CS & IT ENGINEERING

Algorithms

Divide & Conquer

Lecture No.- 01

By- Aditya sir

Topics to be Covered

Topic

Topic

Intro to DnC

Min Max Algo

About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 12,000+ students & working professions in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

Topic : Design Strategies

1. Divide & Conquer (DnC)
2. Greedy
3. Dynamic programming → (DP)

DAA

Topic : Design Strategies

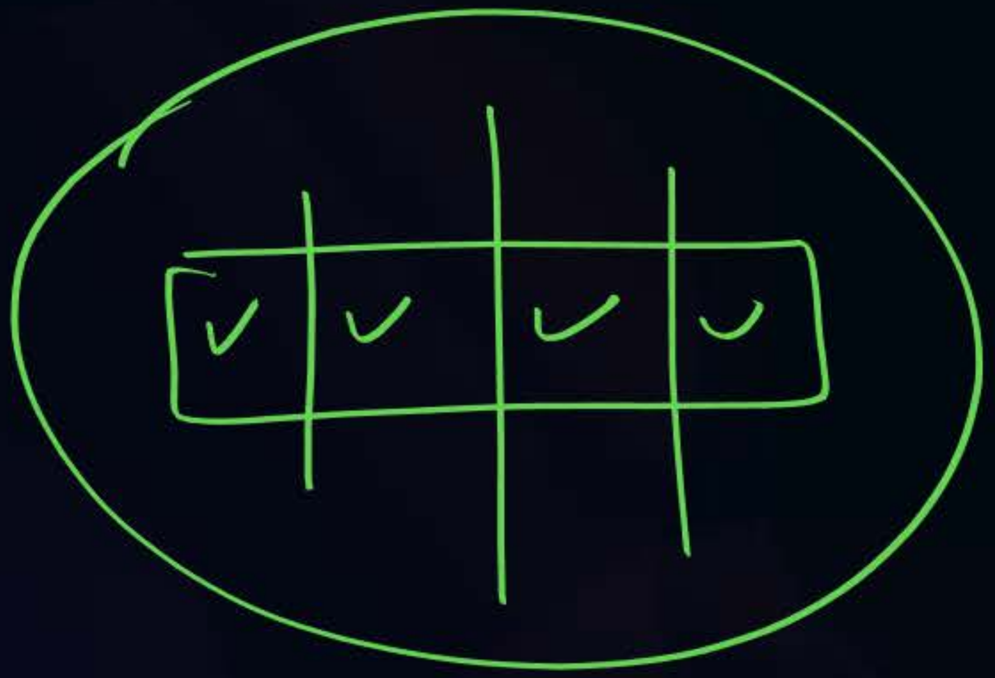
→ When the problem becomes large/complex, then divide the problem into subproblems, into further subproblems, until the subproblem becomes small.

→ Solve the smaller subproblems 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.

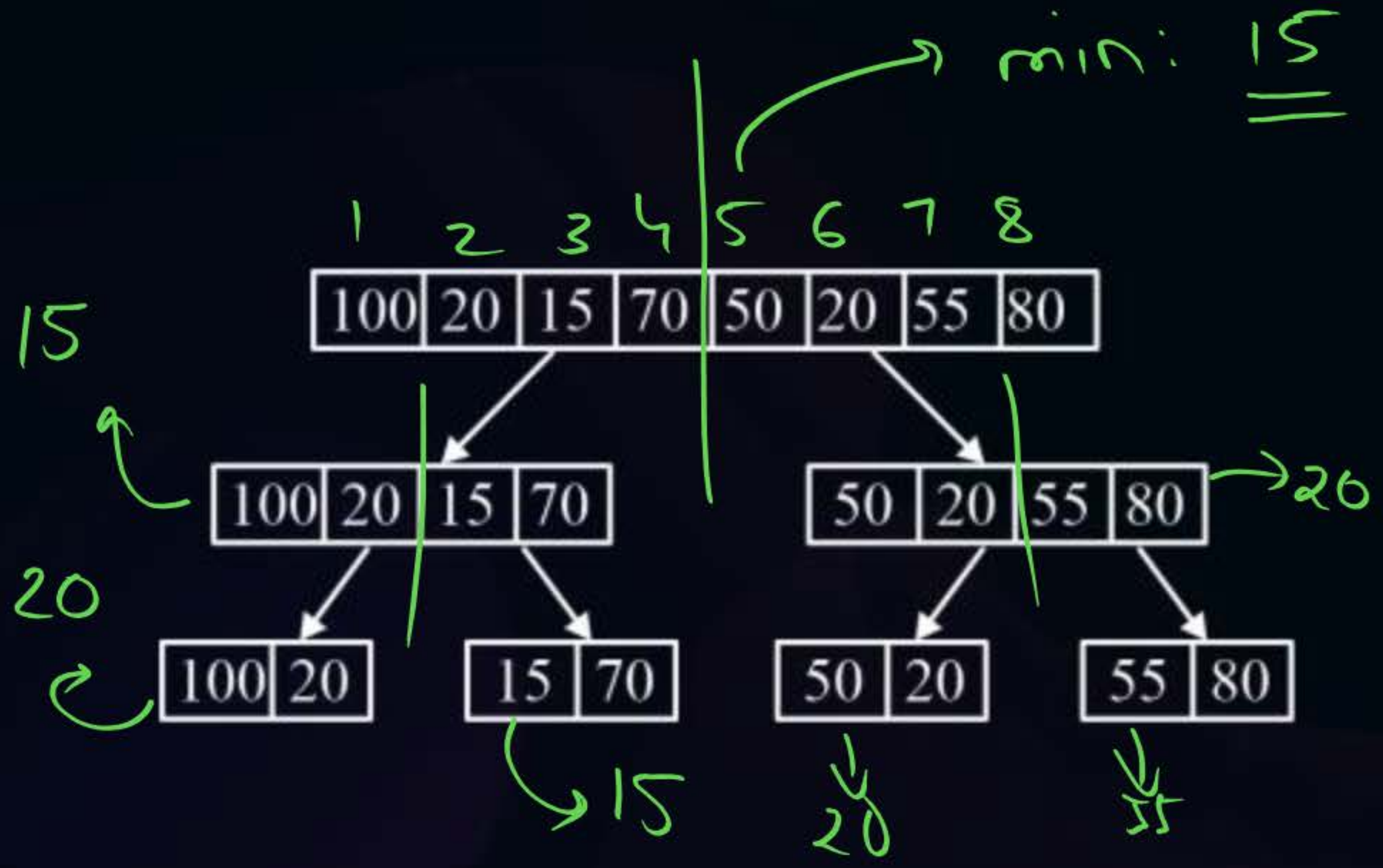
Britisher Rule

Eg.2. Divide & Conquer Rule

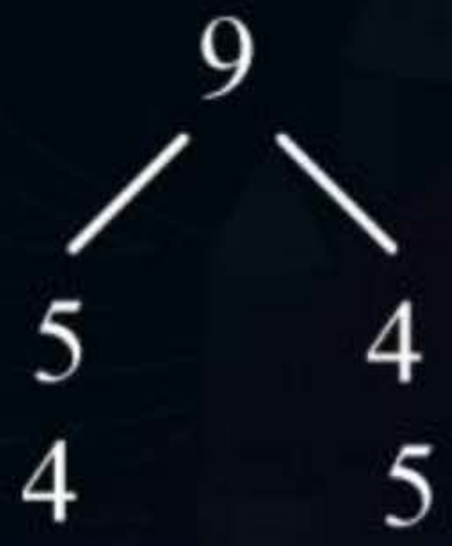


Topic : Design Strategies

#Q. Given an array of elements, find the minimum element in it.



$\left\lfloor \frac{l+r}{2} \right\rfloor$



$\left\lfloor \frac{0+5}{2} \right\rfloor = 2$

Topic : Design Strategies

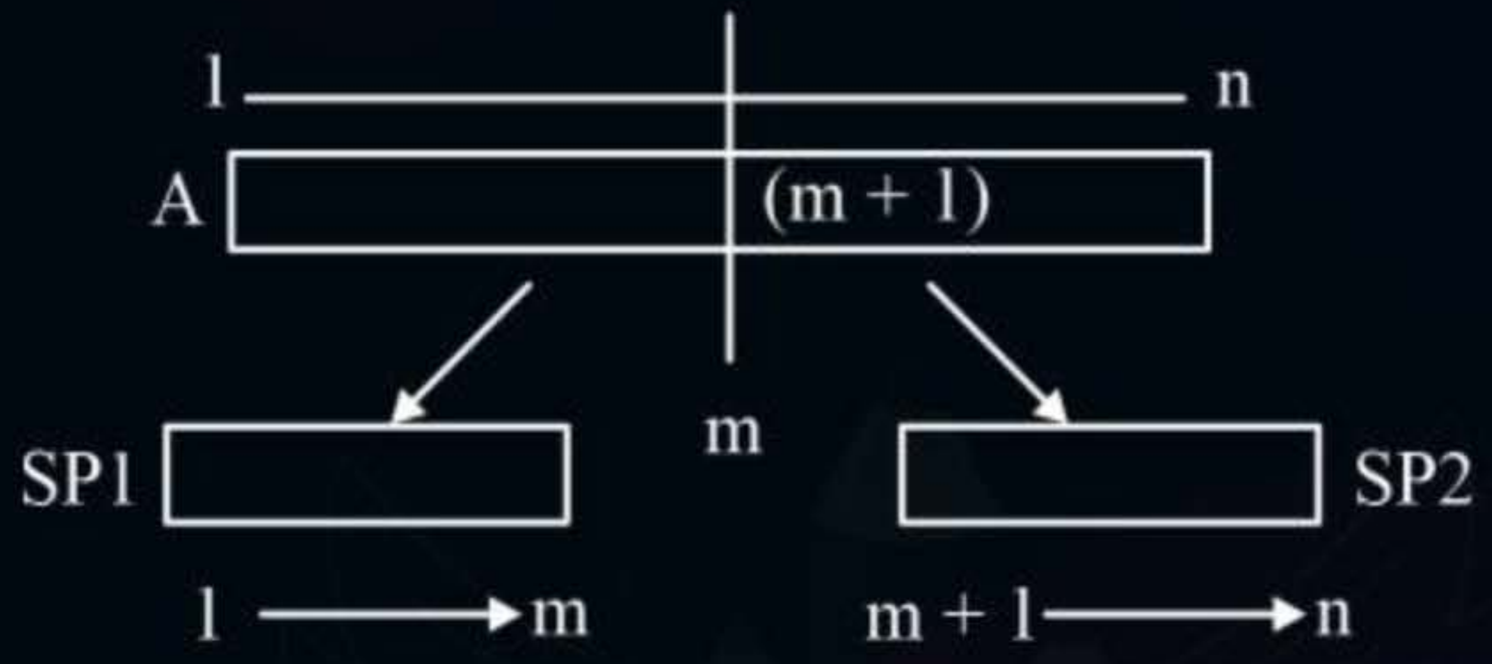
```
Algorithm DAndC(P)
{
        if Small(P) then return S(P);
        else
        {
                divide P into smaller instances P1, P2, ..., Pk, k ≥1 1;

                Apply DAndC to each of these subproblems;

                return Combine(DAndC(P1), DAndC(P2), ... , DAndC(Pk));
        }
}
```

Topic : Design Strategies

```
Algorithm DAndC (A, 1, n)
{
        if (SMALL (1, n))
        return (S(A, 1, n));
        else
        {
        m ← DIVIDE (1, n)
        s1 ← DAndC (A, 1, m)
        S2 ← DAndC (A, m + 1, n)
        Combine (S1, S2);
        }
}
```



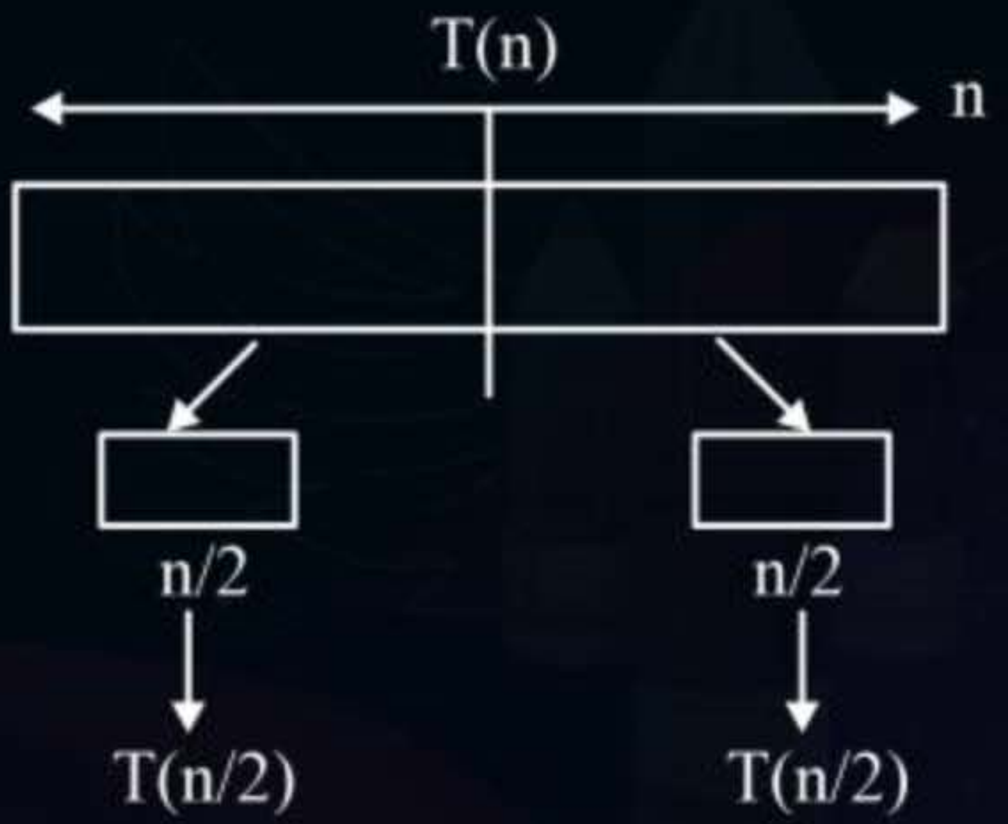
Topic : Design Strategies

Time complexity for divide & conquer problem:-

Let $T(n) \rightarrow$ original problem P.

$$T(n) = g(n), \text{ if } n \text{ is small}$$

$$= 2*T(n/2) + f(n), \text{ if } n \text{ is large}$$



Topic : Design Strategies

Computing time of DAndC is described by the recurrence relation

$$T(n)\begin{cases} g(n) & \text{n samll otherwise} \\ T(n_1)+T(n_2)+\ldots+T(n_k)+f(n), & \text{otherwise} \end{cases}$$

Topic : Design Strategies

Recurrence Relations in Divide & Conquer Problem

(1) Symmetric

(2) Asymmetric → Type-1, Type - 2

Topic : Design Strategies

(1) Symmetric D & C

General form: $T(n) = a * T(n/b) + f(n)$

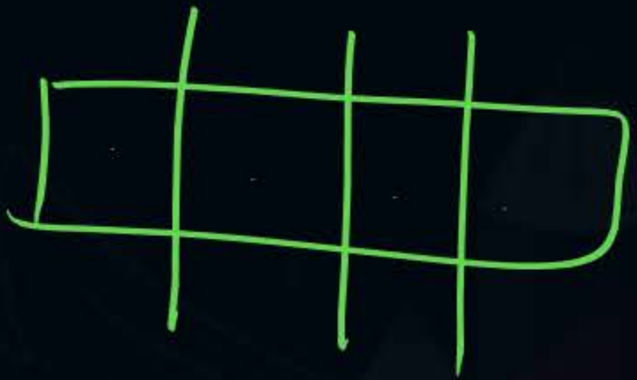
Eg.2. $T(n) = 2T(n/2) + n$

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

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

no. of subprob

size



Masters mtd

Topic : Design Strategies

Asymmetric D & C Type: -1

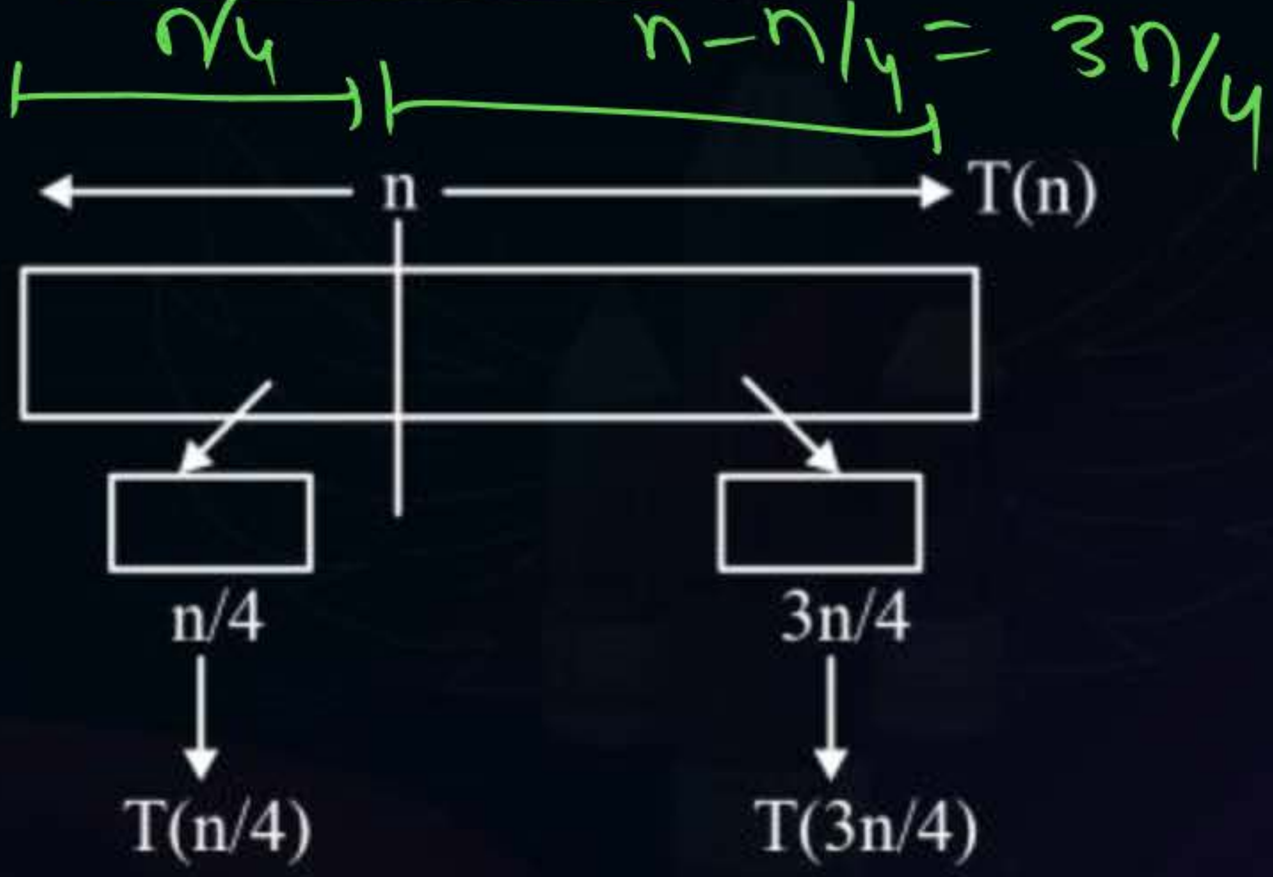
Eg. $T(n) = T(n/4) + T(3n/4) + f(n)$

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

General form

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

Recursion Tree Approach:



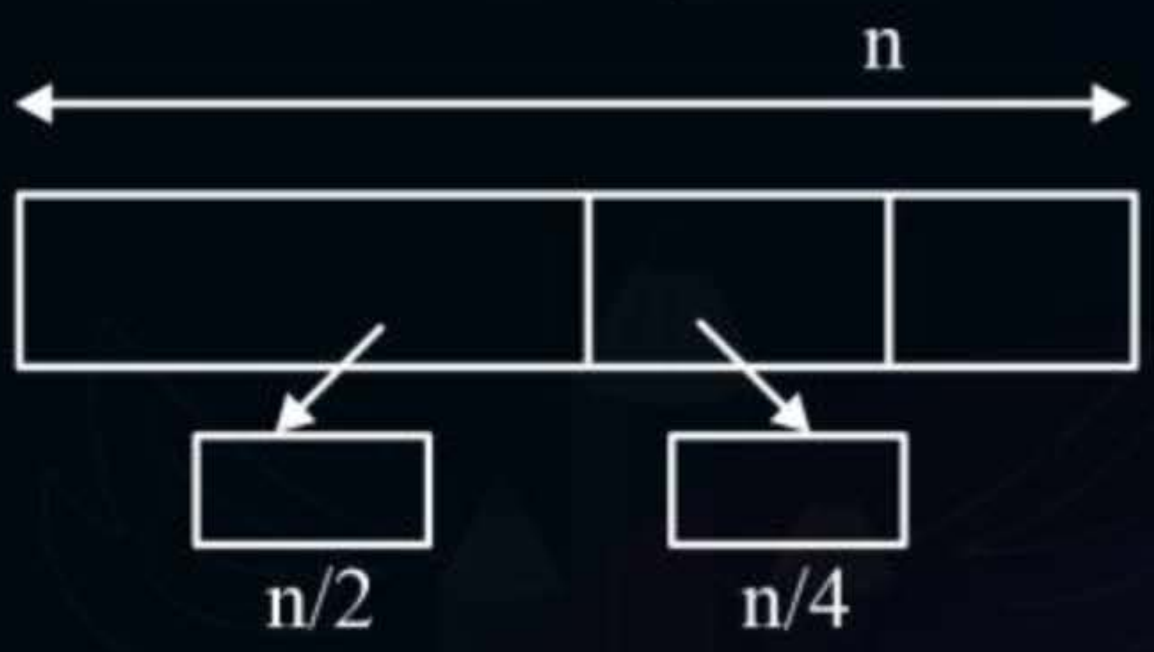
Topic : Design Strategies

(2) Asymmetric D & C Type: -2

Eg. $T(n) = T(n/2) + T(n/4) + f(n)$

General
$T(n) = T(\alpha n) + T(\beta n) + T(\delta n) \ldots + f(n)$

Recursion Tree Approach:



Topic : Design Strategies

Problem → Divide & Conquer:-

Divide → Mandatory

Conquer → optional

Eg. Binary Search

→ Divide

→ But no need to combine

Topic : Design Strategies

(1) Min – Max Problem

min & max elem

Topic : Design Strategies

(1) Min – Max Problem:

Find min & max element of the array simultaneously

Algo1 → $\boxed{\longrightarrow \text{min}}$ ⟶ $(n-1)$

Algo2 → $\boxed{\longrightarrow \text{max}}$ ⟶ $(n-1)$

$2(n-1)$

$= \underline{\underline{2n-2}}$

Topic : Design Strategies

Appr I :- Non – D & C Based

Eg.

7	15	–1	–6	17	1	8	59	29	24
1	2	3	4	5	6	7	8	9	10

Algo AJ – Min Max (A, n)

```
{
min = max = A[1];
for(i = 2; i ≤ n; i++)
if(A[i] < min)
{                    -> n-1
        min = A[i];
}
if(A[i] > max)
{                    -> n-1
        max = A[i];
}
}
```

min Comp $= 2(n-1)$

Max Comp $= 2(n-1)$

Topic : Design Strategies

Algo2 AJ – min-max (A, n) → non DnC

```
{
    min = max = A[1];
    for(i = 2; i ≤ n; i++)
    {
        if(A[i] < min)
        min = A[i];
        else if (A[i] > max)
        max = A[i];
    }
}
```

when?

min Comp → $(n-1)$

max Comp → $2(n-1)$

Topic : Design Strategies

Time complexity analysis of Aglo2

(1) Best case: input in Decr order

Eg.

10	7	5	2

if → n-1

else if → 0

(2) Worst case: input in Incr order

Eg.

2	5	7	10

if → n-1

else if → n-1

Summary:

Case	if	else if	Total Comparison
Best case	$(n-1)$	0	$(n-1)$ ✓
Worst case	$(n-1)$	$(n-1)$	$2\cdot(n-1)$ ✓
Avg case	$(n-1)$	$\frac{(n-1)}{2}$	$\frac{3}{2}(n-1)$

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

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

Topic : Min-Max Problem

Algo 2:- Without DAndC

Total no of comparisons

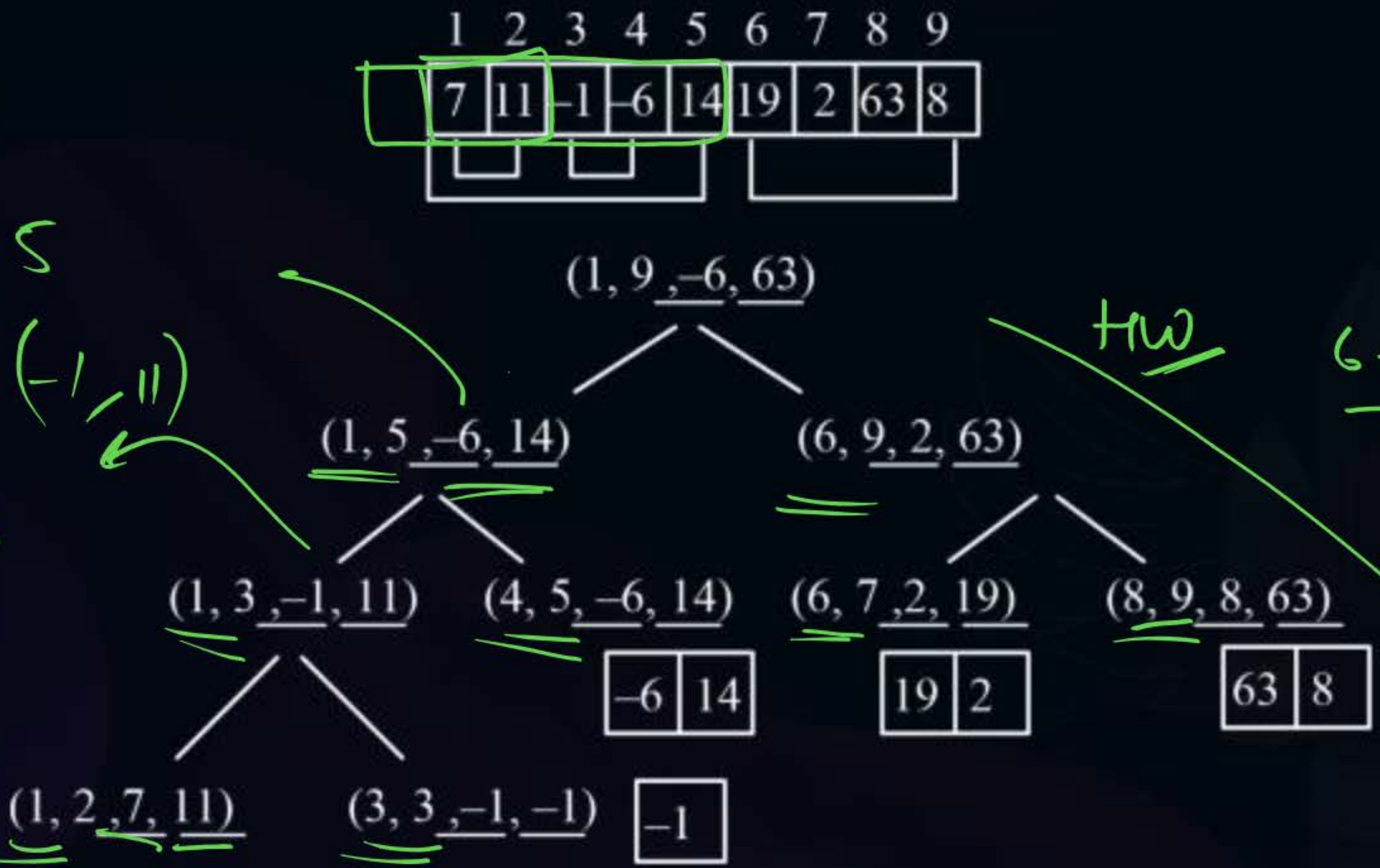
Best Case → (n-1)

Worst Case → 2*(n-1)

Average Case → $\frac{3}{2}$(n-1)

Divide & Conquer based

$$\left\lfloor \frac{1+9}{2} \right\rfloor = 5$$



$$\frac{1+5}{2} = 3$$

$$\frac{1+3}{2} = 2$$

$$\frac{6+9}{2} = \frac{15}{2} = 7$$

THANK - YOU