Time & Space Complexity of Recursive Algorithms - LIVE

Special class

Vishal Pachauri • Sept 17, 2022

→ Time Complexity :-

What?

Time required to run algo as a function of input

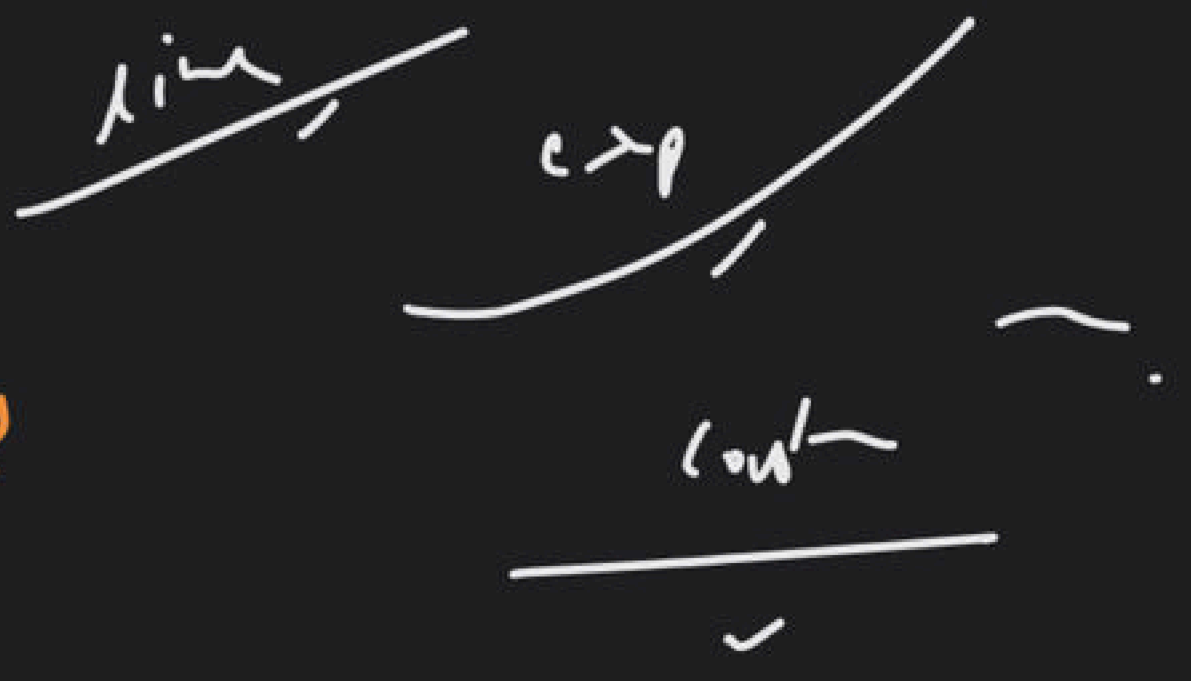
f(n)

①

```
for (int i=0; i<n; i++)
{
    cout << "Babbar";
}
```

$$\left.\begin{array}{l} i=0 \\ i=1 \\ i=2 \\ \vdots \\ i=n-1 \end{array}\right\} \text{n times}$$

T.C → O(n)



1

```
for (int i=0; i<n; i++)      → n times
{
    for (int j=0; j<n; j++)  n times
    {
        cout << ;
    }
}
```

T.C → O(n²)

→ O(n*n)

(3)

```
for (int i=0; i<n; i++)
{
    for (int j= i; j<n; j++)
    {

    }
}
```

i = 0
j → 0 → n

i = 1
j → 1 → n

i = 2
j = 2 → n

i = 4
j → 4 → n

n, n-1, n-2 - - - - 1

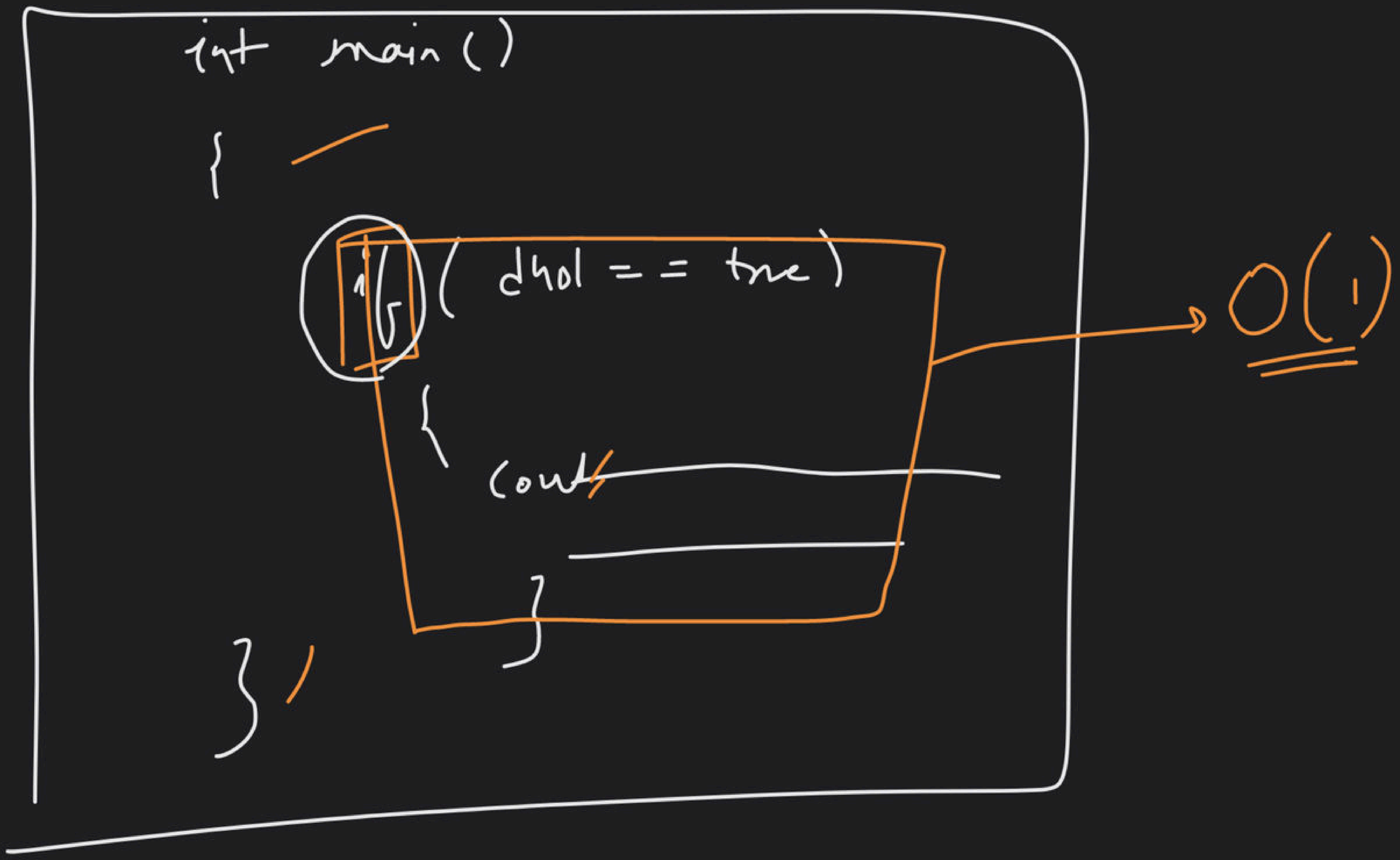
1 + 2 + 3 — — - n → $\boxed{\frac{n \times (n+1)}{2}}$

$$O\left(\frac{n(n+1)}{2}\right)$$

$$O\left(\boxed{\frac{n^2}{2}} + \cancel{\frac{n}{2}}\right)$$

$$O\left(\frac{n^2}{\cancel{2}}\right)$$

$$O(n^2) =$$

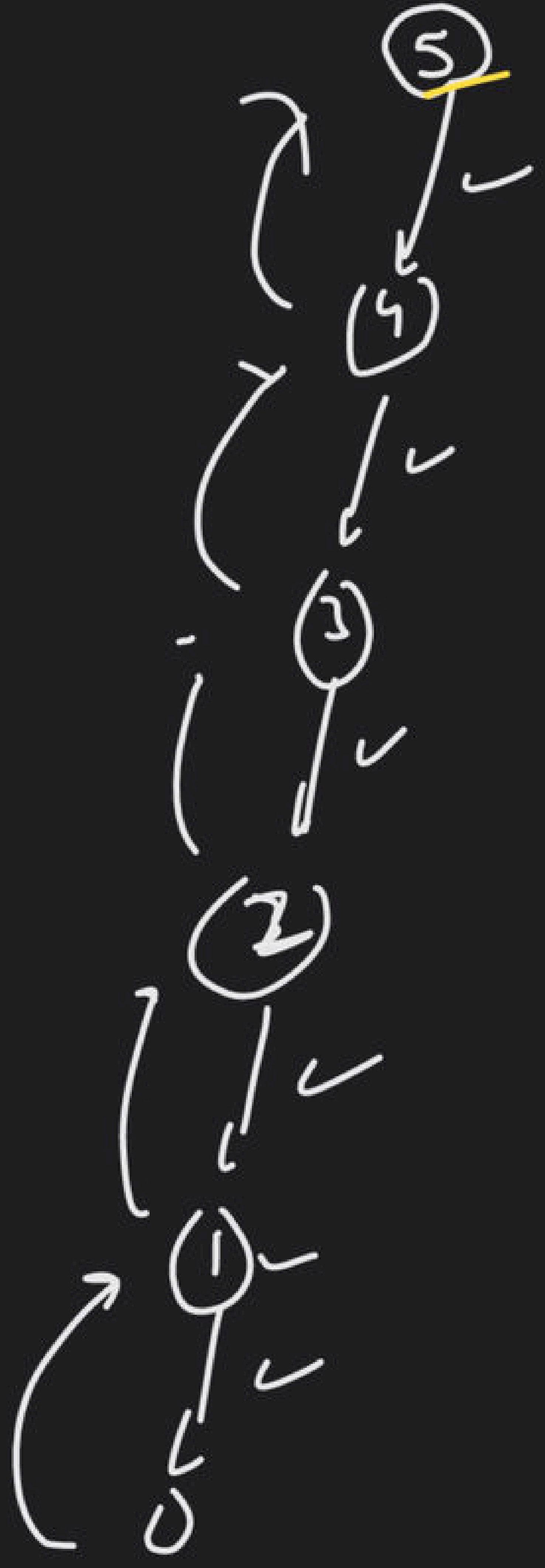

int main()
{
if (dhol == true)
{
cout
}
}
O(1)

① Counting

Recursive relation

$f(n) \longrightarrow f(n-1)$

$$\boxed{T(n) = K_1 + K_2 + T(n-1)}$$



```
void print(n)
{
    B.C        → K1
    Count      → K2
    //R-C
    print(n-1);
}
```

$$T(n) = K_1 + K_2 + T(n-1) \qquad K_1 + K_2 \rightarrow K$$

$$\left.\begin{aligned}
\boxed{T(n)} &= K + \cancel{T(n-1)} \\
\cancel{T(n-1)} &= K + \cancel{T(n-2)} \\
\cancel{T(n-2)} &= K + \cancel{T(n-3)} \\
&\vdots \\
T(1) &= K + \cancel{T(0)}
\end{aligned}\right\} \text{n times}$$

$$\cancel{T(0)} = \boxed{K_1}$$

$$T(n) = n * K + K_1$$

$T(n) = n * k + \cancel{k}$ 1

$T(n) = n * \cancel{k}$

$T(n) = n$

T·C $\longrightarrow$ $O(n)$

2 1

1 1

15 1

→ factorial :-

$$f(n) \longrightarrow f(n-1)$$

```
int factorial (int n)
{
    // B.C
    if (n == 0)
        return 1;

    return n * factorial(n-1)
}
```

$$\boxed{T(n) = K_1 + K_2 + T(n-1)}$$

$$\downarrow$$

$$T(n) = K + T(n-1) \longrightarrow O(n)$$

K_1

$$\boxed{T(n)} = K + \cancel{T(n-1)}$$

$$\cancel{T(n-1)} = K + \cancel{T(n-2)}$$

$$\cancel{T(n-2)} = K + \cancel{T(n-3)}$$

$$\vdots$$

$$\cancel{T(1)} = K + \cancel{T(0)}$$

$$\cancel{T(0)} = K_1$$

n times

$$T(n) = n * K + K_1$$

$T(n) = n * k + k$

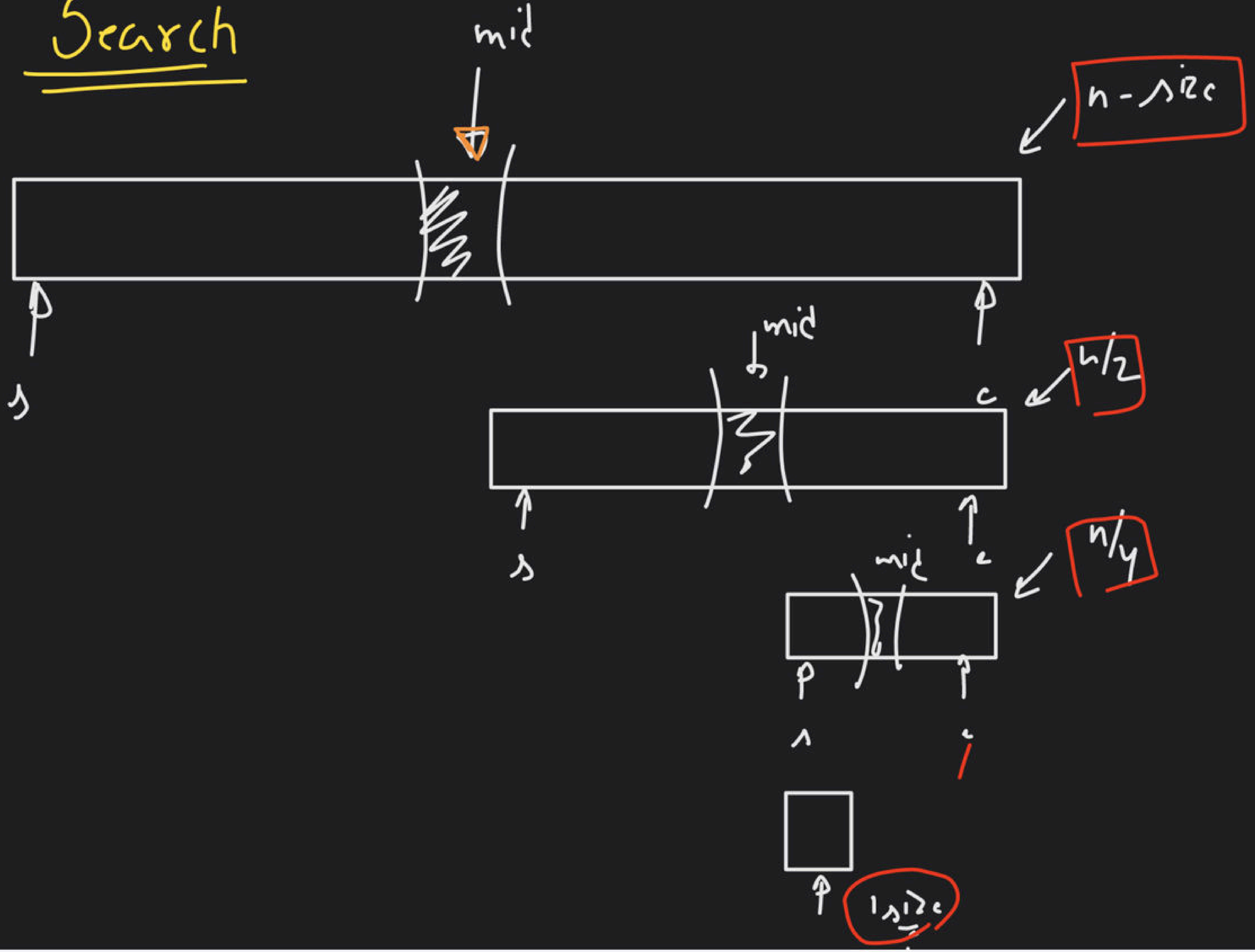
α

$T(n) = n * \cancel{k}$

$T(n) = n$

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

① Binary Search



B.S

```
bool BS ( int arr[], int size, int s, int e)      // e = target
{
    // B.C
    if (s > e)
        return false;                              -> k1
    int mid = (s+e)/2                              -> "
    if (arr[mid] == target)
        return true;                               -> k3
    if (arr[mid] > target)
        return f(arr, size, s, mid-1);             -> T(n/2)
    else
        return f(arr, s-e, mid+1, e)               -> T(n/2)
}
```

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

K

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

$f(n) \rightarrow \frac{n}{2^0} \rightarrow \frac{n}{1}$

$a = ?$

$f\left(\frac{n}{2}\right)$ $\quad \frac{n}{2^1} = \frac{n}{2}$

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

$f\left(\frac{n}{4}\right)$ $\quad \frac{n}{2^2}$

$n = 2^a$

$\log n = a$

$f\left(\frac{n}{8}\right)$ $\quad \frac{n}{2^3}$

$\vdots$

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

$f(1)$ $\quad \frac{n}{2^a}$

$$\boxed{T(n)} = K + T(n/2)$$

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

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

$\vdots$

a times

$$T(1) = K$$

$$T(n) = a * K$$

$$T(n) = a + k$$

$$\alpha$$

$$T(n) = a$$

$$T(n) = \log n$$

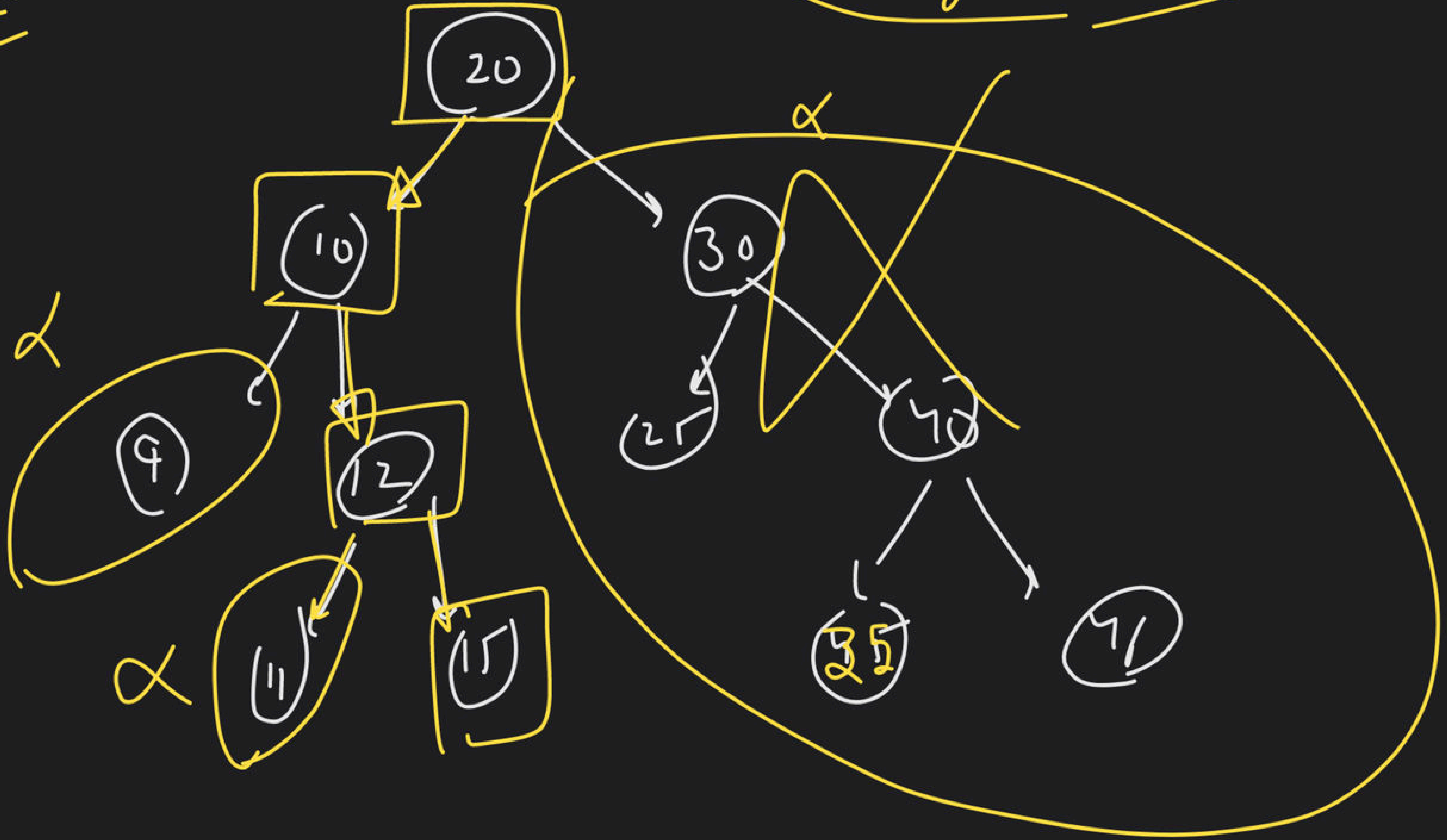
$$O(\log n)$$

BST

Bs

$O(\log n)$

target : 15



1st $B.S(n) \rightarrow \frac{n}{2^0}$

2nd $B.S\left(\frac{n}{2}\right) \rightarrow \frac{n}{2^1}$

3rd $B.S(n/4) \rightarrow \frac{n}{2^2}$

4th $B.S(n/8) \rightarrow \frac{n}{2^3}$

- - - - - - -

$B.S(1) \rightarrow \boxed{\frac{n}{2^a}}$

a^{th} call

- Total calls → "a" calls

Eg

$2 \times 2 = 4$

$16 = 2^x$

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

$n = \boxed{2}^a$

$\log n = a$

$\boxed{n = 2^a}$

$\boxed{\log(n)} = \boxed{\log(2^a)}$

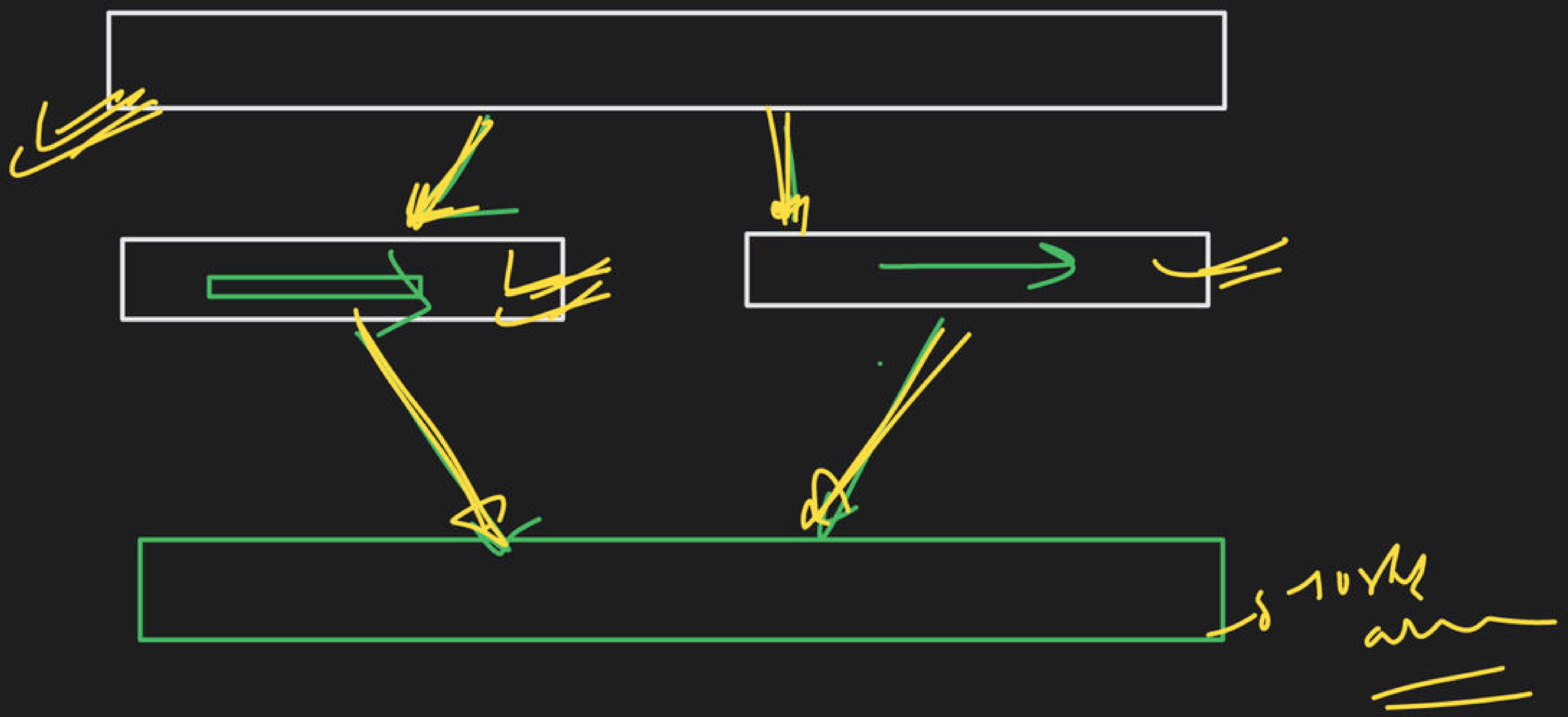
$\log n = \boxed{a} \times \boxed{\log(2)} \longrightarrow \log n = a$

Co —— $O(n)$

fact $\nearrow$

B.S $\longrightarrow$ $\underline{\underline{O(\log n)}}$

Merge Sort:-



→ Merge Sort :-

$$T(n) = K_1 + K_2 + \overset{\text{left}}{T\left(\frac{n}{2}\right)} + \overset{\text{right}}{T\left(\frac{n}{2}\right)} + K_3 n + K_4 n$$

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

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

$K_5 = p$

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

n $\quad T(n) = 2T\left(\frac{n}{2}\right) + n * p$

$n/2$ $\quad 2T\left(\frac{n}{2}\right) = 4T\left(\frac{n}{4}\right) + \frac{n}{2} * p$

$n/4$ $\quad 4T\left(\frac{n}{4}\right) = 8T\left(\frac{n}{8}\right) + \frac{n}{8} * p$

$1 \quad 1 \quad 1$

$1 \quad 1 \quad 1$

$\log n$ $\quad 1$ $\quad T(1) = 2T(0) \rightarrow p_1$

$n * p$

$T(n) = a * n * p + \cancel{p}$

$T(0)$

a times

$a = \log n$

$T\left(\frac{1}{1}\right) \rightarrow T0$

$2^{\log n}$

$$T(n) = a * n * \cancel{p}$$

$$T(n) = n * a \qquad a = \log n$$

$$T(n) = n * \log n$$

$$T.C \rightarrow O(n \cdot \log n)$$

fib

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

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

$\to O(1)$

$O(1)$

$O(1)$

$T(n) = K_1 + T(n-1) + T(n-2)$

$$T(n) = K + T(n-1) + T(n-2) \quad \alpha$$

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

$$T(n-2) = K + T(n-3) + T(n-4) \quad \alpha$$

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

$$T(n-3) = K + T(n-4) + T(n-5) \quad \alpha$$

$$T(n \quad \alpha$$

$\vdots$

α

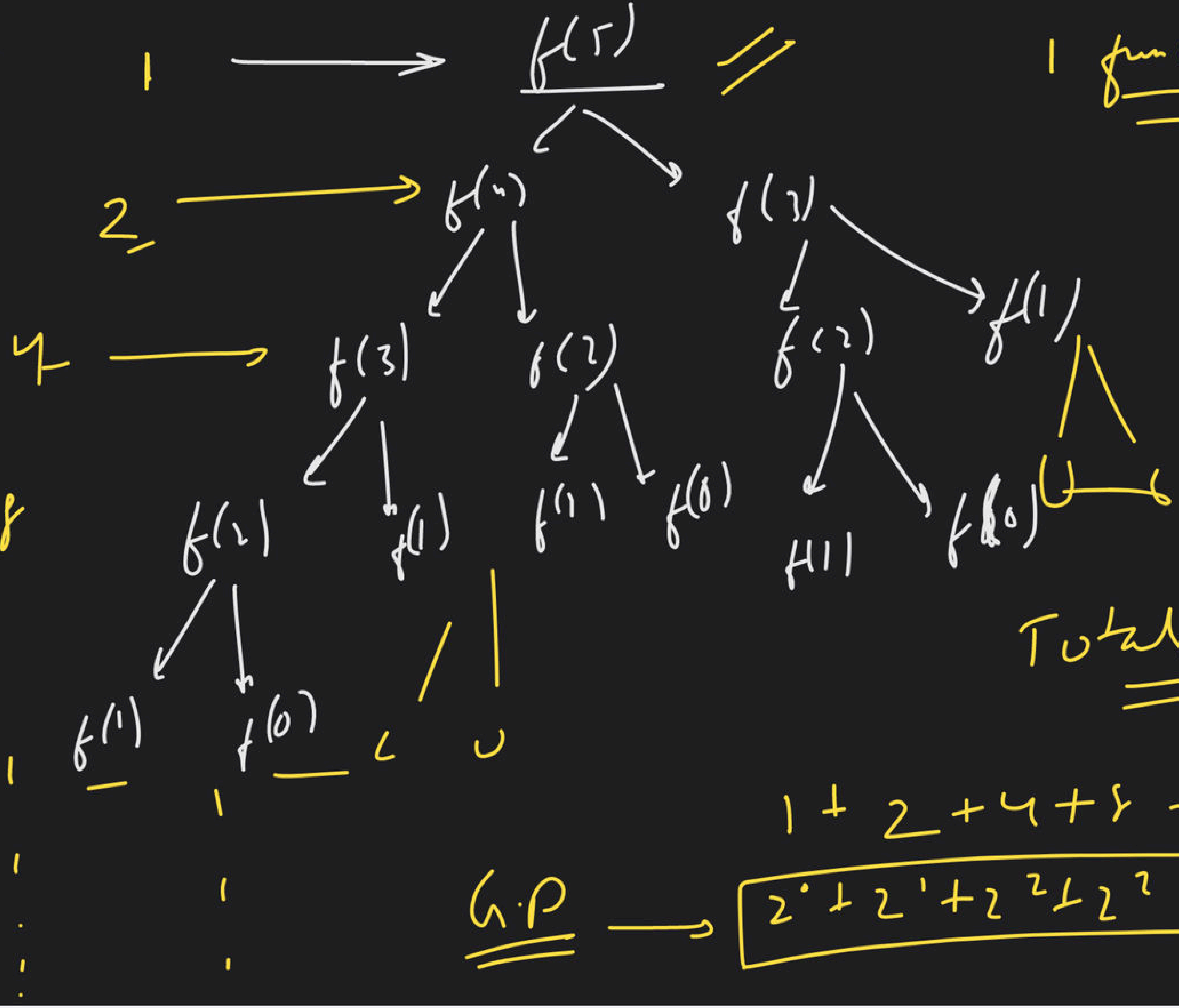
2^0 1

2^1 2

2^2 4

2^3 8

2^4 16



1 functio n time → $\underline{\underline{O(1)}}$

Total func calls

$1 + 2 + 4 + 8 - - - - 2^n$

$\underline{\underline{G.P}} \rightarrow \boxed{2^0 + 2^1 + 2^2 + 2^3 - - 2^n}$

$$\frac{2^{n+1}-1}{2-1} = \frac{2^{n+1}-1}{1}$$

$$2^{n+1}-1$$

$$2^n \times 2^1$$

$$O(2^n)$$

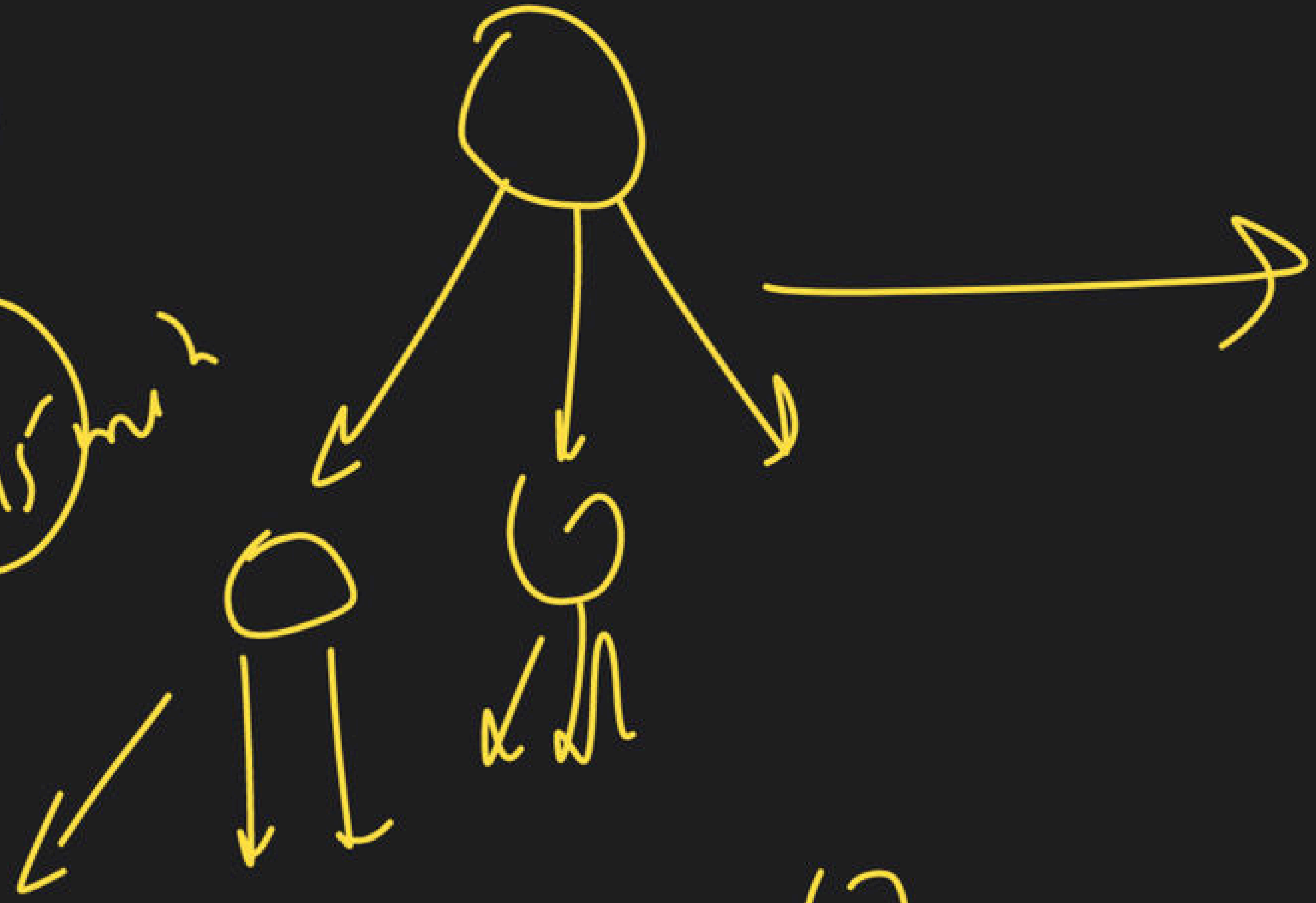
exponential

$$T \cdot C \to O(2^n)$$

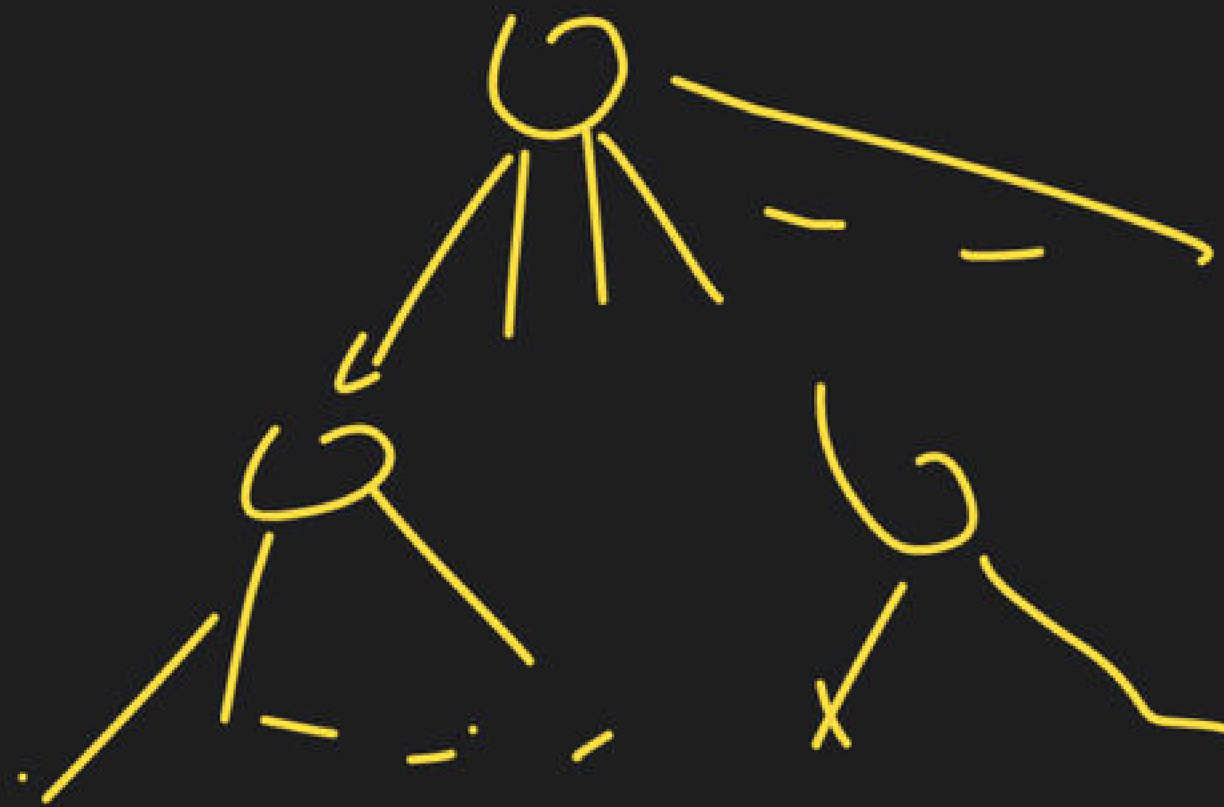
Master's theoram

H/W

1/10/15 min



3^4



n^4

$$2^{a-1}$$

$$\longrightarrow \quad T(1) = 2T(0)$$

$$2^{a-1} * (T(1)) = 2^a * \boxed{T(0)}$$

$$= 2^a$$

$$T(n) = ank + 2^a$$

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

$$\boxed{T(1) = 2T(0)}$$

M.S → S.C → '12-

$n = 2^{10}$

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

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

$= \underline{2^{10} \times 10} + 2^{10}$

4pm - 6pm