CS & IT ENGINEERING

Algorithms

Divide & Conquer

Lecture No.- 01

By- Aditya sir

Recap of Previous Lecture

Topic

Topic

Sorting

- insertion
- Radix

PYQs

Topics to be Covered

- Topic
- Topic
- Topic

Divide & Conquer

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.

@ADITYASIR_PW

Telegram Link for Aditya Jain sir: https://t.me/AdityaSir_PW

Topic : Time Complexity

#Q. for (i = 2; i ≤ n; i = i²)

V. Imp

→ O(log logn)

```
for (j = 1 ; j ≤ n; j ++)
    for (k = 1; k ≤ n; k = k + j)
        x = y + z;
```

→ O (nlogn)

nested dependent

A. O(log(logn))

B. O(nlogn)

C. O(log(nlog))

D. O(log(logn)*nlogn)

Overall TC = O(nlogn * log(logn))

$$i = 2 \rightarrow 2^2 \rightarrow (2^2)^2 \rightarrow (2^{2\times 2})^2 \quad \cdots\cdots \quad \underline{\underline{k^{th}\ \text{iter}}}$$

$$(2^2)^2 = 2^{2\times 2}$$

$$2^{2\times 2\times 2}$$

$$2^{2^1} \qquad 2^{2^2} \qquad 2^{2^3} \qquad (2^{2^k})$$

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

$$2^k = \log n$$

$$\boxed{k = \log(\log n)}$$

k & j Combined

$$\left.\begin{array}{lll}
j=1 & k: 1 \longrightarrow n\,;\ +1 & \Longrightarrow n \\
j=2 & k: 1 \longrightarrow n\,,\ +2 & \Longrightarrow n/2 \\
j=3 & k: 1 \longrightarrow n\,;\ +3 & \Longrightarrow n/3 \\
\vdots & & \\
j=n & k: 1 \longrightarrow n\,;\ +n & \Longrightarrow 1
\end{array}\right\}$$

j & k combined

$$\Rightarrow \frac{n}{1} + \frac{n}{2} + \frac{n}{3} \cdots 1$$

$$= n\underbrace{\left[1 + \frac{1}{2} + \frac{1}{3} \cdots \frac{1}{n}\right]}_{\log n} \Rightarrow \boxed{n\log n}$$

$$\sum_{i=1}^{n} \frac{1}{i} = \underline{\underline{\log n}}$$

Topic : Time Complexity

V. V. tricky

#Q. for i ← 1 to (n – 1)

for j ← i + 1 to n

for k ← 1 to j

print ("*")

nested dependent

What is the number of times "*" gets printed ?

$i = 1 \rightarrow (n-1)$

$j = i+1 \rightarrow n$

$K = 1 \rightarrow j$

$i = 1 \qquad j: 2 \rightarrow n$

$$\left.\begin{array}{l} j = 2 \longrightarrow 2 \\ j = 3 \longrightarrow 3 \\ j = n \longrightarrow n \end{array}\right\} \frac{n(n-1)}{2} \Rightarrow \sum_{2}^{n} j$$

$i = 2 \quad j : 3 \rightarrow n \implies \sum_{3}^{n} j$

$i = 3 \quad j : 4 \rightarrow n \implies \sum_{4}^{n} j$

$i = n-1 \quad j : n \rightarrow n \implies \sum_{n}^{n} j$

$$\text{Total} = \left[\sum_{2}^{n} j + \sum_{3}^{n} j + \sum_{4}^{n} j \cdots \sum_{n}^{n} j \right]$$

Design Strategies

Algo → Design & Analysis

Topic : Design Strategies

1. Divide & Conquer (D & q)
2. greedy
3. Dynamic programing → (DP)

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 problem, combine their results if required to get the solution of original problem.

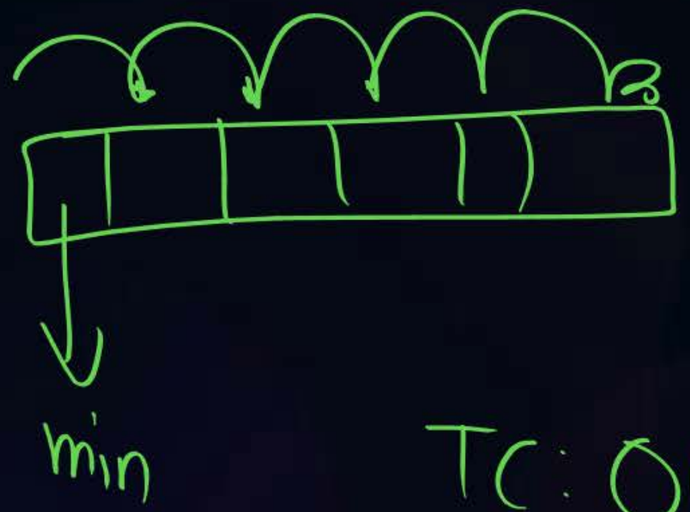
→ In general a problem is said to be small, if it can be solved in one/two basic operations.

Constant time

Topic : Design Strategies

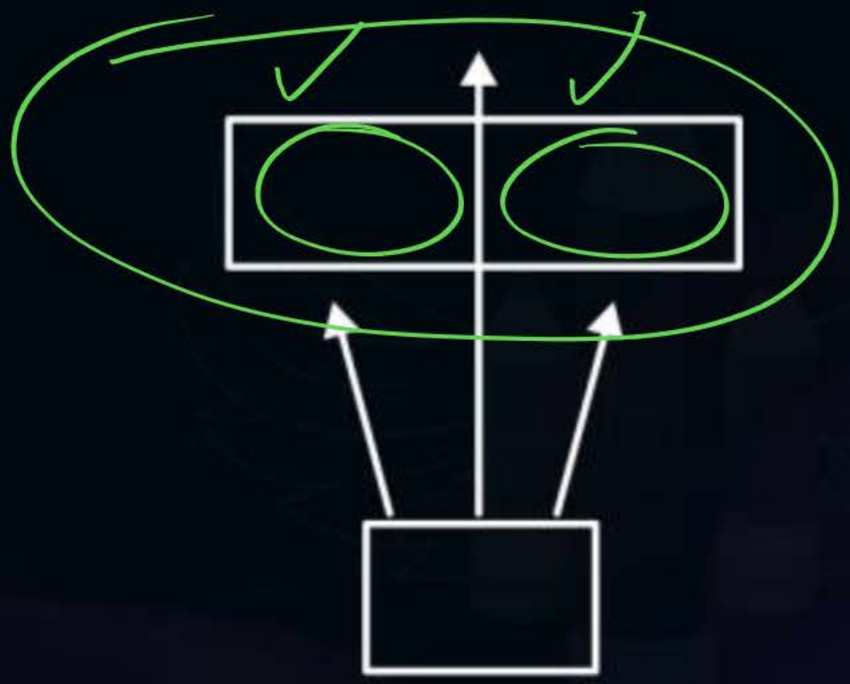
Britisher Rule

Eg.2. Divide & Conquer Rule



TC: O(n)

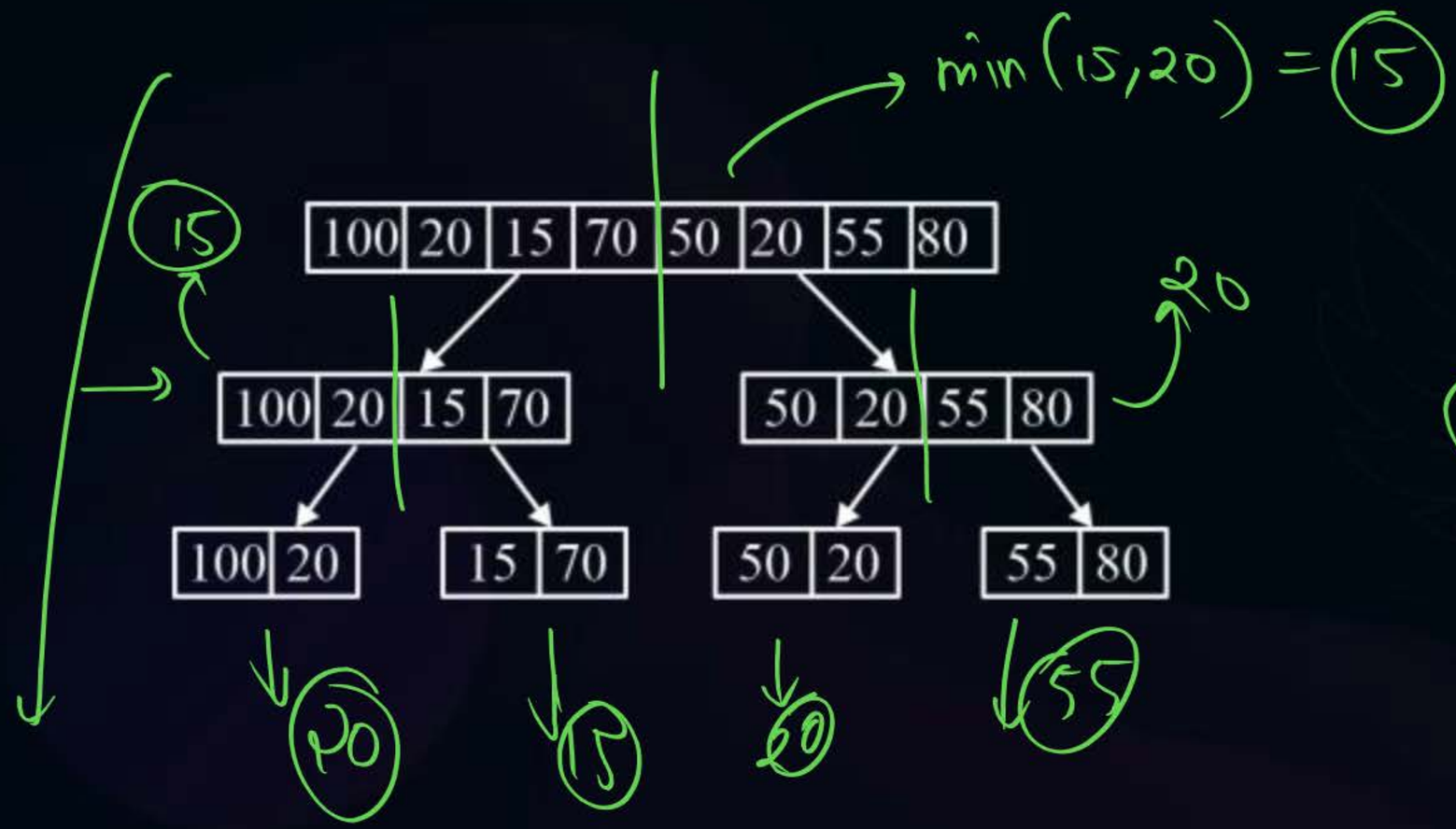
Background



Topic : Design Strategies

#Q. Given an array of elen, find the min element.

Idea



Topic : Design Strategies

Idea

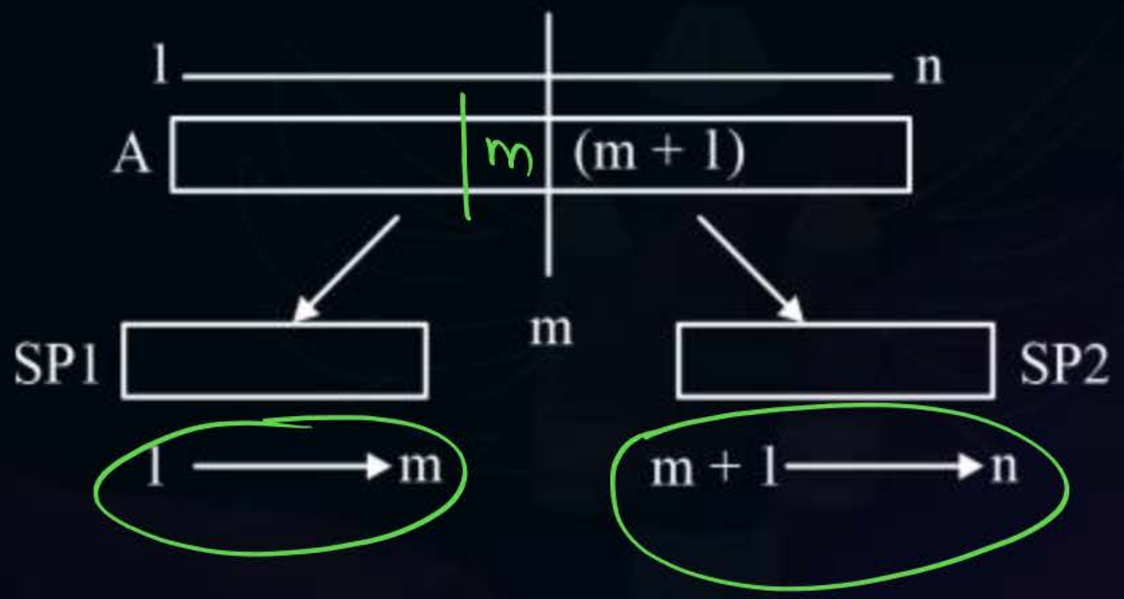
```
Algorithm DAndC(P)
{
    if Small(P) then return S(P);
    else
    {
        divide P into smaller instances P1, P2, ..., Pk, k ≥ 1;
        Apply DAndC to each of these subproblems;
        return Combine(DAndC(P1),DAndC(P2), ... , DAndC(Pk));
    }
}
```

Soln(P)

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);
}
```

overall soln



Topic : Design Strategies

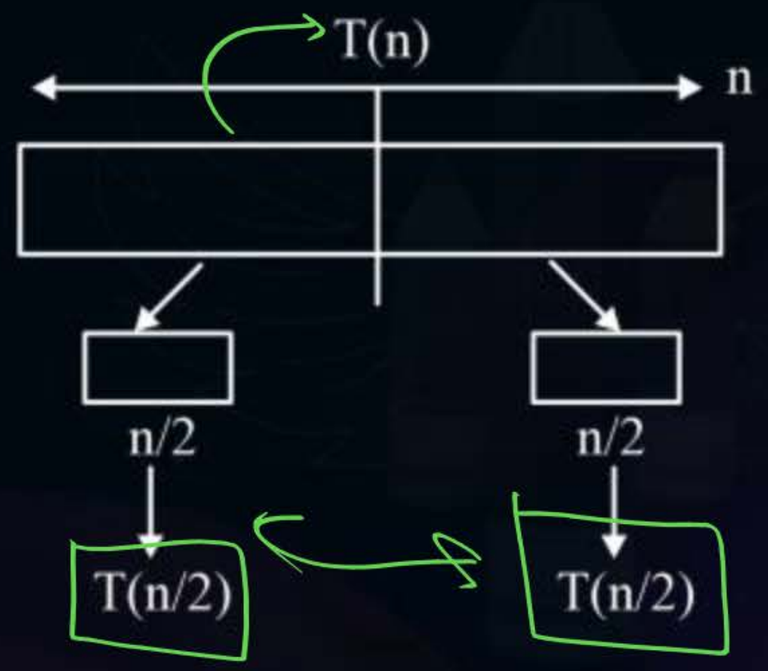
Time complexity for divide & conquer problem:-

Let T(n) → original problem P.

T(n) = g(n), if n is small

= 2*T(n/2) + F(n), if n is large

T(n/2) + T(n/2)



Topic : Design Strategies

Computing time of DAndC is described by the recurrence relation

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

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: $\boxed{T(n) = a * T(n/b) + F(n)}$

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

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

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

no. of subproblems

size of each subp

Mastery Mtd

Topic : Design Strategies

(2) Asymmetric D & C Type: -1

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

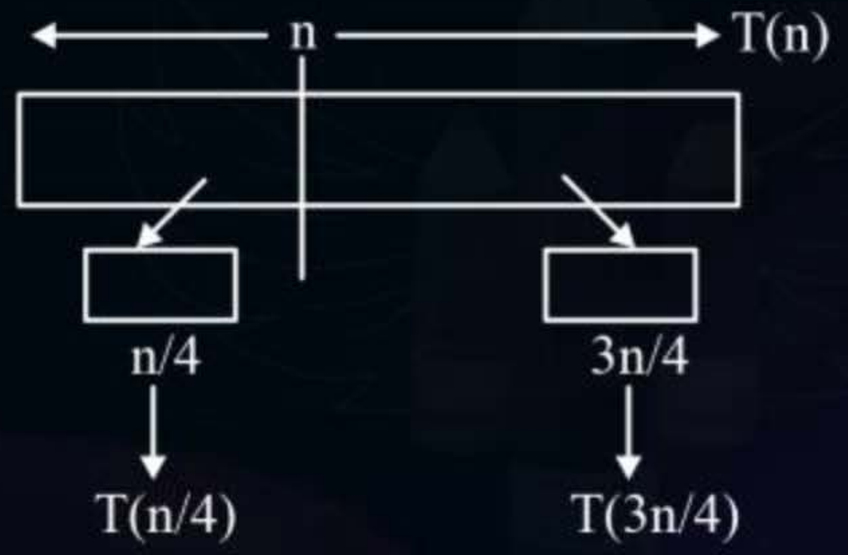
Eg. $T(n) = T(n/3) + T(2n/3) + g(n)$

General form

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

Recurrence Tree

Recursion Tree Approach:



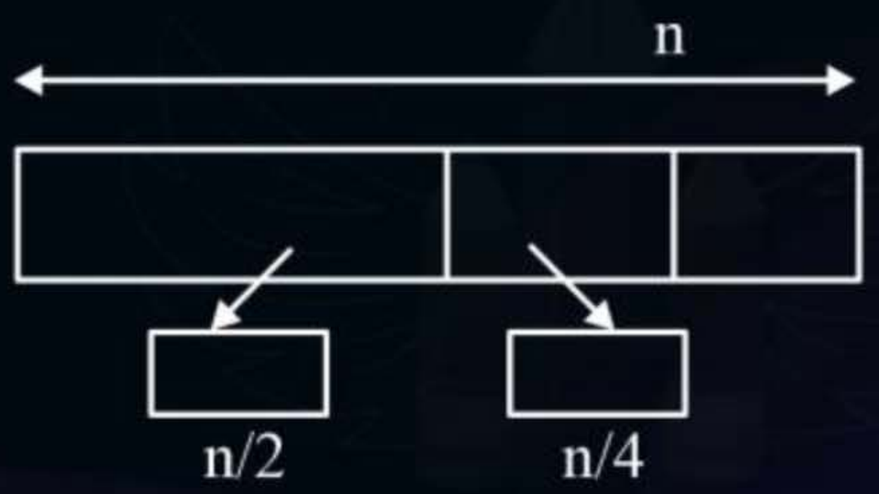
Topic : Design Strategies

(2) Asymmetric D & C Type: -2

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

General

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



Recursion Tree Approach:

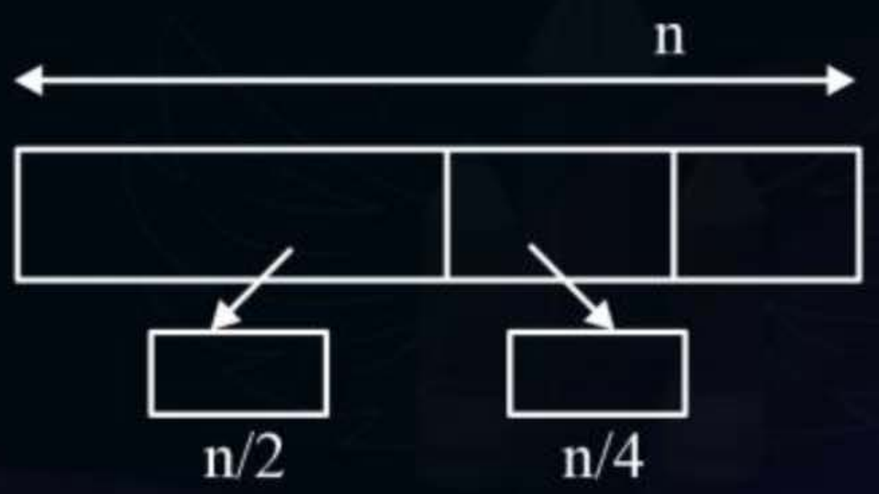
Topic : Design Strategies

(2) Asymmetric D & C Type: -2

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

General

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



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

Topic : Design Strategies

(1) Min – Max Problem:

Find min & max element of the array simultaneously

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)
{
    min = A[i]
}
if(A[i] > max)
{
    max = A[i]
}
```

$n-1$

$n-1$

Total Compr = $2*(n-1)$

Topic : Design Strategies

```
Algo    AI – min-max (A, n)
        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]
        }
```

Topic : Design Strategies

Time complexity analysis of Aglo2

(1) Best case: input in Decr order

Eg.

10	7	5	2

$\longrightarrow (n-1)$ Compr

(2) Worst case: input in Incr order

Eg.

2	5	7	10

$\Rightarrow 2*(n-1)$ Compr

$(n-1)$ if

$(n-1)$ else if

Topic : Design Strategies

1 possible Scenario

Non-DnC

Bonus: Avg Case:

On an avg, if condition fails 1/2 of the times.

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

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

V.imp Result

Logic:

If ( ) → True (n/2–1) → 0; False (n/2) → n/2

(n–1)

elet if ( )

Topic : Design Strategies

Summary: V.V.Imp

Exact

$n = 10$

	if	else if	Total Comparision
Best case	((n–1) + 0)	⇒	(n–1)
Wrost case	(n–1)	(n–1) ⇒	2*(n–1)
Avg case	(n–1)	n/2 ⇒	$\boxed{\frac{3n-1}{2}}$ → v.imp

Desc :

Asc :

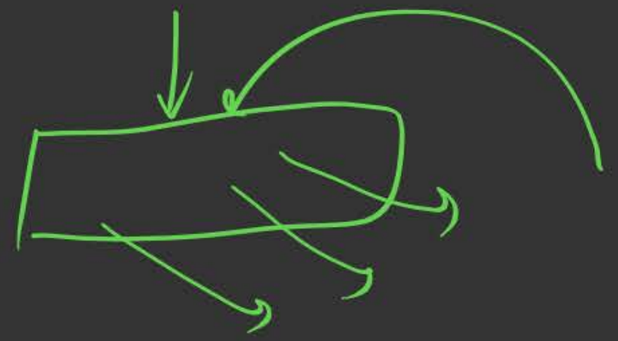
→ $\Theta(n)$

One of the Cases

$\frac{3 \times 10}{2} - 1 = 15 - 1 = 14$

Soln 2nd :- Insertion Sort

$\Rightarrow O(n^2)$



2 mins Summary

- Topic
- Topic
- Topic
- Topic
- Topic

Intro to DnC

Min Max

THANK - YOU