CS & IT ENGINEERING

Algorithms

Design Strategies

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;
```

$\log\log n$

$j = 1 \quad j = 2$

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

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

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

$\frac{n}{a}$

$\frac{n}{j}$

$$\Rightarrow T(n) = O\left(\log\log n \cdot (n \cdot \log n)\right)$$

```
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

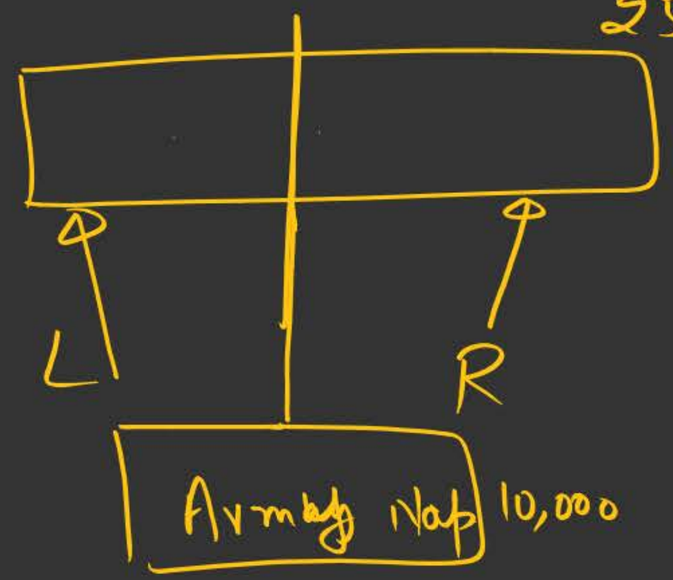
Divide & Conquer : (DandC)

⟹ 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;

⟹ Napolean



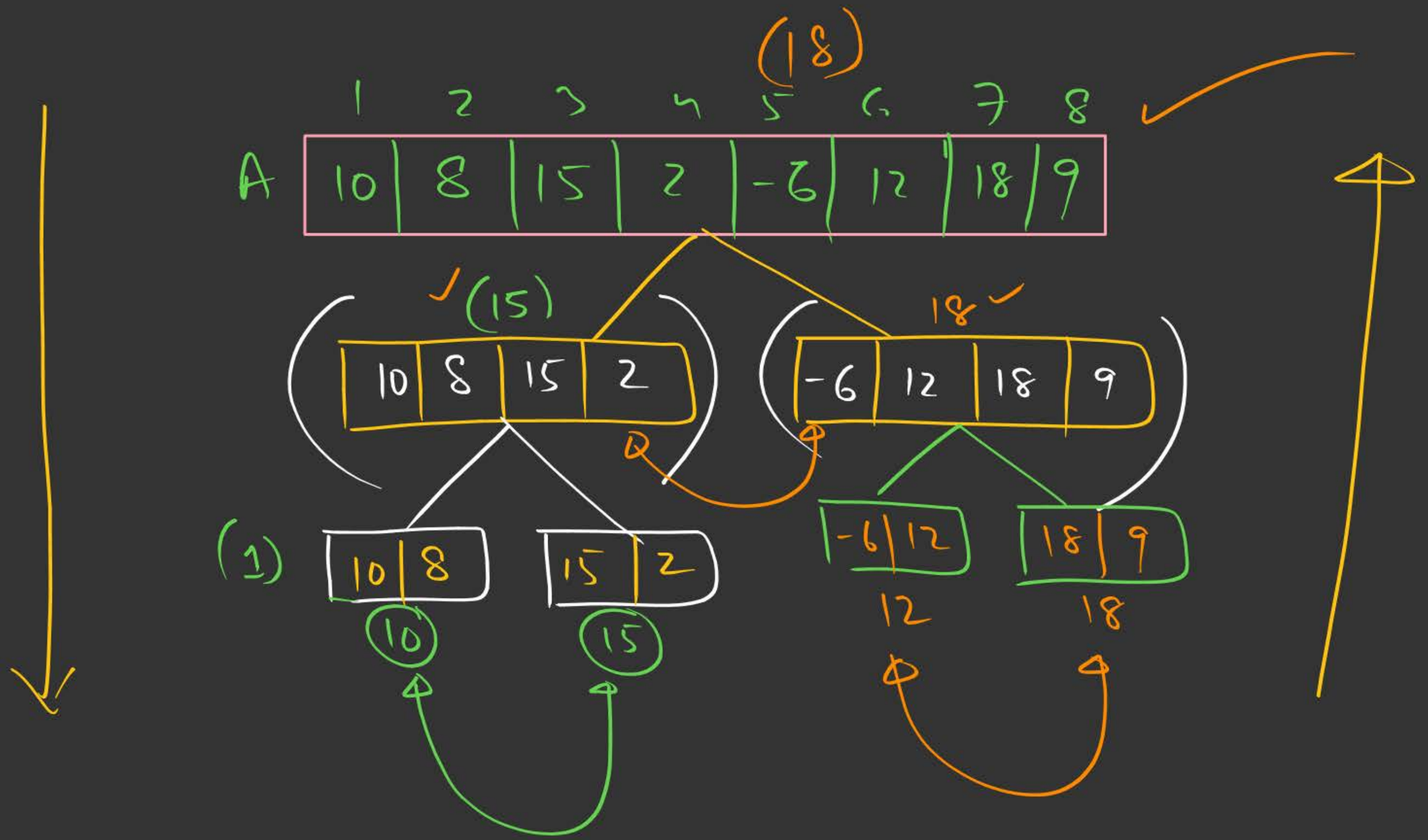
(ii) Britishers

(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)
10 8 15 2
18
-6 12 18 9
(1)
10 8
15 2
-6 12
18 9
10
15
12
18

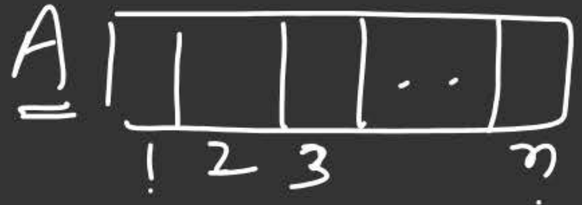
Topic : Divide and Conquer

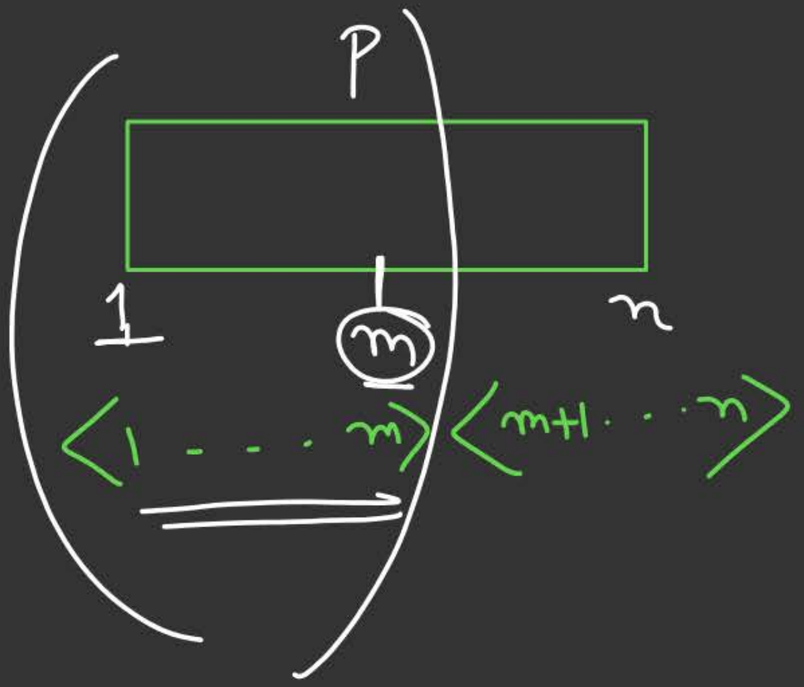
Control Abstraction

```
1.   Algorithm DAndC(P)
2.   {
3.       if Small(P) then return S(P);
4.       else
5.       {
6.           divide P into smaller instances P1, P2,... ,Pk, k ≥ 1;
7.           Apply DAndC to each of these subproblems;
8.           return Combine(DAndC(P1),DAndC(P2) ,.....,DAndC(Pk));
9.       }
10.  }
```

(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

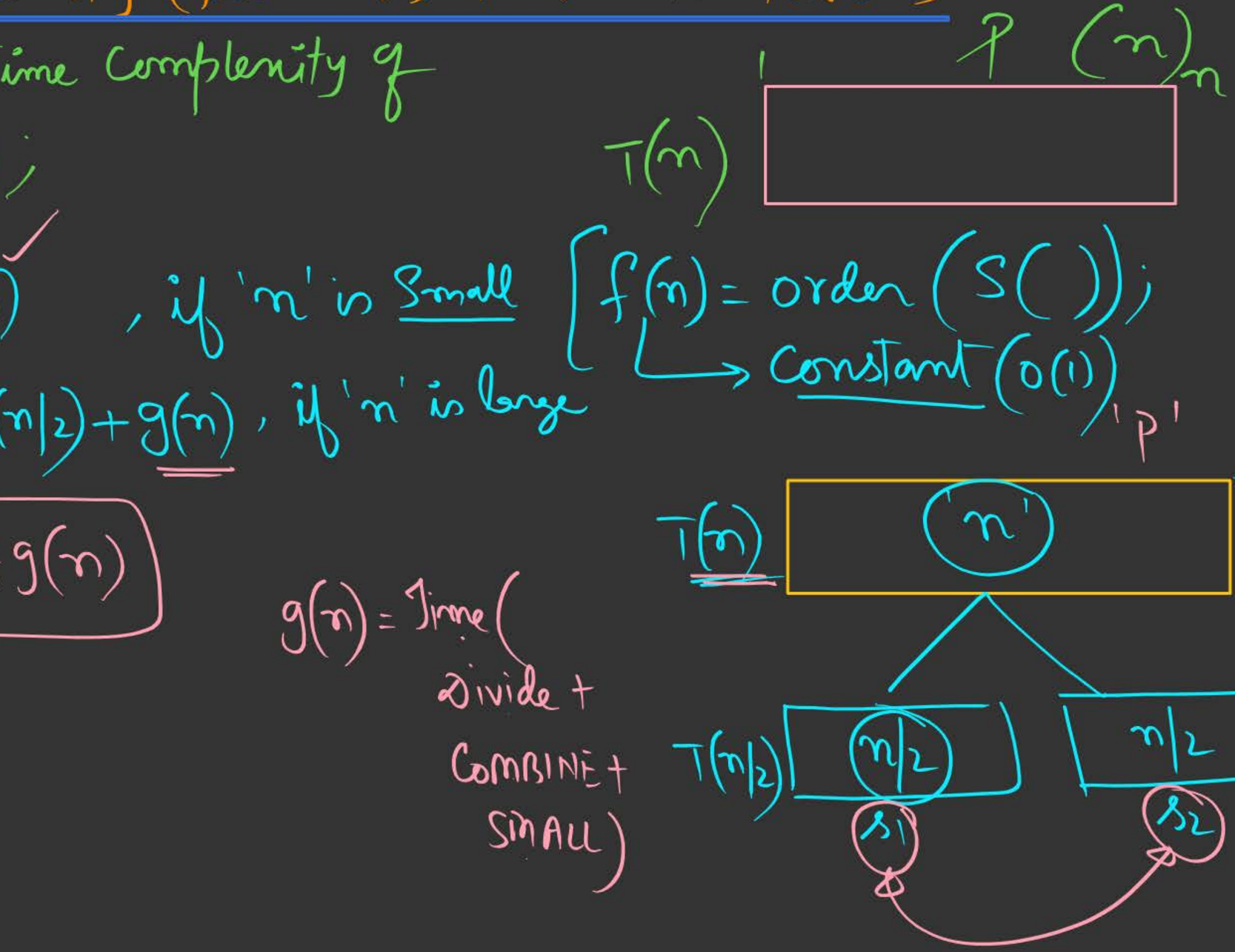
→ Let T(n) repr. Time Complexity of DandC(n);

$$T(n) = \begin{cases} f(n) & \text{, if 'n' is } \underline{\text{Small}} \\ 2 \cdot T(n/2) + g(n) & \text{, if 'n' is large} \end{cases}$$

[f(n) = order(S()); → Constant (O(1))]

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

g(n) = Time(Divide + Combine + SMALL)



$T(n) = 2 \cdot T(n/2) + g(n)$ → Divide & Conquer recurrence

- 2 → No. of Subproblems
- $n/2$ → Size of each Problem
- $g(n)$ → Time for Divide & Combine

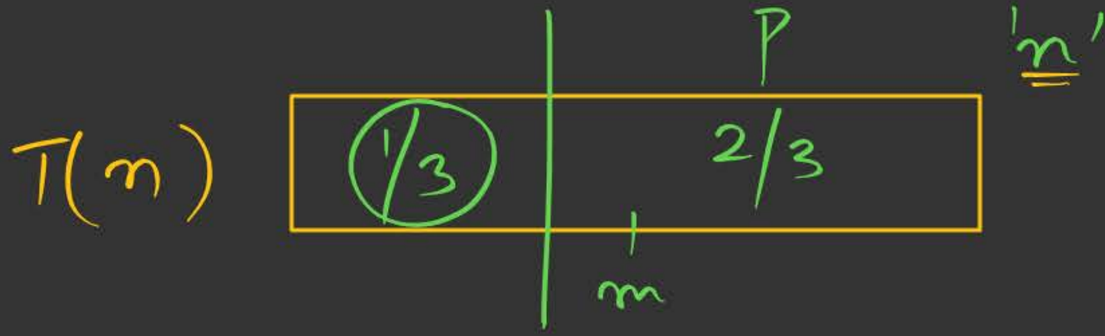
Generalized form:

$$T(n) = a \cdot T(n/b) + g(n) \quad \checkmark$$

- a → No. of Subproblems
- n/b → 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 I

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

I

General form:

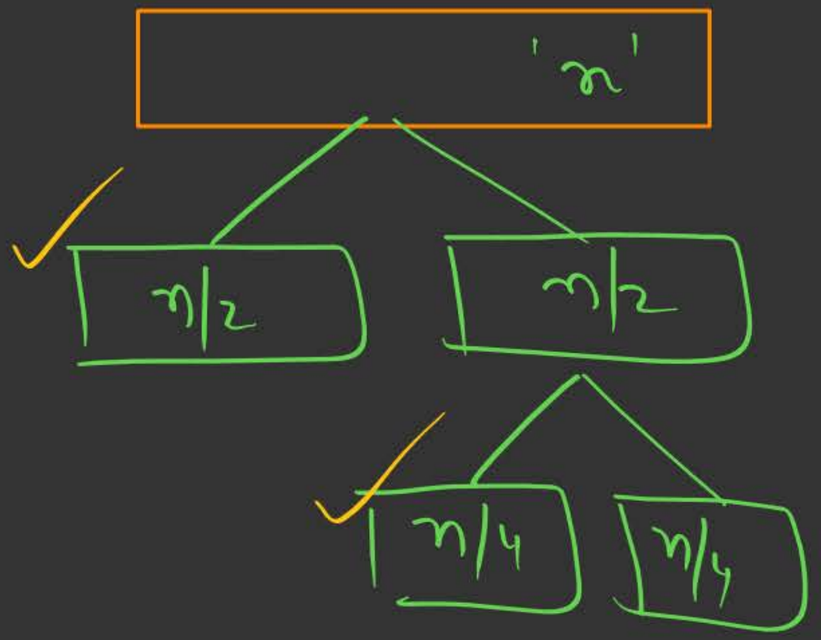
$$T(n) = T(\alpha n) + T((1-\alpha)n) + g(n)$$

$$0 < \alpha < 1$$

Asymmetric case II:

Ex:

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



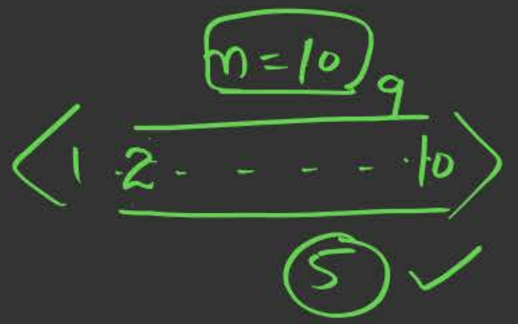
Note 1 : In DandC Strategy, Divide is Mandatory but Combine is optional (depends on applications)

Topic : Divide and Conquer

Computing time of DAndC is described by the recurrence relation

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

1) MaxMin: Procedure to Find Max & Min (Simultaneously) in an array of Size 'n';



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

Comparison ← Index / Element ✓

```
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 | nc

⇒ 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 Elems (n)

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

Space: O(1)

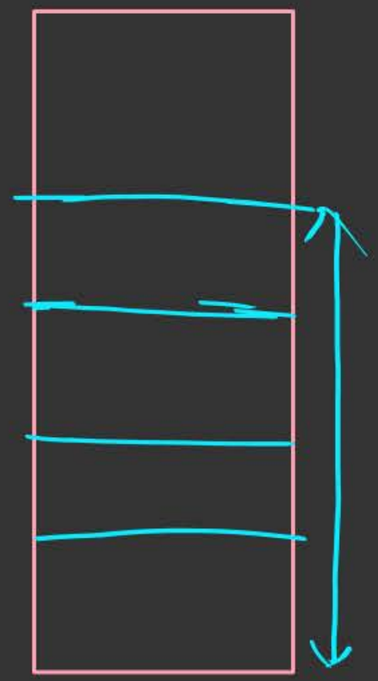
Inc order

	1st Comp	2nd Comp	Total
1) Best Case	$(n-1)$	0	$(n-1)$
2) Worst Case exc order	$(n-1)$	$(n-1)$	$2(n-1)$
3) Avg. Case	$(n-1)$	$n/2$	$\left(\frac{3n}{2}-1\right)$

Time : $O(n)$

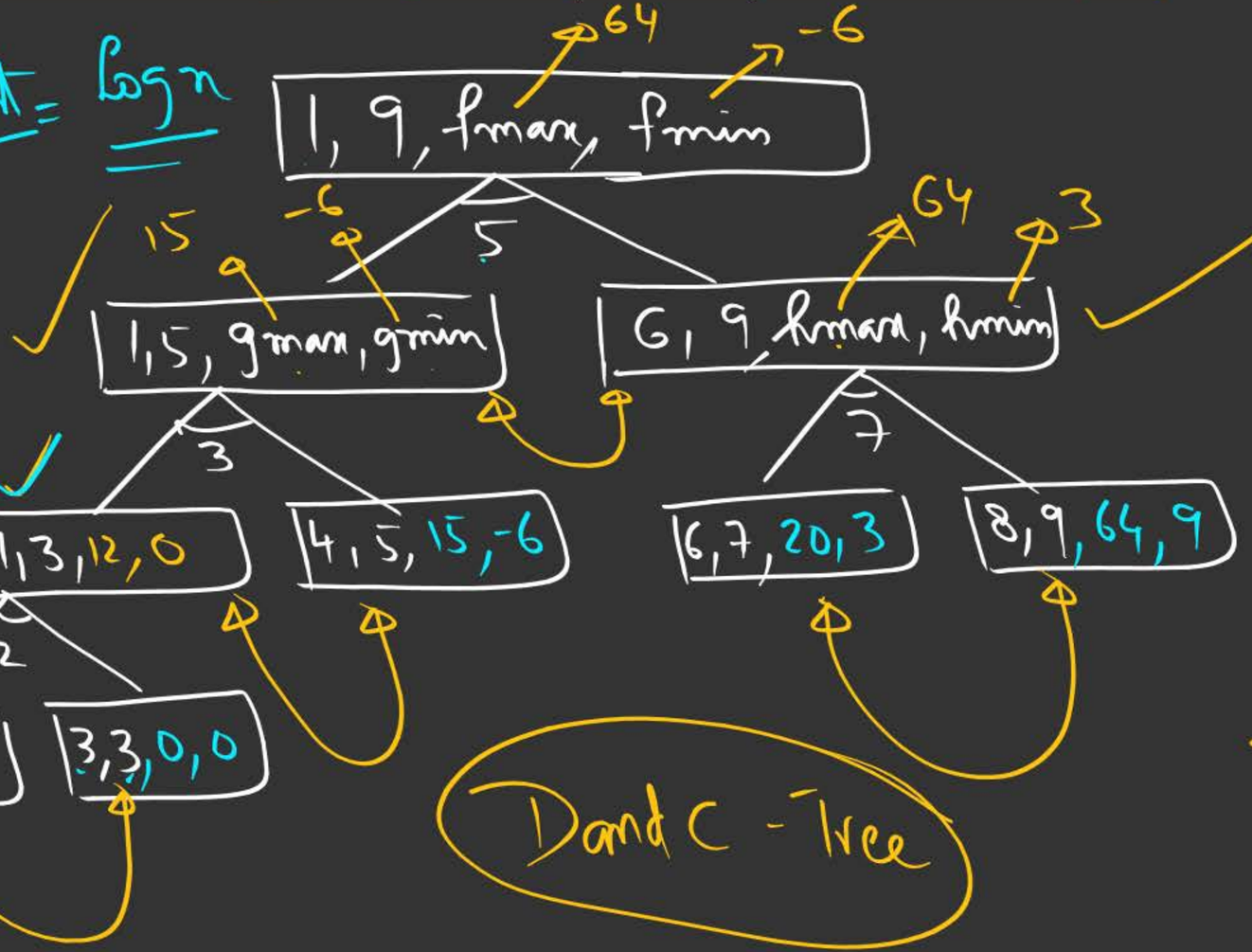
II. DandC - MAXMIN

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



STACK

Height h = $\log n$



DandC - Tree

Stack Space:

(Height of the Tree)

$h = (\log n)$

Topic : Divide and Conquer

DandC l, h

```
1.  Algorithm MaxMin (i, j, max, min)
2.  // a[1 : n] is a global array. Parameters i and j are integers.
3.  {                    Small : 1
4.       If (i = j) then max: min : = a[i]; // Small (P)
5.       else if (i = j –1) then // Another case of Small (P)
6.            {
7.                 if (a[i] < a [j]) then
8.                 {
                        max : = a[j]; min : = a[i];
                   }
```

Topic : Divide and Conquer

```
13.                else
14.                {
15.                        max: = a[i]; min : = a[j]
16.                }
17.          }
18.
```

Topic : Divide and Conquer

Problem is large

```
19.     else {

21.         mid: = ⌊(i + j) / 2⌋;
23.         MaxMin (i, mid, max, min);
24.         MaxMin (mid + 1, j,max1, min1);
26.         If (max< max1) then max: =max1;
27.         If (min> min1) then min: = min1;
28.     }
29. }
```

Combine

Topic : Divide and Conquer

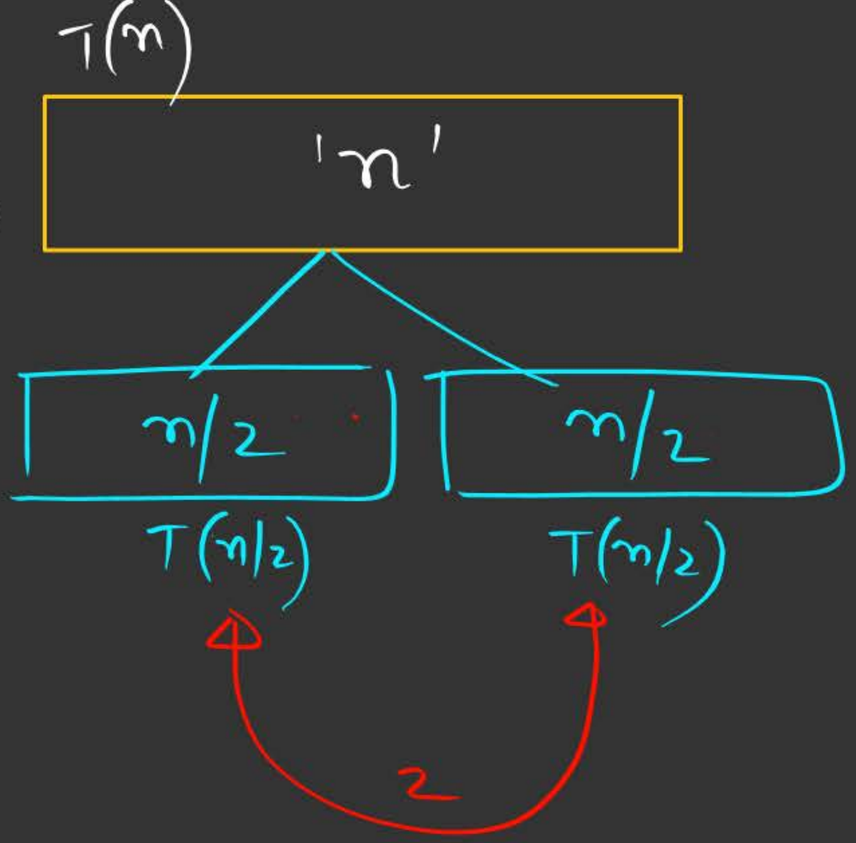
```
1   Algorithm MaxMin(i, j, max, min)
2   // a[1 : n] is a global array. Parameters i and j are integers,
3   // 1 ≤ i ≤ j ≤ n. The effect is to set max and min to the
4   // largest and smallest values in a[i : j], respectively.
5   {
6       if (i = j) then max := min := a[i]; // Small(P)
7       else if (i = j − 1) then  // Another case of Small(P)
8           {
9               if (a[i] < a[j]) then
10              {
11                  max := a[j]; min := a[i];
12              }
13              else
14              {
15                  max := a[i]; min := a[j];
16              }
17          }
18          else
19          {   // If P is not small, divide P into subproblems.
20              // Find where to split the set.
21                  mid := ⌊(i + j)/2⌋;
22              // Solve the subproblems.
23                  MaxMin(i, mid, max, min);
24                  MaxMin(mid + 1, j, max1, min1);
25              // Combine the solutions.
26                  if (max < max1) then max := max1;
27                  if (min > min1) then min := min1;
28          }
29  }
```

Performance of DandC - MAXMIN

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

$$T(n) = \begin{cases} 0, & n=1 \\ 1, & n=2 \end{cases} \quad \text{Small}$$

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



$$T(n) = 2 \cdot T(n/2) + 2 \quad -(1)$$

$$T(n/2) = 2 \cdot T(n/4) + 2 \quad -(2)$$

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

$$= 4 \cdot T(n/4) + 4 + 2 \quad -(3)$$

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

$$\vdots$$

$$= 2^k \cdot T\left(n/2^k\right) + \sum_{i=1}^{k} 2^i$$

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

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

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

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

$O(n)$

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

No. of Elem. Comp's (All-Cases)

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

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

	Non-DC	Dand 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