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

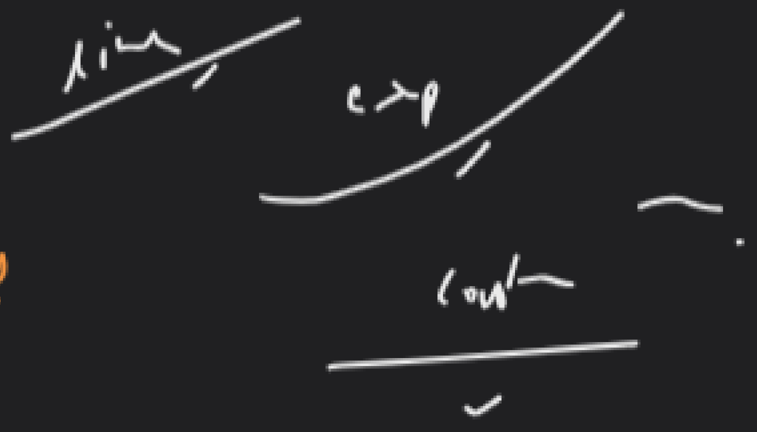
$f(n)$

①

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

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

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



①

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

n+1 → n

T.C → $O(n^2)$

→ $O(n*n)$

③

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

$i = 0$
$j \to 0 \to n$

$i = 1$
$j \to 1 \to n$

$i = 2$
$j = 2 \to n$

$i = 4$ ——
$j \to 4 \to n$

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

$1 + 2 + 3$ —— — . $n \to \boxed{\frac{n \times (n+1)}{2}}$

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

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

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

$$O(n^2)$$

```
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);
}
```

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

$K_1 + K_2 \to K$

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

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

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

$$\vdots$$

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

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

n times

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

$T(n) = n * k + \cancel{k_1}$

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

$T(n) = n$

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

2
1
15

→ factorial :-

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

$\to K_1$

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

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

$\downarrow$

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

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

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

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

$$\vdots$$

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

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

n times

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

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

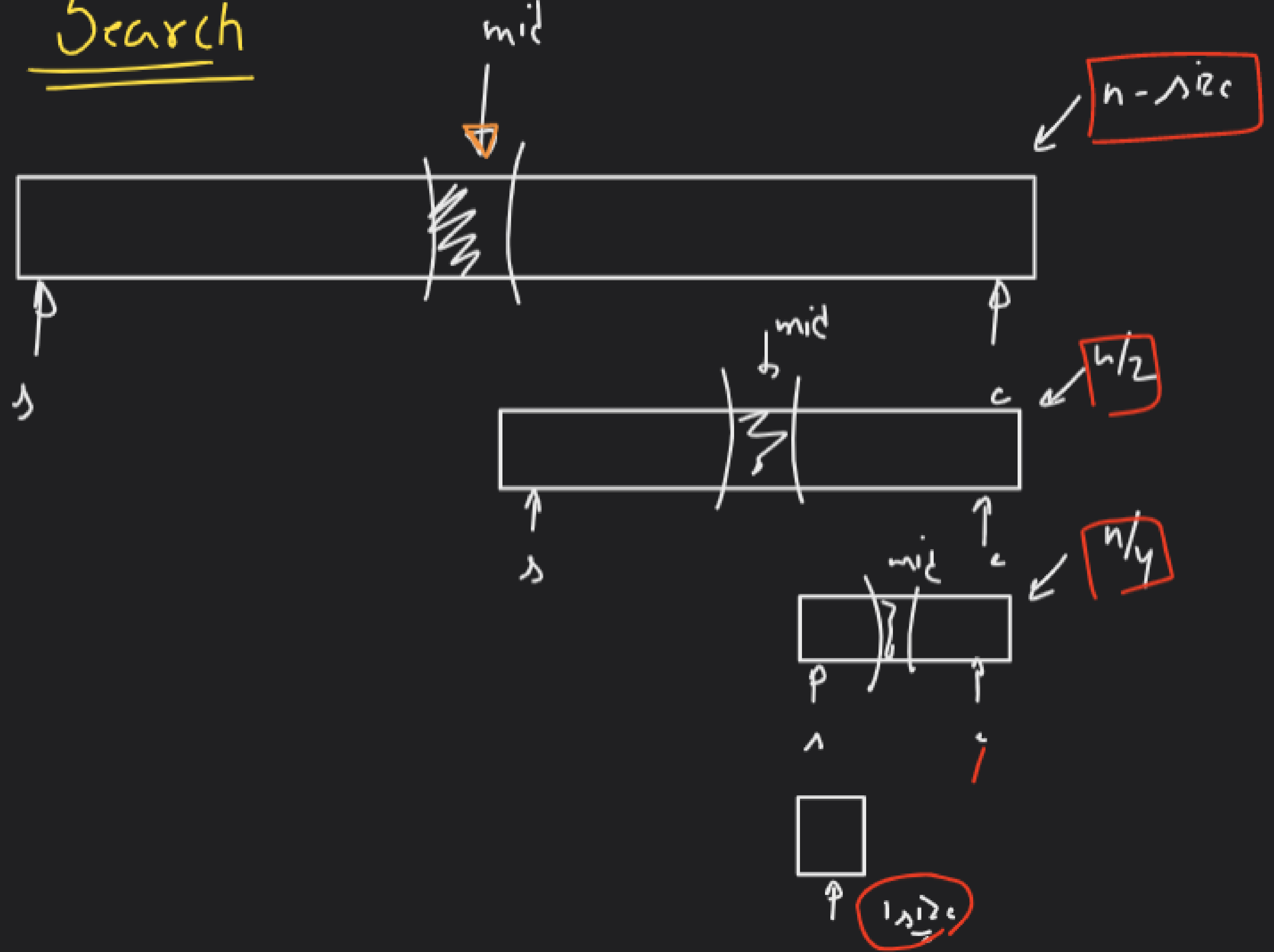
α

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

$$T(n) = n$$

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           → k2
  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, size, mid+1, e);   → T(n/2)
}
```

$$\boxed{T(n) = k_1 + k_2 + k_3 + T\left(\frac{n}{2}\right)}$$

K

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

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

$a = ?$

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

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

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

$n = 2^a$

$\log n = a$

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

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

$\vdots$

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

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

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

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

$\vdots$

a times

$$T(1) = k$$

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

$T(n) = a \cancel{x} k$

d

$T(n) = a$

$T(n) = \log n$

$O(\log n)$

BST

Bs

$O(\log n)$

target : 15

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

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

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

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

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

a^{th} call

- Total calls → "a" calls

Ex: $2 \times 2 = 4$

$16 = 2^x$

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

$n = \boxed{2}^a$

$\log n = a$

$\boxed{n = 2^a}$

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

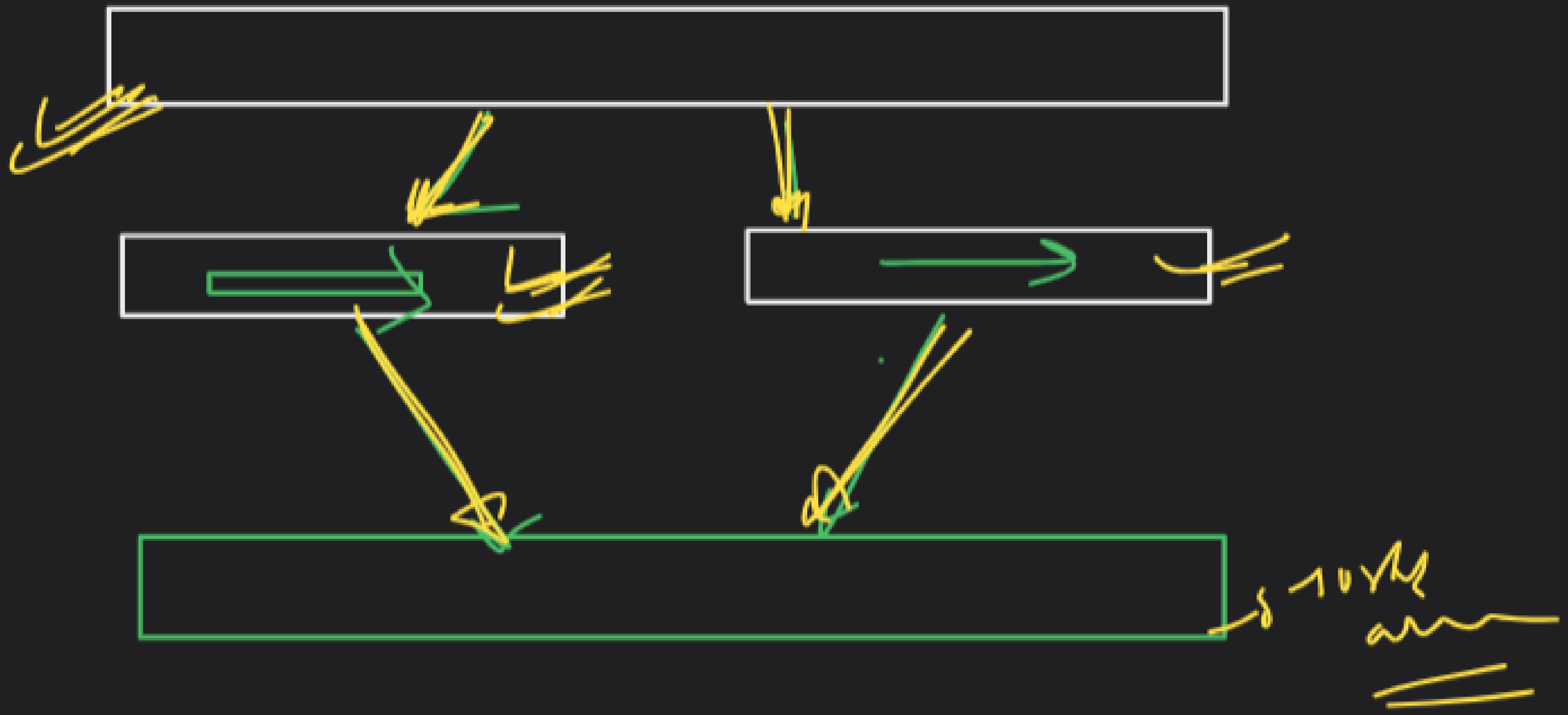
$\log n = \boxed{a}\,\boxed{\log(2)} \rightarrow \log n = a$

Co —— $O(n)$

fact $\nearrow$

B.S $\longrightarrow$ $O(\log n)$

Merge Sort:-



→ Merge Sort :-

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

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

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

$K_5 = p$

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

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

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

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

n

$n/2$

$n/4$

1

1

1

$\log n$

$1 \quad 1 \quad 1$

$1 \quad 1 \quad 1$

$n * p$

$$T(1) = 2T(0) \rightarrow p_1$$

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

$T(0)$

→ a times

$a = \log n$

$T\left(\frac{1}{1}\right) \rightarrow 16$

$2^{\log n}$

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

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

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

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

fib

```
int fib(int n)
{
    //logic
    if (n==0 || n==1)
        return n;
    return fib(n-1) + f(n-2);
}
```

$\to O(1)$

$O(1)$ $O(1)$

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

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

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

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

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

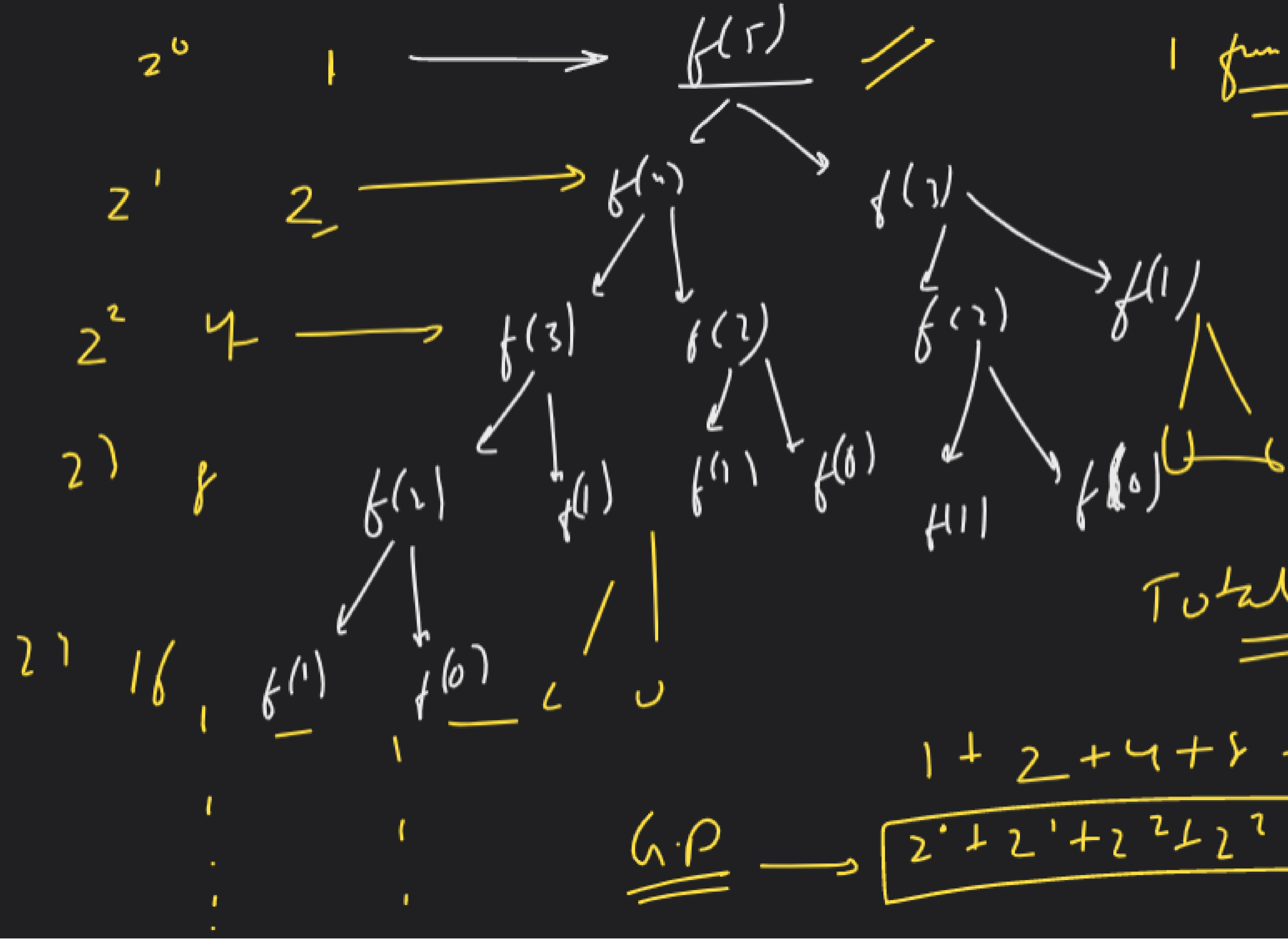
$$T(n-3) = K + T(n-4) + T(n-5) \quad \alpha$$

$T(n$

$\vdots$

α

α



1 function in time

$\hookrightarrow O(1)$

Total func calls

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

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

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

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

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

$$O(2^n)$$

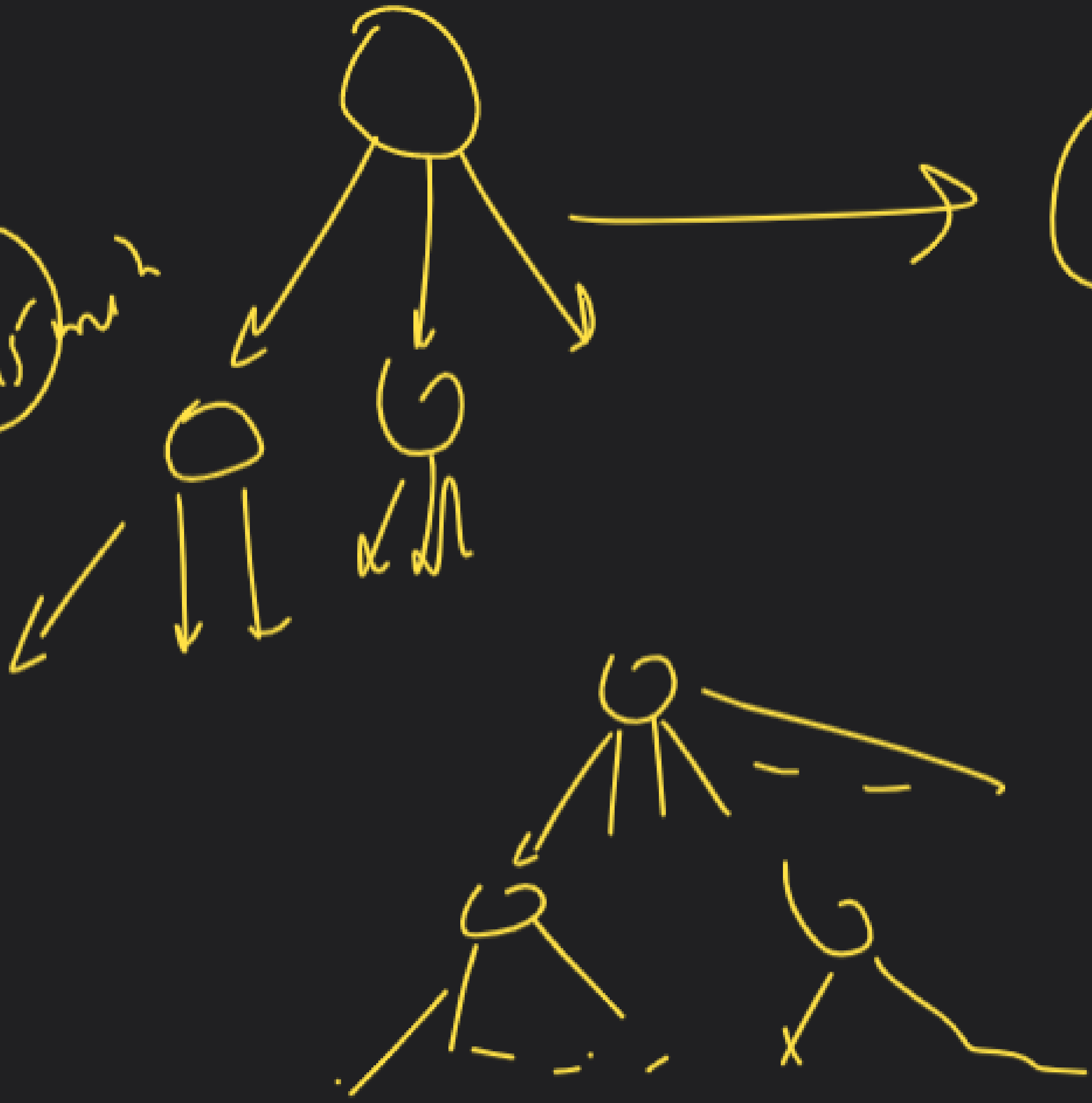
exponential

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

Master's theoram

H/W

10/15 min



3^n

n^n

2^{a-1}

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

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

$$= 2^a$$

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

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

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

M.S $\rightarrow$ S.C $\rightarrow$ '12-

$n = 2^{10}$

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

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

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

4pm - 6pm