CS & IT ENGINEERING

Algorithms

Divide & Conquer

Lecture No.- 01

By- Dr. Khaleel Khan Sir

Recap of Previous Lecture

Topic Analysis of Algorithms

Topic Time and Space Complexities

Topic

Topic

Topic

Topics to be Covered

Topic Control Abstraction

Topic Min – Max Problem

Topic

Topic

Topic

```
for (i = 2; i ≤ n; i = $i^2$)
    for (j = 1; j ≤ n; j++)
        for (k = 1; k ≤ n; k = k + j)
            x = y+z;
```

LogLog n

$\longrightarrow \frac{n}{j}$

$j=1 \quad j=2$

$K = \frac{n}{1} \quad \frac{n}{2} \quad \frac{n}{3} \quad \cdots \quad \frac{n}{n}$

$$n \cdot \boxed{\sum_{x=1}^{n} 1/x} = n \log n$$

```
for(i=1; i<=n; i += a)
```

$\frac{n}{a}$

$$\boxed{\Longrightarrow T(n) = O\left(\text{LogLog}n \cdot (n \cdot \log n)\right)}$$

**

```
for i ← 1 to (n-1)
    fo j ← i+1 to n
        for K ← 1 to j
            print("*");
```

What is the No. of Times '*' gets printed

Design Strategies

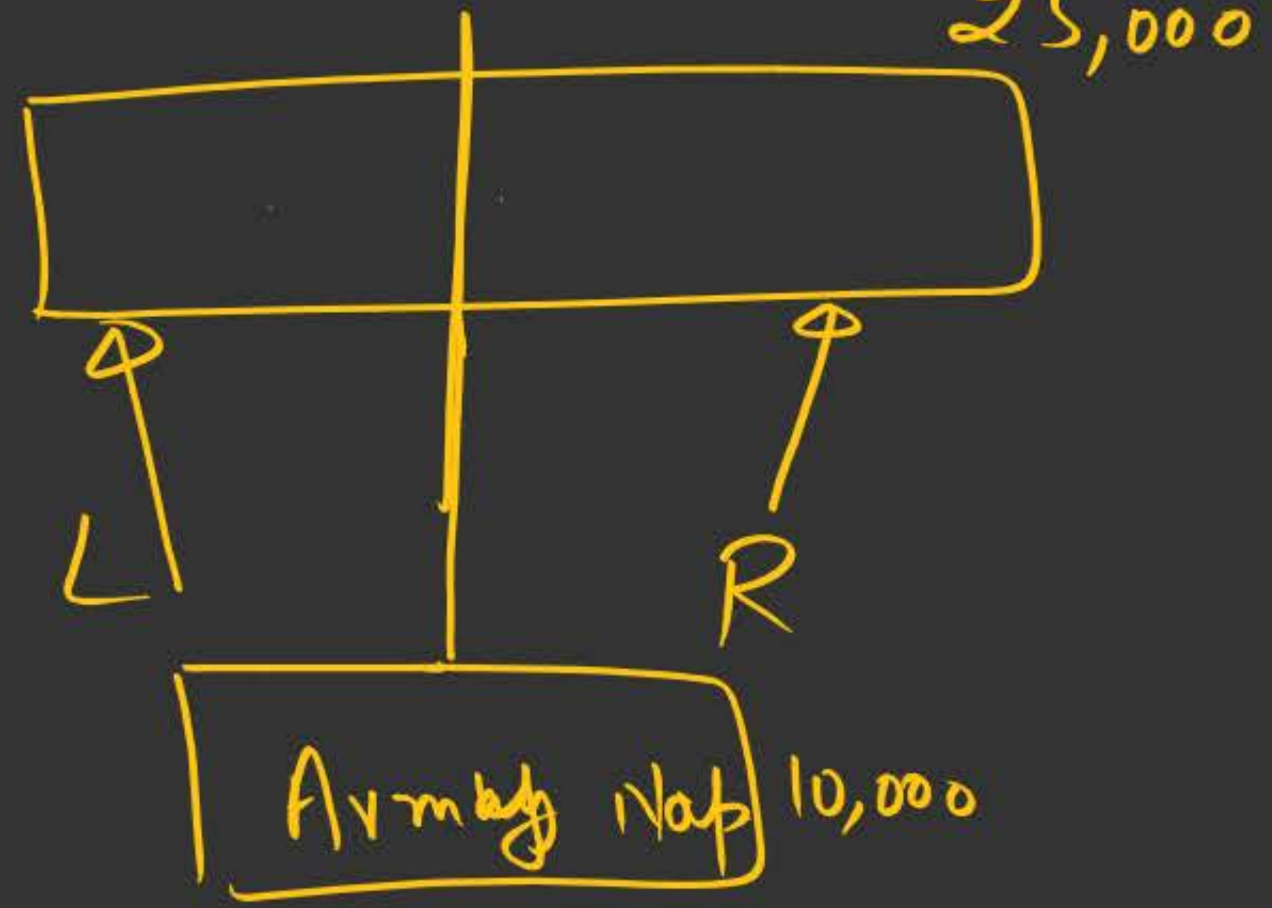
Divide & Conquer : (DandC)

⟹ When the Problem becomes large/Complex, then divide the problem into Subproblems, into Further Subproblems, until the Subproblem becomes Small,

⟹ Solve the Smaller problems, Combine their results if required to get the Solution of original Problem;

⟹ In general a problem is said to be Small, if it Can be solved in one/two basic operations;

⇒ Napoleam



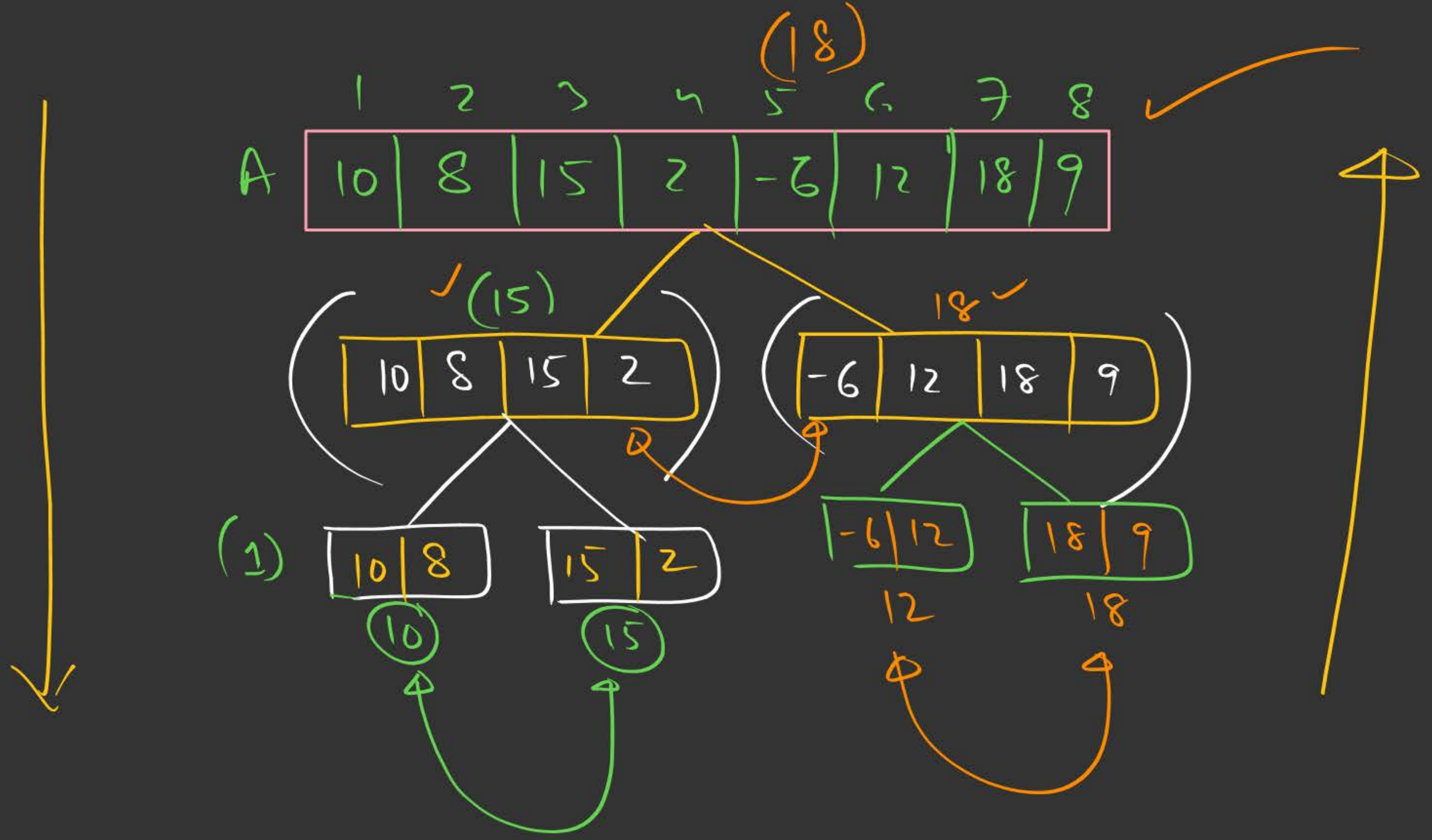
(ii) Britishers

(Divide & Rule)

(GATE Problem)

D and C

Problem → (Input)


(18)
1 2 3 4 5 6 7 8
A
10 8 15 2 -6 12 18 9
(15)
18
10 8 15 2
-6 12 18 9
(1)
10 8
15 2
-6 12
18 9
10
15
12
18

Topic : Divide and Conquer

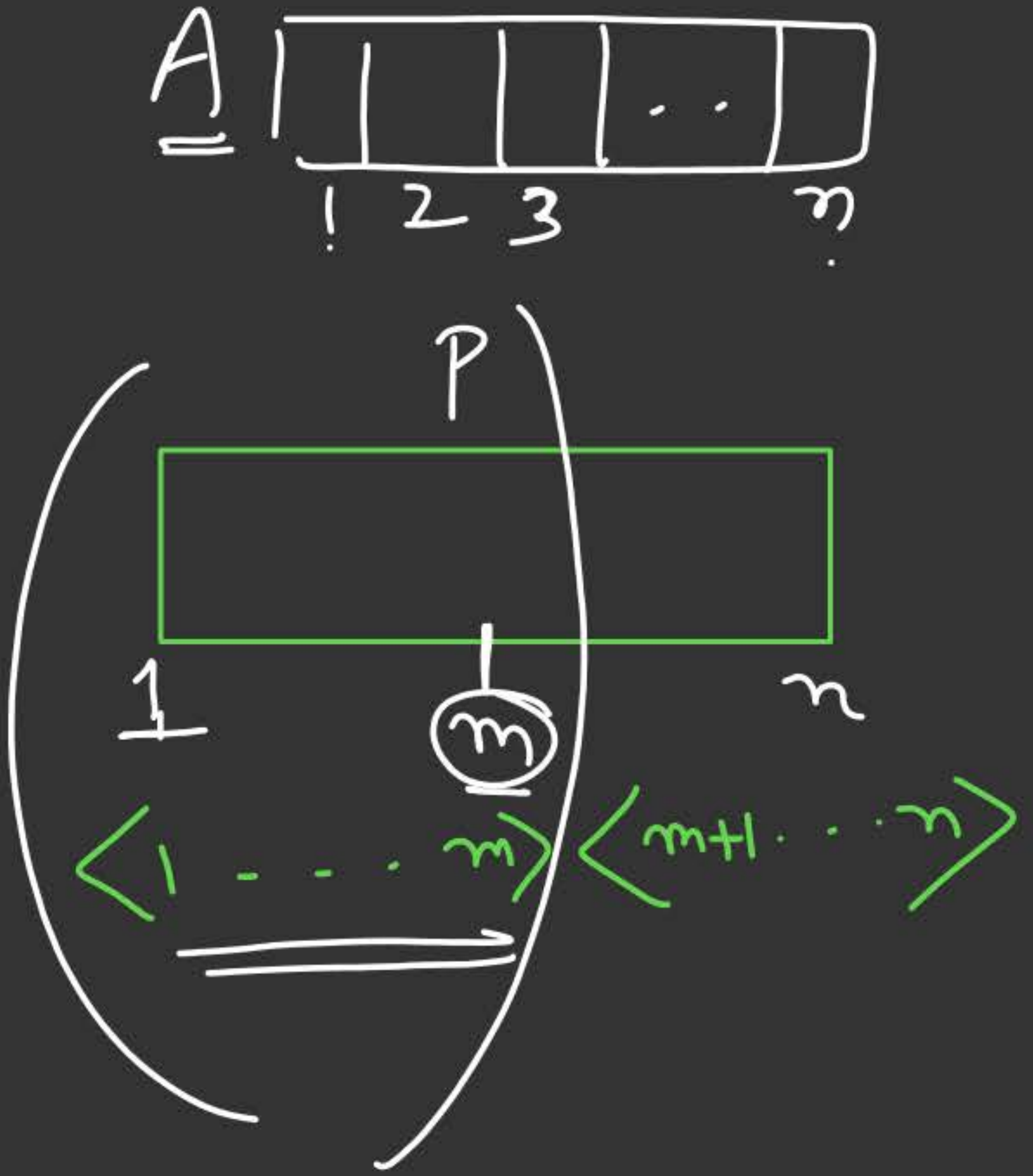
Control Abstraction

```
Algorithm DAndC(P)
{
    if Small(P) then return S(P);
    else
    {
        divide P into smaller instances $P_1, P_2, \dots, P_k$, $k \geq 1$;
        Apply DAndC to each of these subproblems;
        return Combine(DAndC($P_1$), DAndC($P_2$), ......., DAndC($P_k$));
    }
}
```

(recursion)

```
Algorithm DandC(A, 1, n)
{
    if (SMALL(1, n))
        return (S(A, 1, n));          f(n);
    else
    {
        m ← DIVIDE(1, n)
        s1 ← Dandc(A, 1, m);          n/2
        s2 ← Dandc(A, m+1, n);        n/2

        COMBINE(s1, s2);
    }
}
```



Time Complexity (framework) for DandC Problems

→ Let T(n) repr. Time Complexity of DandC(n);

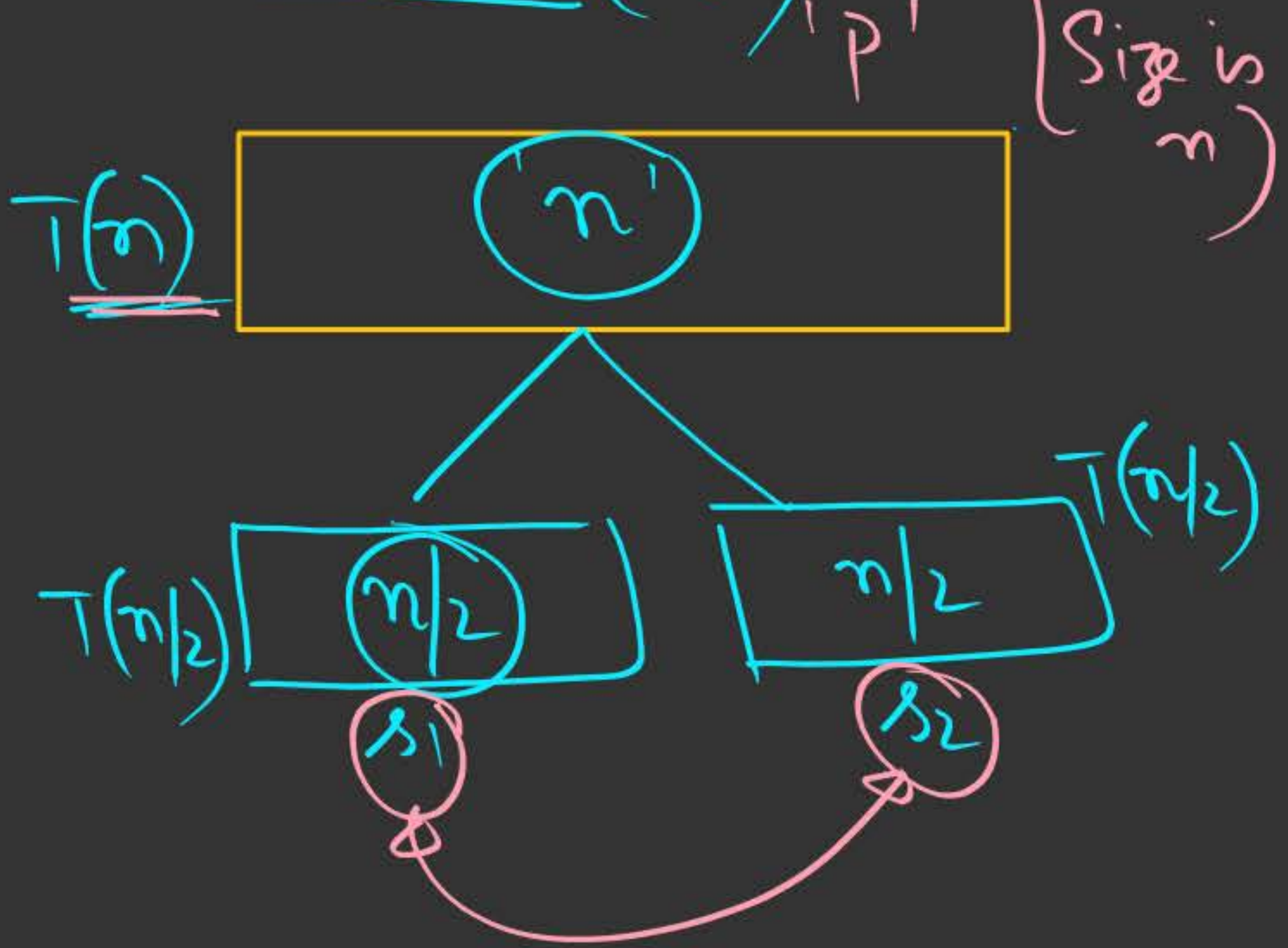
$$T(n) = f(n) \quad , \text{if 'n' is } \underline{\text{Small}}$$

$$= 2 \cdot T(n/2) + \underline{g(n)} \quad , \text{if 'n' is large}$$

[$f(n) = \text{order}(S())$;
↳ $\underline{\text{Constant}}$ $(O(1))$

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

$g(n) = \text{Time}($ Divide + COMBINE + SMALL $)$



$$T(n) = 2\cdot T(n/2) + g(n) \longrightarrow \text{Divide \& Conquer recurrence}$$

No. of Subproblems

Size of each Problem

Time for Divide & Combine

Generalized Form :

$$\boxed{T(n) = a\cdot T(n/b) + g(n)} \checkmark$$

No. of Subproblems

Size of each Subproblem

$a \geq 1$

$b > 1$

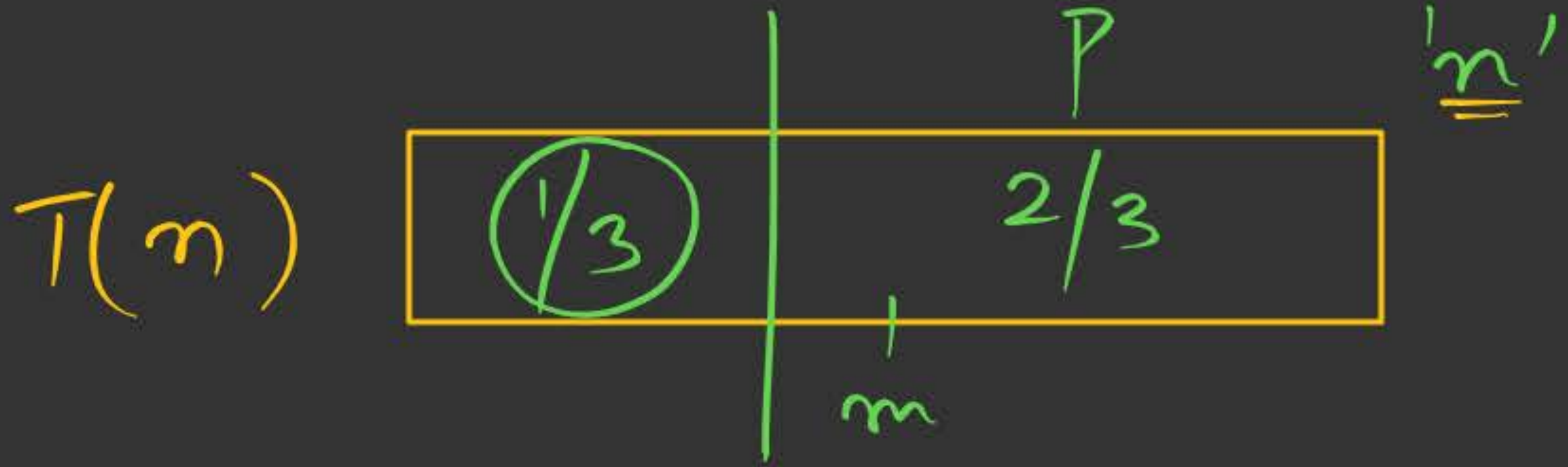
$g(n)$ is +ve

(i) Symmetric

Ex's 1) $T(n) = 2T(n/2) + g(n)$

2) $T(n) = T(n/2) + g(n)$

3) $T(n) = 4T(n/2) + g(n)$



Asymmetric Case

$$T(n) = T(n/3) + T(2n/3) + g(n)$$

I

General form:

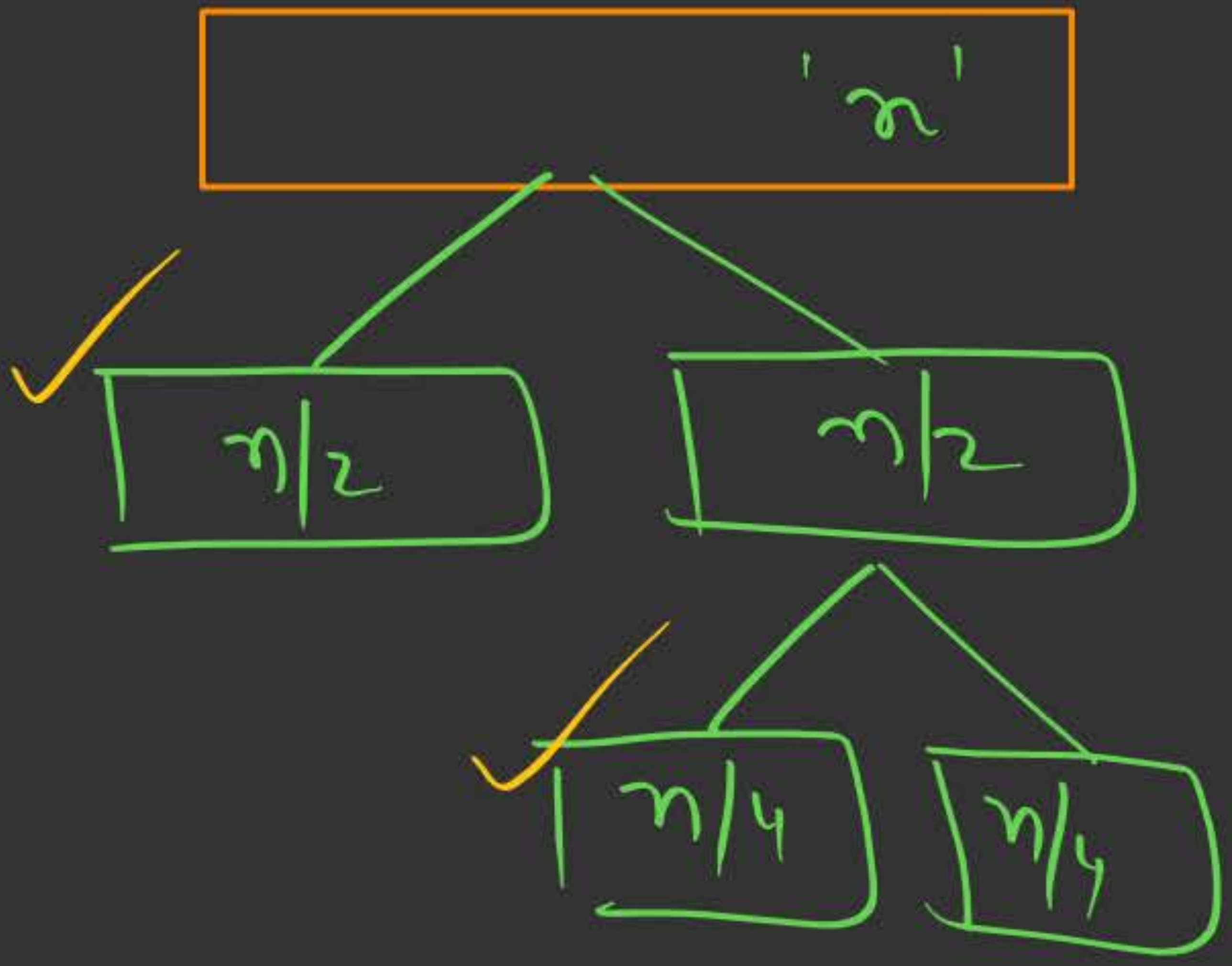
$$\boxed{T(n) = T(\alpha n) + T((1-\alpha)n) + g(n)}$$

$$0 < \alpha < 1$$

Asymmetric case II:

Ex:

$$\boxed{T(n) = T(n/2) + T(n/4) + g(n)}$$



Note 1 : In DandC Strategy, Divide is Mandatory but Combine is optional (depends on applications)

Topic : Divide and Conquer

Computing time of DAndC is described by the recurrence relation

$$T(n)=\begin{cases} g(n) & : n \text{ is Small} \\ \underline{T(n_1)}+\underline{T(n_2)}+\ldots+\underline{T(n_k)}+f(n) & \cancel{n \text{ small}} \text{ otherwise} \end{cases}$$

1) **MaxMin:** Procedure to find Max & Min (Simultaneously) in an array of size 'n';

$n = 10$ 9

$\langle 1 \; 2 \; - - - - - \; 10 \rangle$

5 ✓

```
Algorithm Non-DC (A, n, max, min)
{
    1. max ← min ← A[1];
    2. for i ← 2 to n   (n-1)
       {
           if (A[i] > max) : 1
               max ← A[i];
           else
           if (A[i] < min) : 1
               min ← A[i];
       }
}
```

~~5|8|9|10|12~~

	1	2	3	4	5	6	7	8	9	10
A	8	16	0	-5	18	2	9	60	30	25

Comparison < Index, Element ✓

⟹ The total # of Element Comparisons made in Non-DC = $2 * (n-1)$ [ALL]

(i) Best Case: Inc. order : $(n-1)$ ✓

(ii) worst Case: dec. order : $2(n-1)$ ✓

(iii) Avg. Case: on an avg. first Comparison fails for ½ of given Elem's (n)

$$(n-1) + \frac{n}{2} = \boxed{\frac{3n}{2} - 1}$$

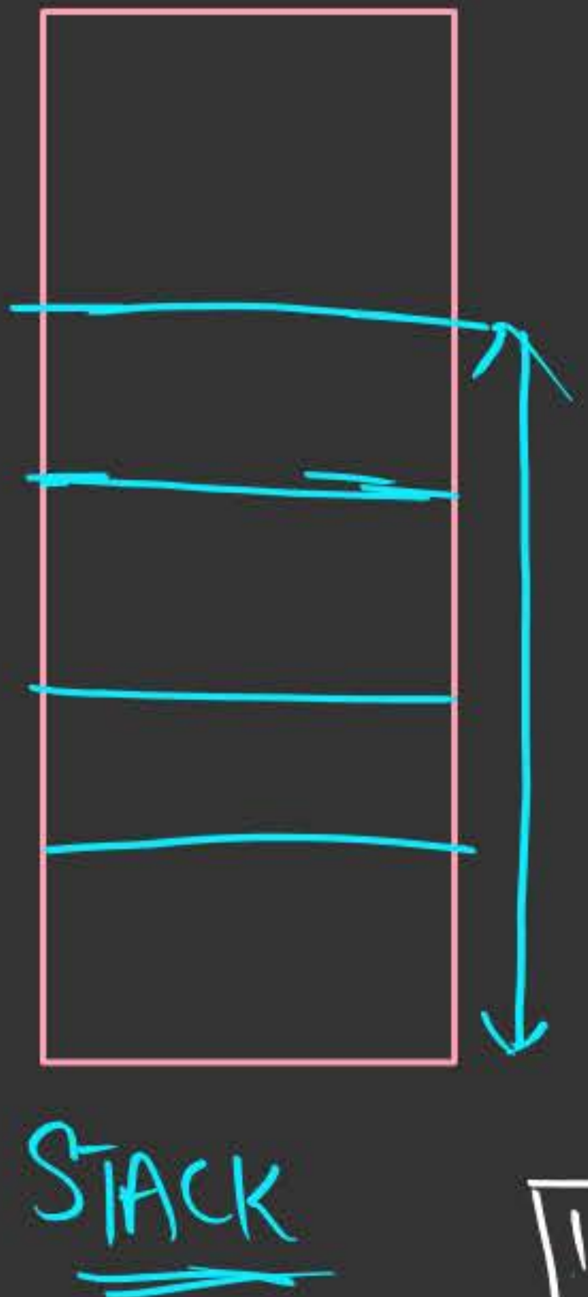
Space: $O(1)$

	1st Comps	2nd Comps	Total
Inc order 1) Best Case	$(n-1)$	0	$(n-1)$
2) worst Case dec order	$(n-1)$	$(n-1)$	$2(n-1)$
3) Avg. Case	$(n-1)$	$n/2$	$\left(\frac{3n}{2}-1\right)$

Time : $\underline{O(n)}$

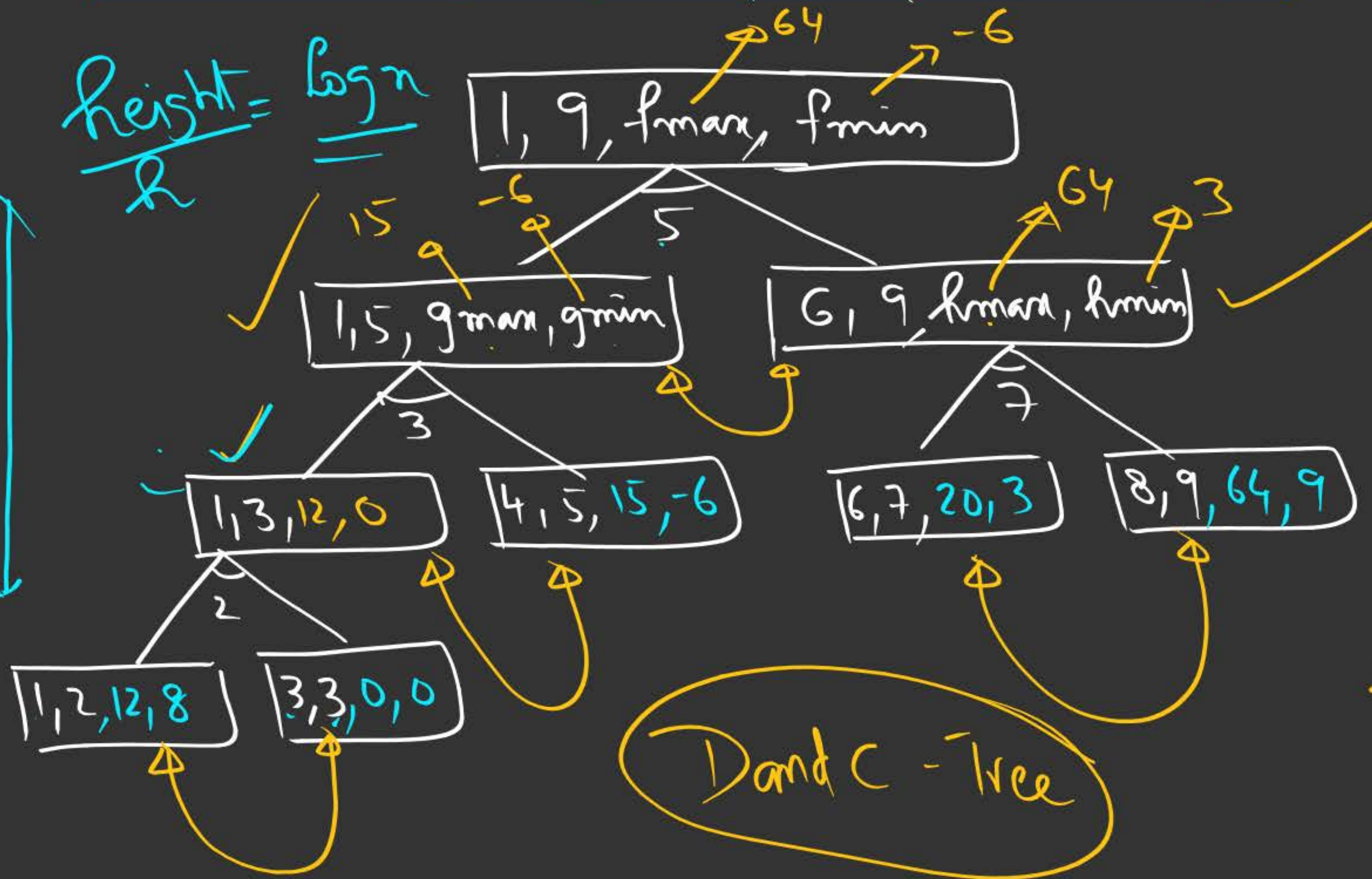
II. DandC - MAXMIN

	1	2	3	4	5	6	7	8	9
A	8	12	0	-6	15	20	3	64	9



height = Logn

h



DandC - Tree

Stack Space:

(Height of the Tree)

h = (Logn)

Topic : Divide and Conquer

DandC l, h

```
Algorithm MaxMin (i, j, max, min)
// a[1 : n] is a global array. Parameters i and j are integers.
{
        If (i = j) then max: min : = a[i]; // Small (P)
        else if (i = j –1) then // Another case of Small (P)
            {
                if (a[i] < a [j]) then
                {
                    max : = a[j]; min : = a[i];
                }
```

Small : 1

Topic : Divide and Conquer

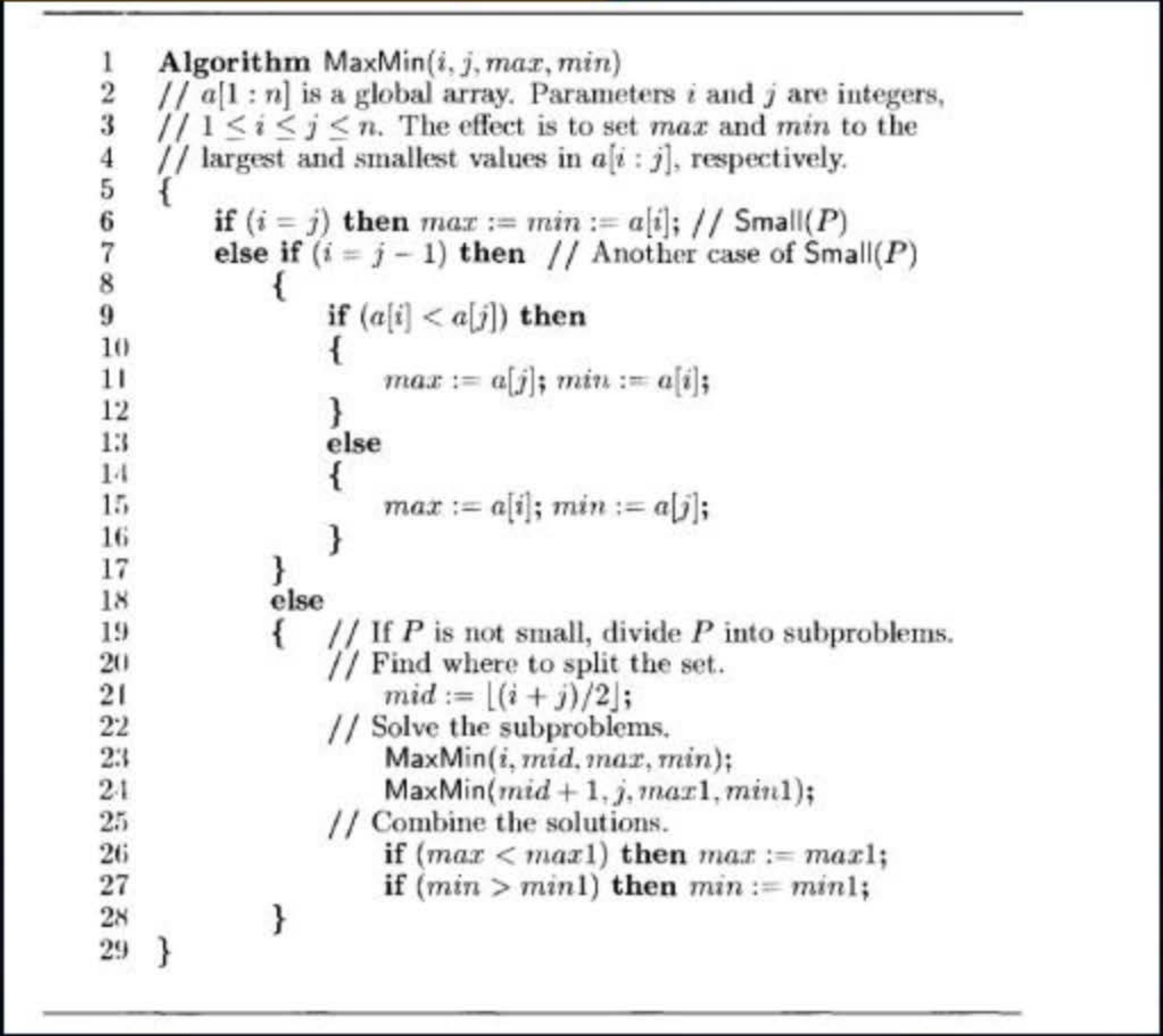
```
    else
    {
        max: = a[i]; min : = a[j]
    }
}
```

Topic : Divide and Conquer

```
else {        Problem is large

        mid: = $\lfloor (i+j)/2 \rfloor$;
        MaxMin (i, mid, max, min);
        MaxMin (mid + 1, j,max1, min1);
Combine (If (max< max1) then max: =max1;
         If (m~~ax~~in> min1) then min: = min1;)
    }
}
```

Topic : Divide and Conquer



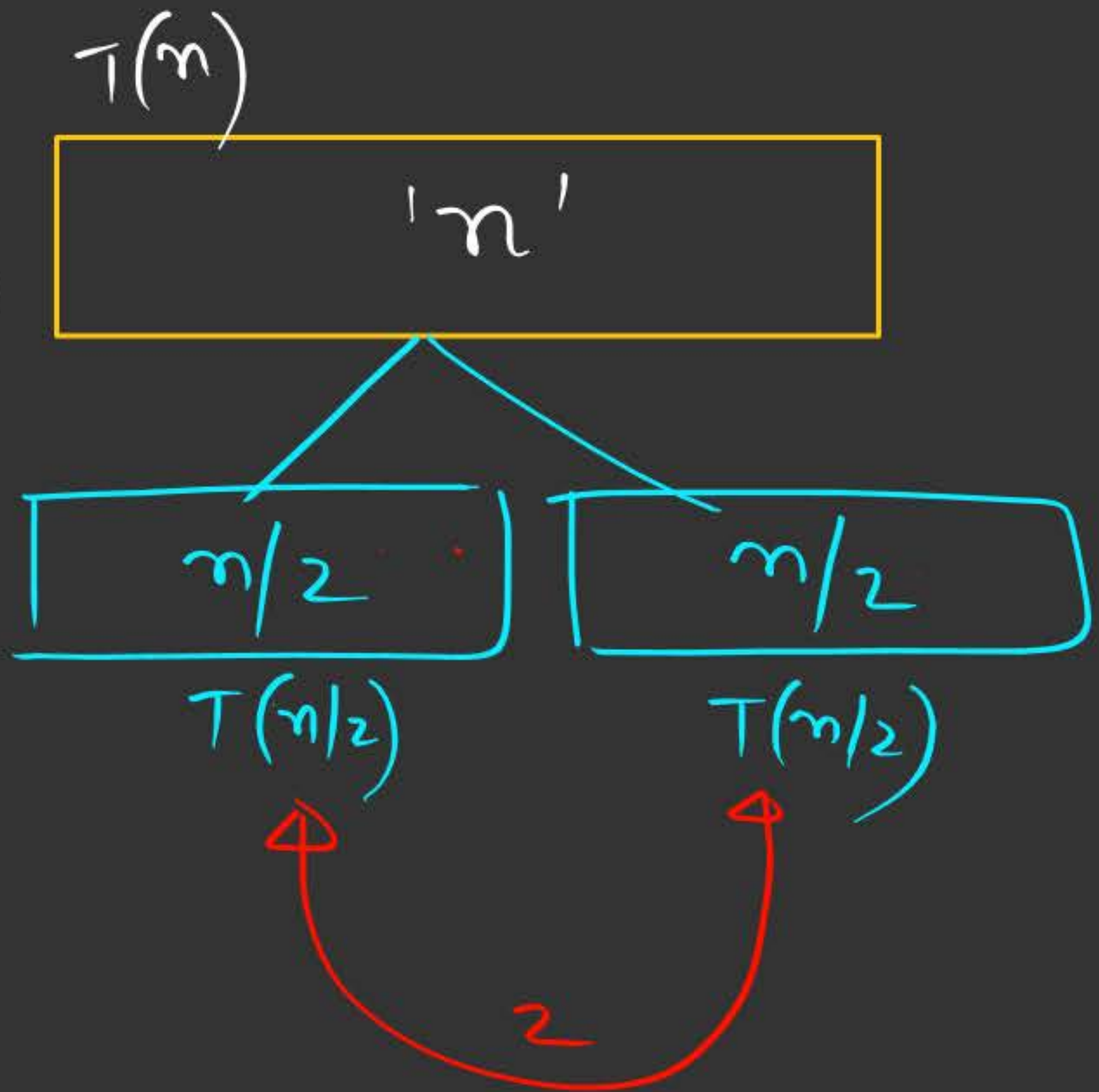
```
Algorithm MaxMin($i, j, max, min$)
// $a[1 : n]$ is a global array. Parameters $i$ and $j$ are integers,
// $1 \leq i \leq j \leq n$. The effect is to set $max$ and $min$ to the
// largest and smallest values in $a[i : j]$, respectively.
{
    if ($i = j$) then $max := min := a[i]$; // Small($P$)
    else if ($i = j - 1$) then // Another case of Small($P$)
        {
            if ($a[i] < a[j]$) then
            {
                $max := a[j]$; $min := a[i]$;
            }
            else
            {
                $max := a[i]$; $min := a[j]$;
            }
        }
        else
        {   // If $P$ is not small, divide $P$ into subproblems.
            // Find where to split the set.
                $mid := \lfloor (i + j)/2 \rfloor$;
            // Solve the subproblems.
                MaxMin($i, mid, max, min$);
                MaxMin($mid + 1, j, max1, min1$);
            // Combine the solutions.
                if ($max < max1$) then $max := max1$;
                if ($min > min1$) then $min := min1$;
        }
}
```

Performance of DandC - MAXMIN

(i) No. of Element Comp's : Let T(n) repr. No. of Element for 'n' elements

$$\left.\begin{aligned} T(n) &= 0, \quad n=1 \\ &= 1, \quad n=2 \end{aligned}\right\} \text{Small}$$

$$\boxed{= 2T(n/2)+2}, \quad n>2$$



$$T(n) = 2.T(n/2) + 2 \quad - ①$$

$$T(n/2) = \underline{2.T(n/4) + 2} \quad - ②$$

$$T(n) = 2\left[2T(n/4) + 2\right] + 2$$

$$= 4.T(n/4) + 4 + 2 \quad - ③$$

$$= 2^2.T(n/2^2) + \sum_{i=1}^{2} 2^i$$

$$\vdots$$

$$= 2^k.T(\underline{\underline{n/2^k}}) + \sum_{i=1}^{k} 2^i$$

$$= \frac{n}{2}.T(2) + 2^{k+1} - 2$$

$$= \frac{n}{2} + 2\cdot 2^k - 2 = \frac{n}{2} + n - 2$$

$$O(n)$$

$$\boxed{T(n) = \frac{3n}{2} - 2}$$

No. of Elem. Comp's

(All-Cases)

$$S_n = \frac{a(r^n - 1)}{r - 1} = \frac{2(2^k - 1)}{2 - 1}$$

$$= 2^{k+1} - 2$$

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

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

	Non-DC	D and C
1) Inc. order	$(n-1)$ ✓	$\left(\frac{3n}{2}-2\right)$
2) Dec. order	$2(n-1)$	$\left(\frac{3n}{2}-2\right)$ ✓
3) Random order	$\left(\frac{3n}{2}-1\right)$	$\left(\frac{3n}{2}-2\right)$ ✓

Space Complexity

Non-DC: $O(1)$

DC: $O(\log n)$

THANK - YOU