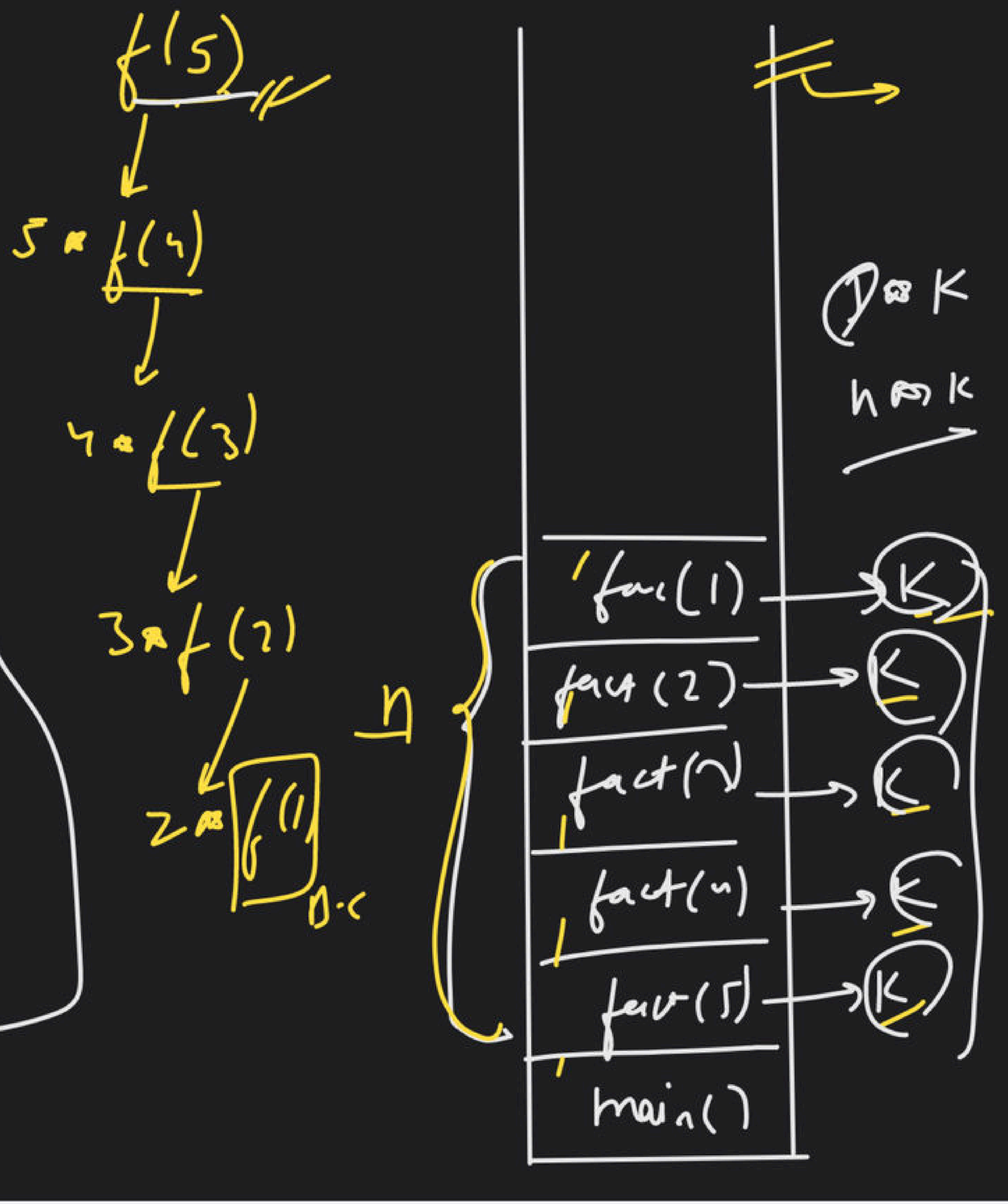
$$S.C \rightarrow \underline{\underline{O(n)}}$$

②

factorial

```
int fact(int n)
{
    //B.C
    if(n==0 || n==1)
        return [illegible];

    return n*fact(n-1);
}
```

constant O(1)



$S.C \rightarrow n * k$

α

$=$

$= O(n)$

③ Binary Search

binSearch(n size) → K

↓

binSearch(n/2 size) → K

↓

binSearch(n/4 size) → K

⋮

binSearch(1 size) → K

a times

$\boxed{a = \log n}$

```
bool   binSearch(arr, s, e, target)
{
    //B.C
    if (s > e
        return false;

    int mid = (s+e)/2;

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

    if (arr[mid] > target)
        return  binSearch(arr, s, mid-1, target);
    else
        return  binSearch(arr, mid+1, e, target);
}
```

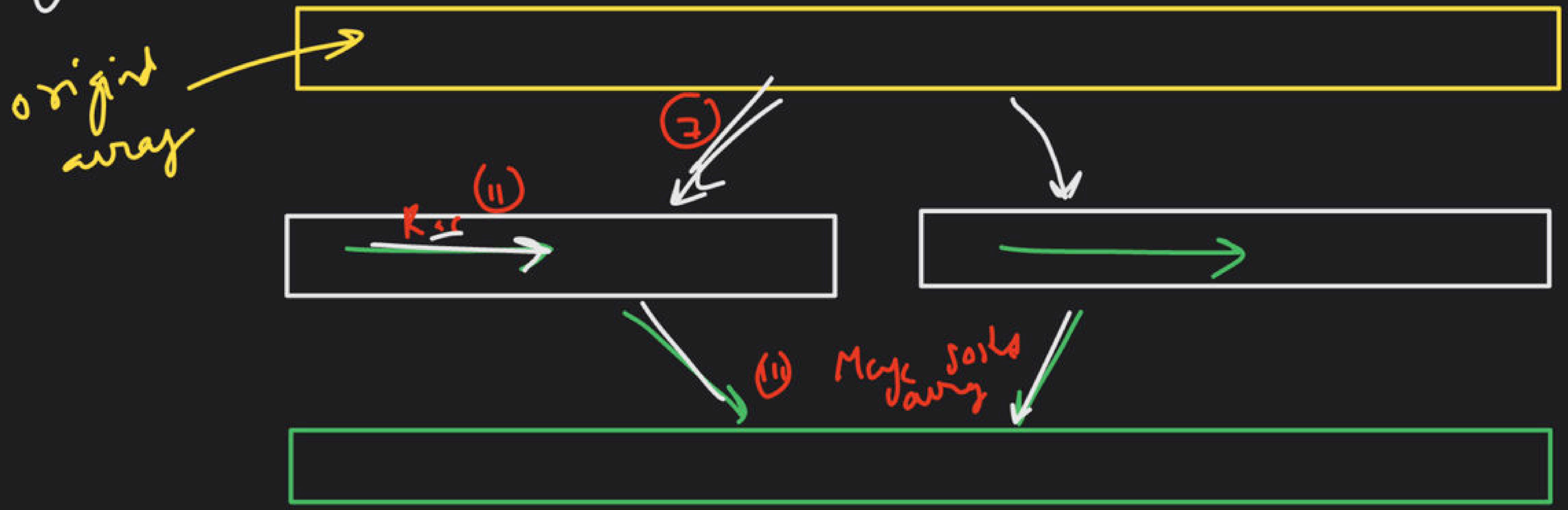
~~Bott~~

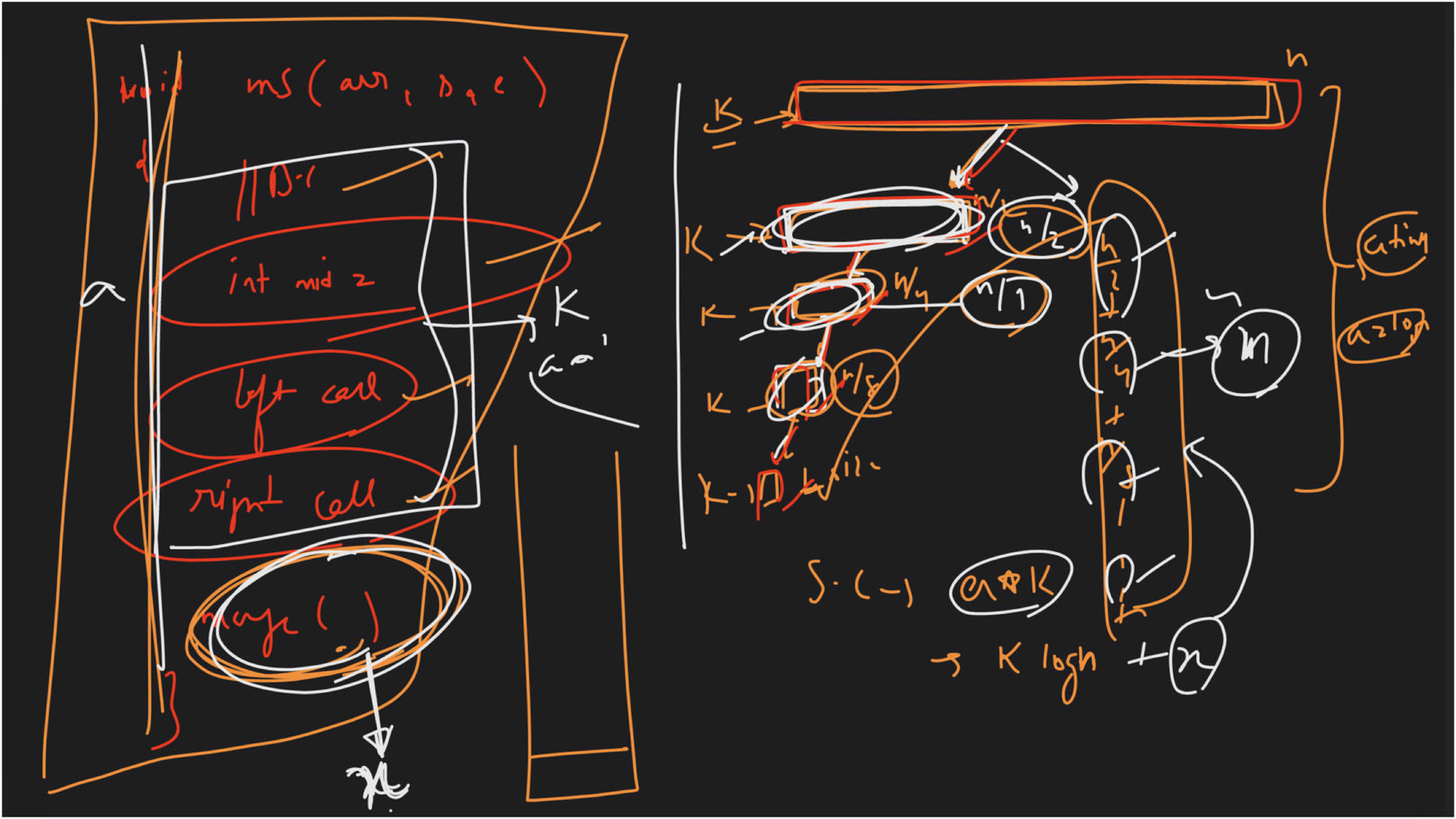
S.C → a * K

$= K * a = \cancel{K} * \log n$

S.C → O(log n)

(4) Merge Sort :-




void ms(arr, s, e)
{
//B.C
int mid =
left call
right call
merge( )
}
a
K
a a'
n
n
K
n/2
n/4
n/8
K-1
S.C (-) a*K
→ K log n + n
n
calling
a=log n

$S.C \rightarrow k \log n + \boxed{n} \rightarrow ?$

$n = \frac{n}{2} + \frac{n}{4} + \frac{n}{8} - - - - + 1$

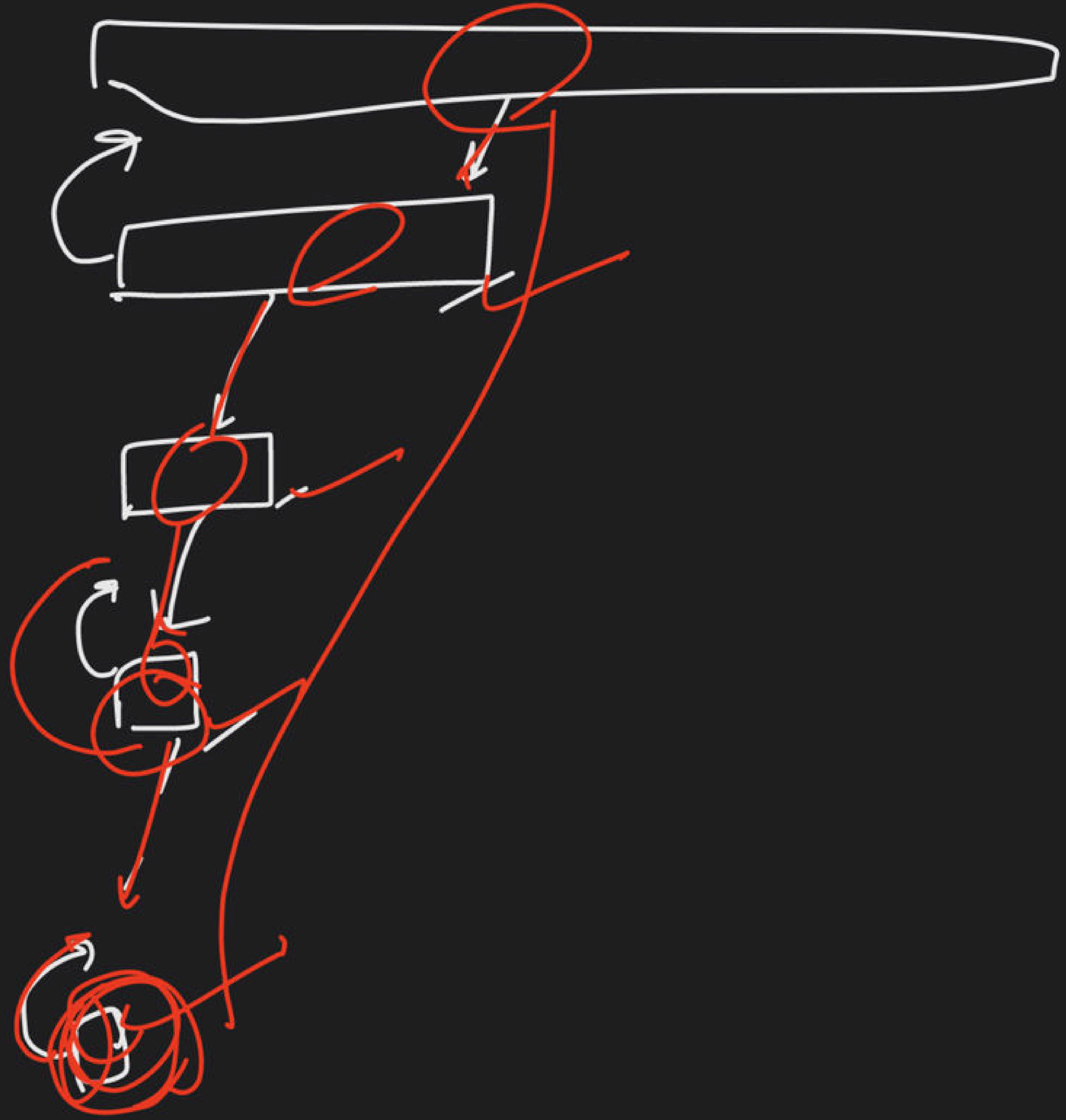
$= n - 1$ or $2n - 1$

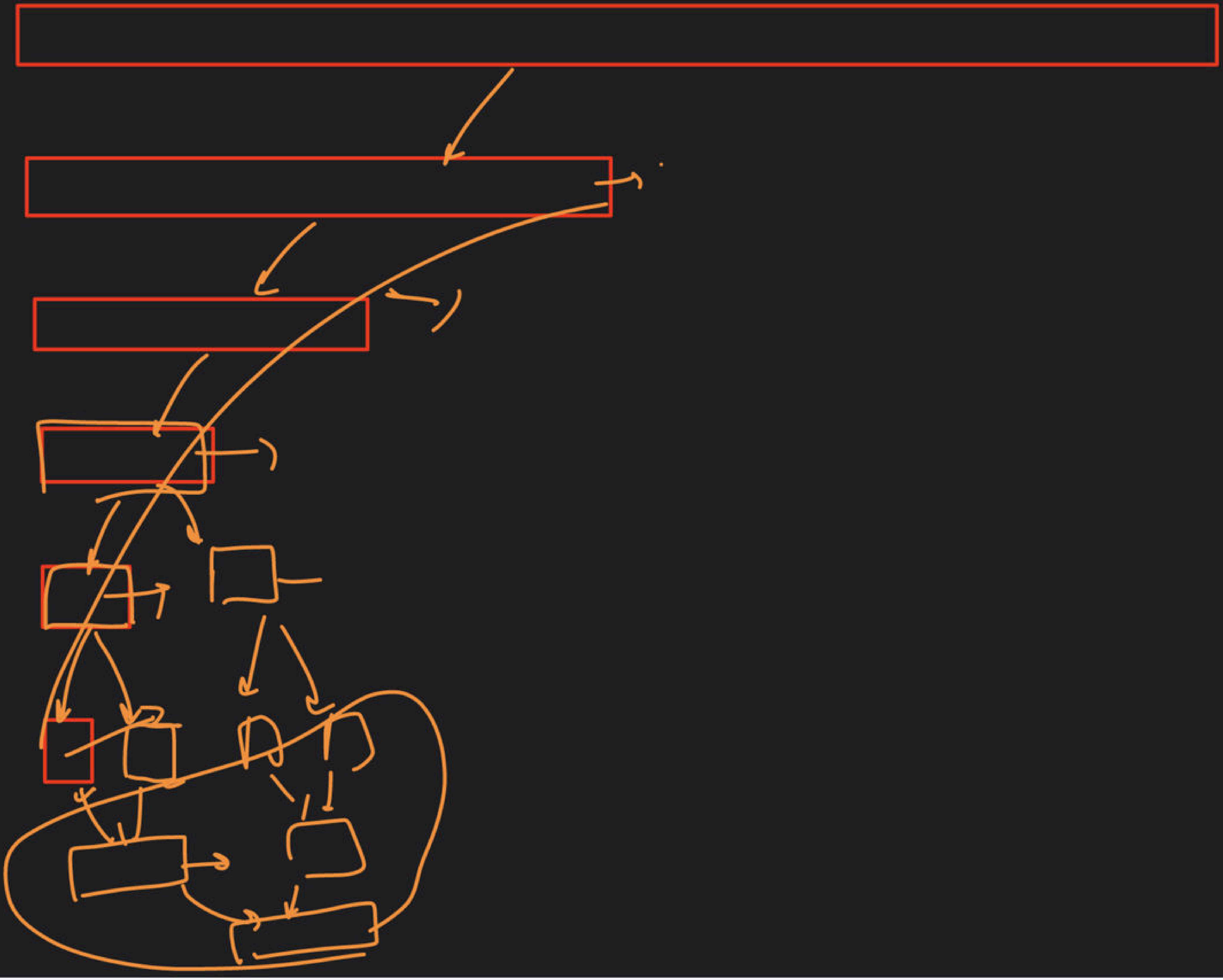
$k \log n <<< n$

$S.C \rightarrow k \log n + n$

$S = O(n)$

$$\begin{tikzcd} \boxed{n =} \; 2^{10} \arrow[r] & \boxed{1024} \arrow[d] \\ \boxed{\log n} = \log(2^{10}) = & \boxed{10} \end{tikzcd}$$





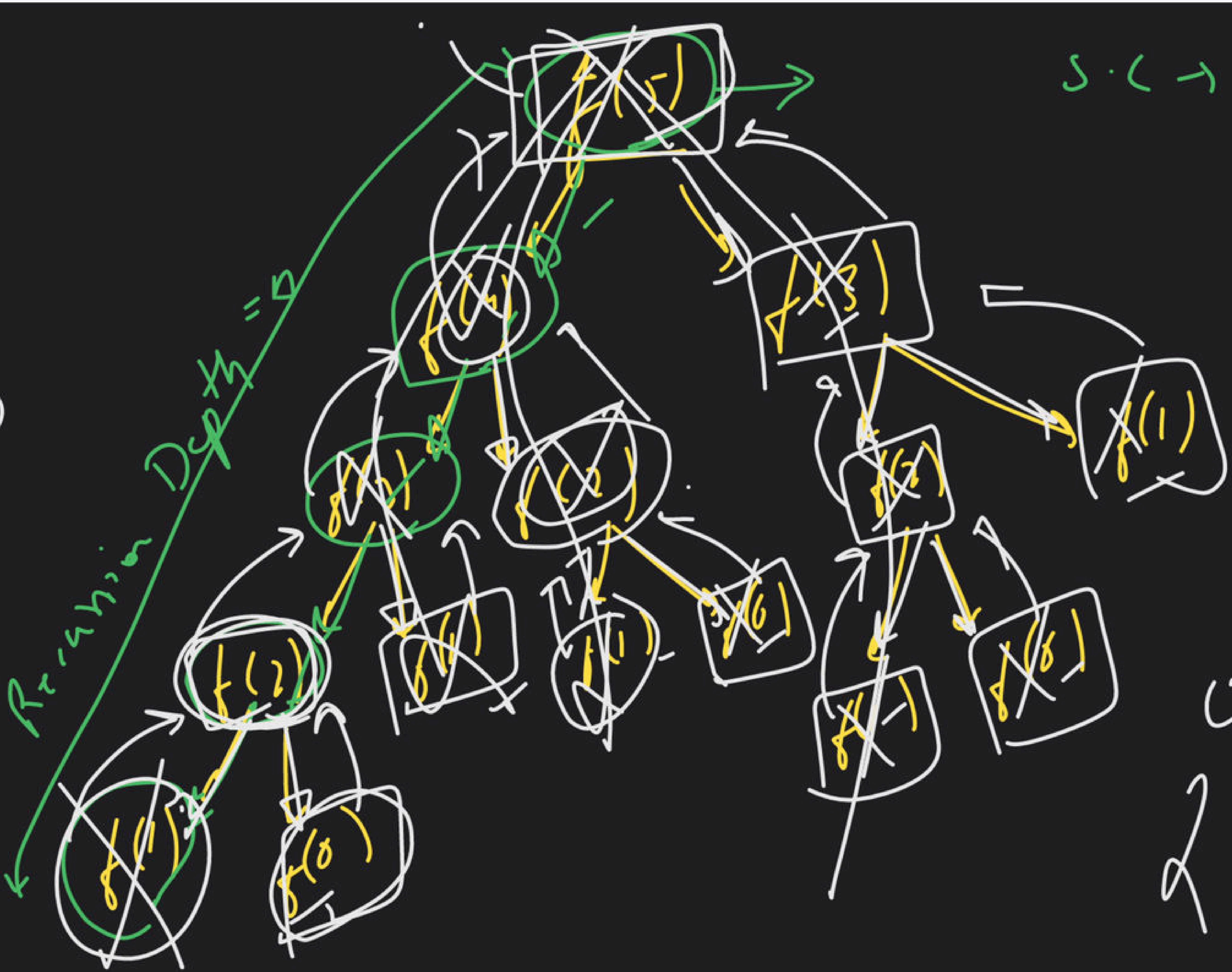
5 Fib Series

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

      return  fib(n-1)+ f(n-2);

  }
```

k

H/W

Codes → S.C → Hid Out



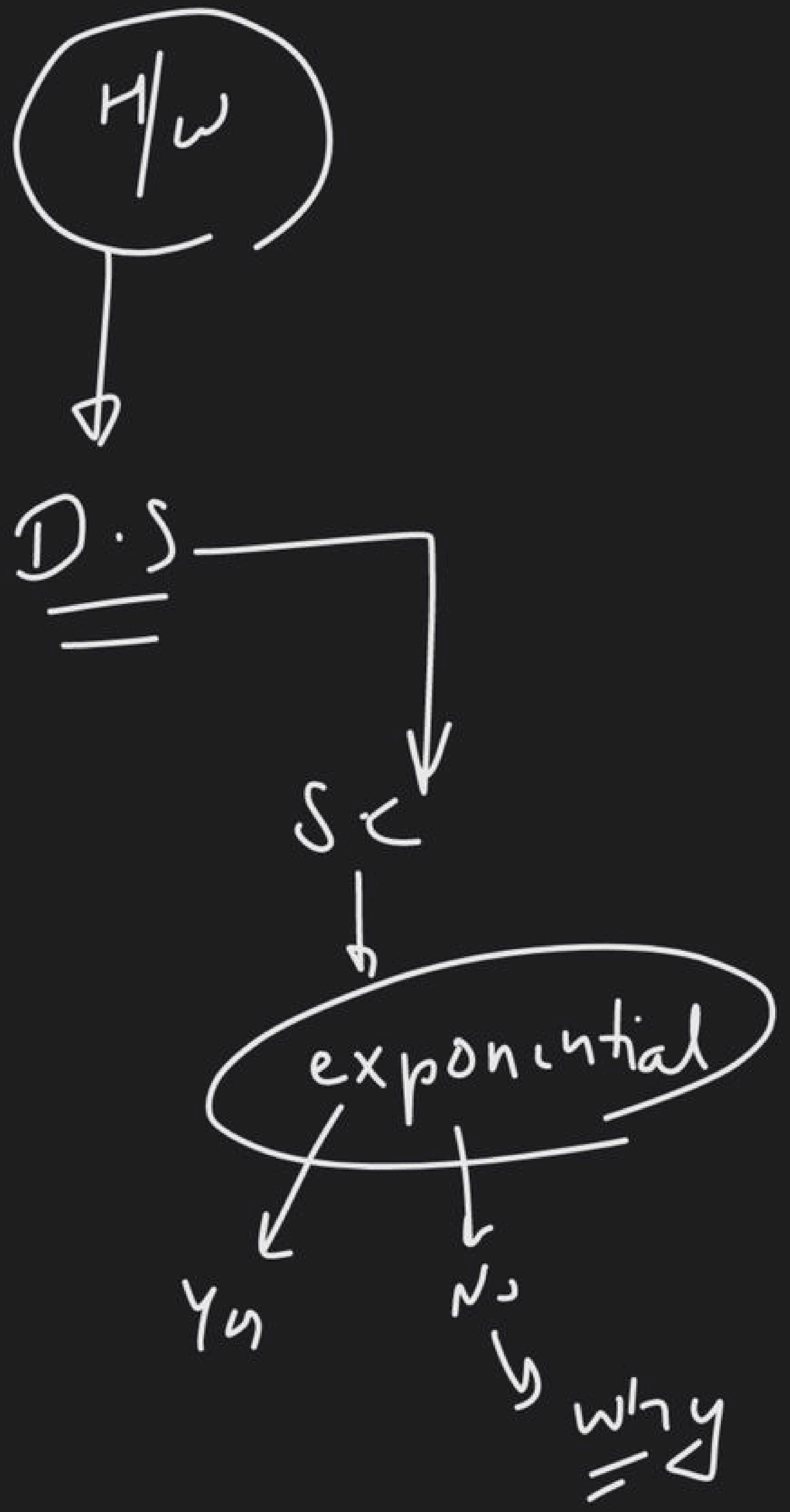
S.C → n@k

= O(n)

O(2^n)

4 pm

	S.C
Counting →	O(n)
fact →	O(n)
B-S →	O(log n)
M.S →	O(n)
fib →	O(n)



H/w

D.S

SC

exponential

Yes

No

why