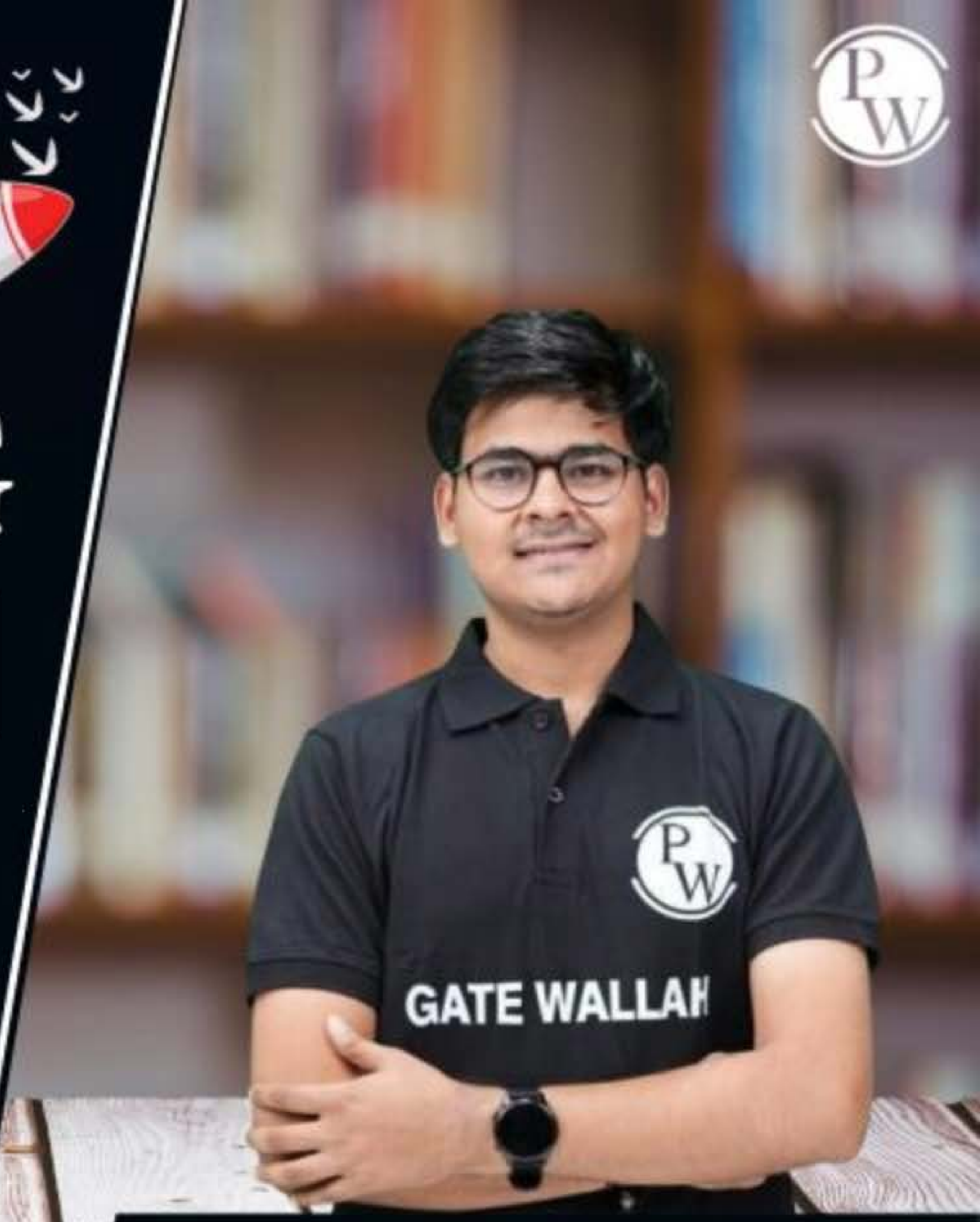
CS & IT ENGINEERING

Algorithms

Analysis of Algorithms

Lecture No.- 06



By- Aditya Jain sir

Topics to be Covered

Topic

Topic   Asymptotic analysis

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 : Analysis of Algorithms

#Q. Asymptotic Comparison of 2 functions: [MSQ]

$f(n) = n$

$g(n) = n \log n$

$n < n\log n$

A $f = O(g)$

$n = o(n\log n)$

C $f = \Omega(g)$

B $f = o(g)$

D $f = \omega(g)$

Ans A, B

Topic : Analysis of Algorithms

#Q. Asymptotic Comparison of 2 functions: [MSQ]

$f(n) = n^2 (\log n)$

$g(n) = n (\log n)^{10}$

$f > g$

A. $f = O(g)$ ✗

B. $f = \Omega(g)$ ✓

C. $f = \omega(g)$ ✓

D. $f = o(g)$ ✗

C, B

$$\cancel{n}^2 \log n > \cancel{n}(\log n)^{10}$$

$$n \log n \qquad (\log n)^{10}$$

$$n \cancel{\log n} \qquad \cancel{\log n} * (\log n)^9$$

$$n > (\log n)^9$$

$\underline{f > g}$

Topic : Analysis of Algorithms

#Q. Asymptotic Comparison of 2 functions: [MSQ]

$$f(n) = n^3, \; 0 < n \leq 10{,}000$$
$$= n, \quad n > 10{,}000$$

V. adv

$$g(n) = n, \; 0 < n \leq 100$$
$$= n^3, \; n > 100$$

Ans :- C, D

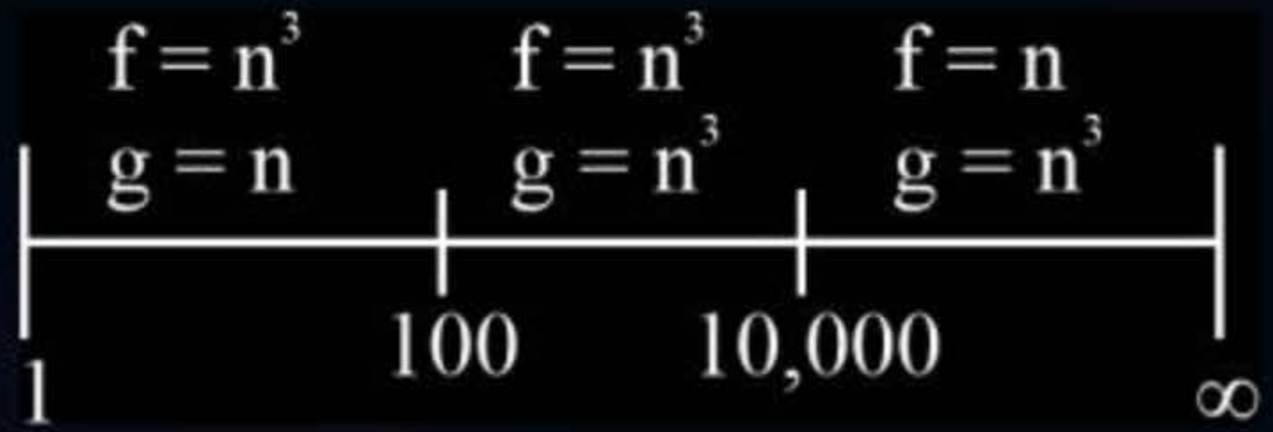
A. $f(n) = \Omega(g(n))$, for $n > 100$ ✗

B. $f(n) = o(g(n))$, for $n > 100$ ✗

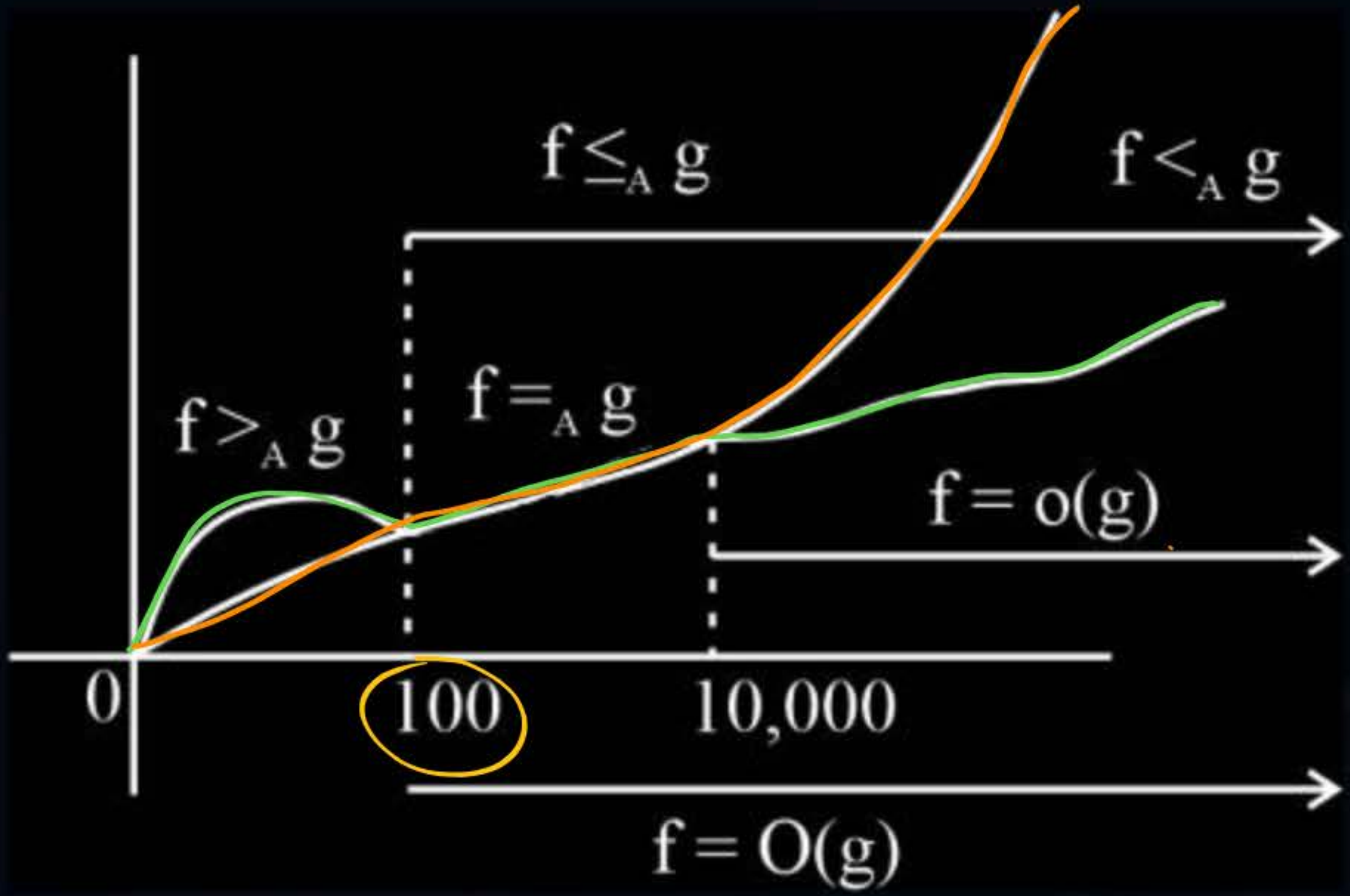
C. $f(n) = O(g(n))$, for $n > 100$ ✓

D. $f(n) = o(g(n))$, for $n > 10{,}000$ ✓

Topic : Analysis of Algorithms



Topic : Analysis of Algorithms



Topic : Analysis of Algorithms

#Q. Arrange the given functions in increasing order of rate of growth:

$f_1 \rightarrow 2^n, f_2 \rightarrow n^{3/2}, f_3 \rightarrow n \log n, f_4 \rightarrow n^{(\log n)}$ [MCQ]

A. f_2, f_3, f_1, f_4 ✗

B. f_3, f_2, f_1, f_4 ✗

C. f_3, f_2, f_4, f_1 ✓

D. f_2, f_3, f_4, f_1 ✗

2^n

$n^{3/2} = n^{1.5} = n\sqrt{n}$

$n \log n$

$n^{\log n}$

$f_3 < f_2$

$\cancel{n} \log n < \cancel{n}\sqrt{n}$

$f_1 > f_4$

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

$n \log_2 2 \qquad \log n * \log n$

$$n > (\log n)^2$$

Topic : Analysis of Algorithms

#Q. You are given a database having 10^x records. [NAT]

There are 2 packages available for processing the data.

Package A takes a time of 10 * n * log n while,

Package B takes a time of 0.0001 * n^2 for processing 'n' records.

Determine the smallest +ve integer x for which package A outperforms package B.

$t_A < t_B$

$t_A = 10n\log n$

$t_B = 0.0001n^2$

$= 10^{-4} * n^2$

$n = 10^x$

$t_A < t_B$

$10 \times 10^x \times \log_{10}(10^x) < 10^{-4} \times (10^x)^2$

$10^{x+1} \times x < 10^{(2x-4)}$

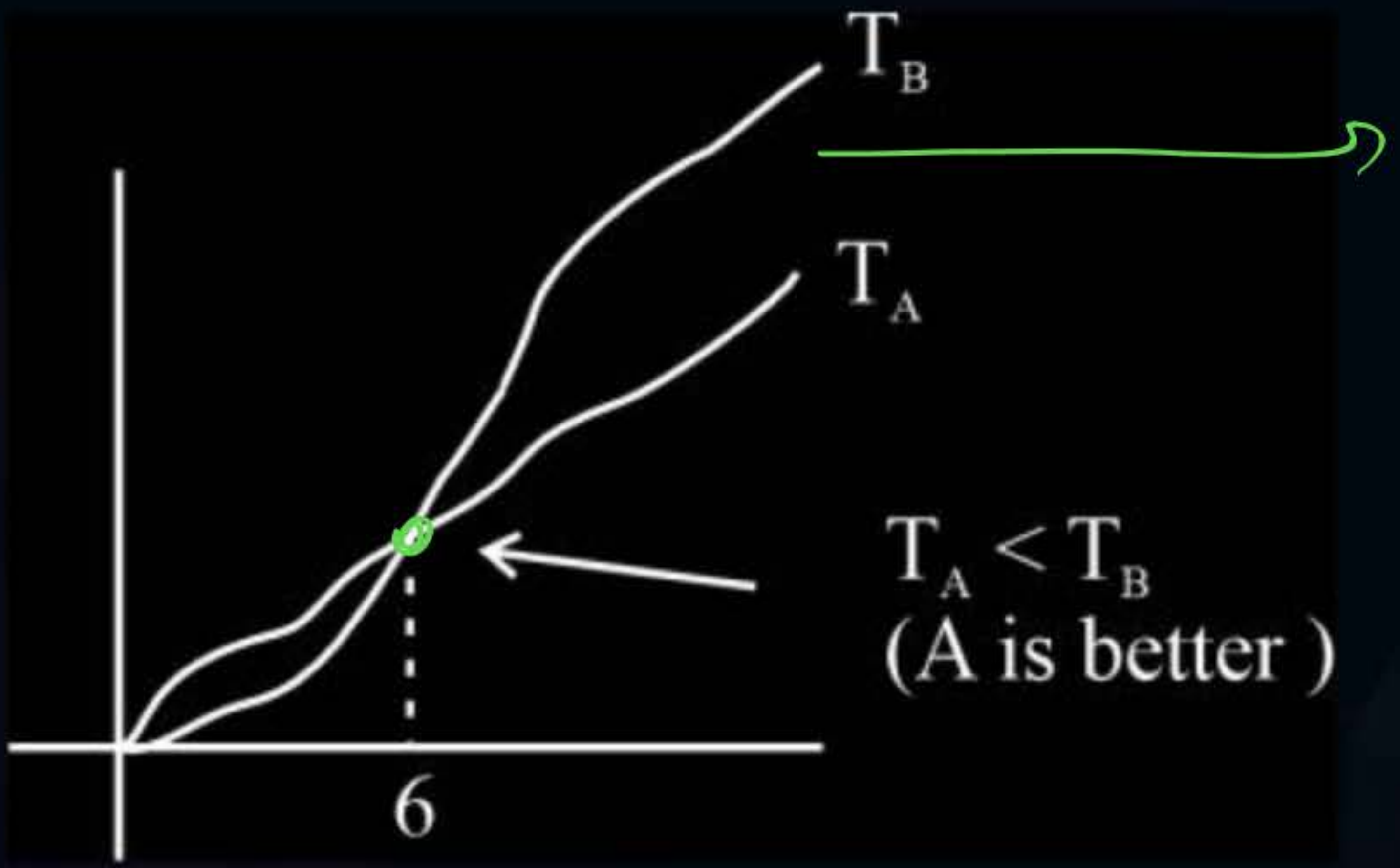
$x < 10^{(2x-4-x-1)}$

$\boxed{x < 10^{x-5}}$

$x=5 \quad 5 < 10^0 \quad \times$

$x=6 \quad 6 < 10^{6-5} \quad 6 < 10 \checkmark$

Topic : Analysis of Algorithms



#Q. An element in an Array is called Leader if it is greater than all elements to the right of it. The time complexity of the most efficient algorithm to print all Leaders of the given Array of size 'n' is ________.

[NAT]

Topic : Analysis of Algorithms

Solution 1: Brute Force

```
Algo AJLeader(A[], n) {
        for (i = 1; i < n; i++){
                for (j = i+1; j ≤ n; j++) {
                        if (A[i] < A[j]) {
                                break;
                        }
                }
                if (j == (n+1)) {
                        print(A[i])
                }
        }
}
```

Time Complexity Analysis of Brute Force Approach:

(1) Best Case:

A is in increasing order.

E.g.: A = 5, 10, 12, 18

Total Comparisons $\rightarrow (n - 1) * O(1) = O(n - 1)$

$\rightarrow O(n)$

Time Complexity Analysis of Brute Force Approach:

(2) Worst Case:

A is in decreasing order.

E.g.: A = 18, 15, 12, 10, 5

Total Comparisons $\rightarrow 4 + 3 + 2 + 1$

In general $\rightarrow (n - 1) + (n - 2) + (n - 3) \ldots 1 = \frac{n(n+1)}{2} - n$

$$= \frac{n^2 - n}{2} \rightarrow O(n^2)$$

#Q. An element in an Array is called Leader if it is greater than all elements to the right of it. The time complexity of the most efficient algorithm to print all Leaders of the given Array of size 'n' is______.

Topic : Analysis of Algorithms

Code walk through

A :	29	5	16	14	8	9
	✓	×	✓	✓	×	×

$a > \max($ $)$

Algo. AJ Leader (a, n)

```
{
    L= A[n];                 → O(1)
O(n)←     for (i = (n -1); i > =1; ~~i++~~ i--)
    {
        if(A [i] > L)
        {
            L= A[i];
            print (A[i]);
        }
    }   }
```

Topic : Analysis of Algorithms

Time complexity Analysis

1. Best Case:- Increasing order $\rightarrow O(n) \rightarrow \Omega(n)$
2. Worst Case:- Decreasing order $\rightarrow O(n) \rightarrow O(n)$

$\left.\right\}$ ~~$\theta(n)$~~ $\theta(n)$

Imp. Irrespective of the input order the Algo will take same amount of time $\rightarrow O(n)$

TC $\Rightarrow$ efficient ~~Fun~~ algo $\theta(n)$

Summary

	Best Case	Worst Case
1. Brute force	$O(n)$	$O(n^2)$
2. Efficient Algo	$O(n)$	$O(n)$

Topic : Analysis of Algorithms

#Q. Arrange the following function in increasing order of their growth rate?

$f_1 \rightarrow n\log n$, $f_2 \rightarrow n^2$, $f_3 \rightarrow e^n$, $f_4 \rightarrow n^{3/2}$, $f_5 \rightarrow n$, $f_6 \rightarrow (1/n)$, $f_7 \rightarrow 2^n$

$\downarrow$ decr

$\boxed{n < n^{3/2} < n^2}$

A. $f_2, f_1, f_4, f_3, f_5, f_7, f_6$ ✗

B. $f_6, f_5, f_1, f_4, f_2, f_3, f_7$ ✗

C. $f_6, f_5, f_1, f_4, f_2, f_7, f_3$ ✓ $\longrightarrow$ HW

D. $f_6, f_5, f_1, f_4, f_7, f_3, f_2$ ✗

$e^n > 2^n$

$e > 2$

Topic : Analysis of Algorithms

#Q. Arrange in decreasing order of growth.

$f_1 \rightarrow n^{1/3},\ f_2 \rightarrow n,\ \cancel{f_2 \rightarrow (\log n)^{10}},\ f_3 \rightarrow n^{7/4},\ f_4 \rightarrow (2.002)^n,\ f_5 \rightarrow e^n$

$f_5 \rightarrow f_4$

$f_1 < f_2 < f_3$

A $f_5 > f_4 > f_1 > f_3 > f_2$

B $f_4 > f_5 > f_2 > f_3 > f_1$

C $f_4 > f_5 > f_3 > f_2 > f_1$

D $f_5 > f_4 > f_3 > f_2 > f_1$

Frame work to determine the time complexity of recursive algorithm.

Topic : Analysis of Algorithms

Recursion:

Recursion stack

n! = n * (n-1) * (n-2) 1

5! = f(5) = 5*f(4) = 5*24

↓

4*f(3) = 4*6

↓

3*f(2) = 3*2

↓

2*f(1) = 2*1

f(1)
f(2)
f(3)
f(4)
f(5)

Base condition terminating condition f(1) =1

Topic : Analysis of Algorithms

Framework to determine TC of Recursive algo.

3 Steps:-

Step-1. Find (Recursive equation for TC) for the recursive code.

Step-2. Solve this Recursive equation to get mathematical expression/value

Step-2. Apply Asymptotic notation on this value.

Topic : Analysis of Algorithms

TC Recursive equation

1. Back substitution method

- ➢ General approach
- ➢ Gives
  - Values of recurrence
  - Also give TC→ O,Θ,Ω

2. Master Method/ Theorem

Work in specific case

- Only gives final TC but not value of recurrence

3. Recursive Tree Approx ach

TC(Final but no. value)

Example:-

```
Algo f(n)
{
  if (n==1)
  return 1;
  return n*f(n-1);
}
```

T(n)

const

T(n-1)

Topic : Analysis of Algorithms

Let T(n) Represent the time complexity of f(n)

$$T(n) = T(n-1) + C \qquad n \geq 1$$

$$T(n) = C_1 \, , \qquad n = 1$$

Topic : Analysis of Algorithms

Eg.1.

```
Algo AJ1(n)          T(n)
{
  if (n==1)    } a
  return;              c1  | c2
  AJ2(n);      -> c        | O(n)
  AJ1(n–1);
}
```

$\downarrow$

T(n-1)

Case-1:

Assume AJ2 (n) takes → O(1) time

S1 :

$$\boxed{T(n) = T(n-1) + a}, \quad n > 1$$

$$T(n) = b, \quad n = 1$$

$$T(n) = T(n-1) + a \quad \text{—— (1)}$$

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

$$T(n) = T(n-2) + 2a \quad \text{—— (2)}$$

$$T(n) = T(n-3) + 3a \quad \text{—— (3)}$$

$\vdots$

$$T(n) = T(n-k) + k * a \quad \text{—— (4)}$$

For B.C, $T(1) \Rightarrow n-k=1$

$$\boxed{k = n-1}$$

$$T(n) = T(1) + (n-1) * a \quad \text{—— (5)}$$

$$T(n) = \boxed{b + (n-1) * a}$$

value of Rec

S3: $\boxed{T(n) = O(n)}$ ✓

Case-2:

Assume AJ2 (n) takes → O(n) time

S1

$$T(n) = T(n-1) + \underbrace{O(n) + a}, \; n > 1$$

$$T(n) = b, \; n = 1$$

$$\boxed{T(n) = T(n-1) + n, \; n > 1}$$

S2:

$$T(n) = T(n-1) + n \quad \text{--- } 1$$

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

$$T(n) = T(n-2) + (n-1) + n \quad \text{--- } 2$$

$$T(n) = T(n-3) + n + (n-1) + (n-2) \quad \text{--- } 3$$

$$T(n) = T(n-k) + n + (n-1) \cdots + (n-(k-1)) \quad \text{--- } 4$$

for B.C

$n - k = 1$

$k = n-1$

$$T(n) = T(1) + n + (n-1) \cdots + (n-(n-2))$$

$$T(n) = T(1) + [n + (n-1) \dots + 2]$$

$$= b + \left[\frac{n(n+1)}{2} - 1\right]$$

$$T(n) = \left[b + \frac{n^2 - n}{2}\right] \longrightarrow S3 \quad \boxed{T(n) = O(n^2)}$$

Topic : Analysis of Algorithms

1. Algo AJ(n) $\longrightarrow T(n)$

```
{
    if (n ==2)
        return 2
    else
        return AJ(√n)
}
```

b

$T(\sqrt{n})$

$$T(n) = T(\sqrt{n}) + a, \quad n > 2$$

$$T(n) = b, \quad n = 2$$

$$T(n) = T(\sqrt{n}) + a \quad \text{--- } (1)$$

$$T(n) = T(n^{1/2}) + a$$

$$T(n^{1/2}) = T((n^{1/2})^{1/2}) + a$$

$$T(n) = T(n^{1/2^2}) + 2a \quad \text{--- } (2)$$

$$T(n) = T(n^{1/2^3}) + 3a \quad \text{--- } (3)$$

$$\vdots$$

$$T(n) = T(n^{1/2^k}) + k * a \quad \text{--- } (4)$$

for B.C, $n^{1/2^k} = 2$

$$\frac{1}{2^k} \log n = 1$$

$$\log n = 2^k$$

$$k = \log(\log n)$$

$$T(n) = T(2) + \log(\log n) * a$$

$$T(n) = \underline{b + \log(\log n) * a}$$

S3

$$\underline{TC : O(\log(\log n))}$$

Topic : Analysis of Algorithms

2. Algo AJ(n) $\rightarrow T(n)$

```
{
    if (n ==1)
        return 1;
    else
        return ((AJ(n-1) + AJ (n-1));
}
```

b

$\underbrace{AJ(n-1)}_{T(n-1)} + \underbrace{AJ(n-1)}_{T(n-1)}$

$$\boxed{\begin{aligned} T(n) &= 2T(n-1) + a, \; n>1 \\ T(n) &= b, \; n=1 \end{aligned}}$$

H.W

THANK - YOU