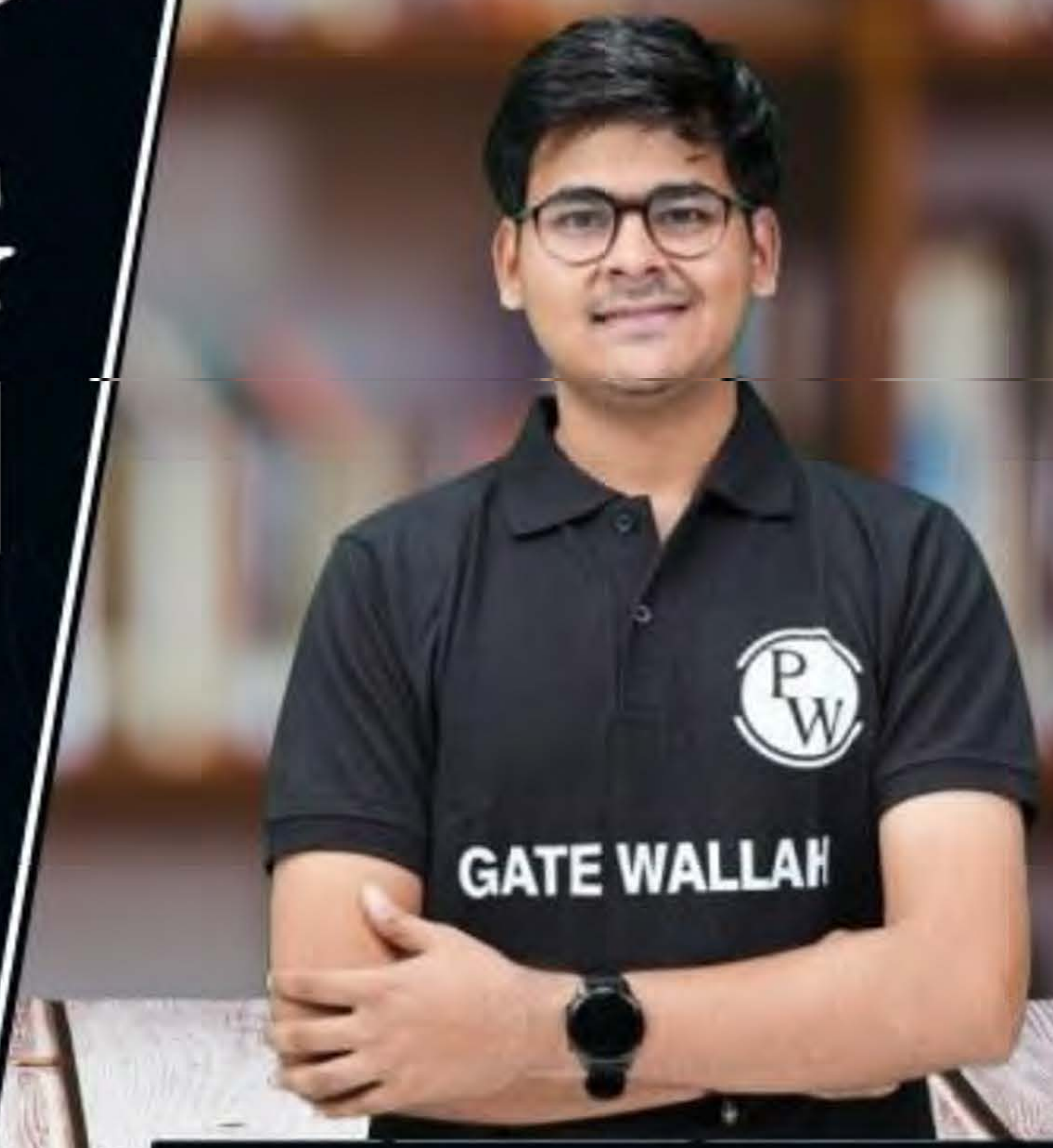
CS & IT ENGINEERING

Algorithm

Analysis of Algorithms

Lecture No.- 03



By- Aditya sir

Recap of Previous Lecture

Topic Apriori Analysis → 1) Step-count

2) Order of Magnitude

Topic Types of Analysis

Topic Worst-case and Best-Case Behavior

↳ Linear Search

$$\boxed{\begin{array}{l} BC \to O(1) \\ WC \to O(n) \end{array}}$$

Topics to be covered

Topic Asymptotic Notations ☆ Imp

Topic Big-Oh, Big Omega, Theta Notations

Topic : Asymptotic Notations

Revise:

Time complexity (n)

Logarithmic | Polynomial | Exponential

$\log(n)$

$\log(n^2)$

n^x

n^2, n^3, n^4

$\sqrt{n}$

x^n

$\Rightarrow 2^n, 3^n, \ldots n^n$

☆ Asymptotic Notations:

→ The bounds/range of the function can be represented using these notations.

* Types of Bounds :-

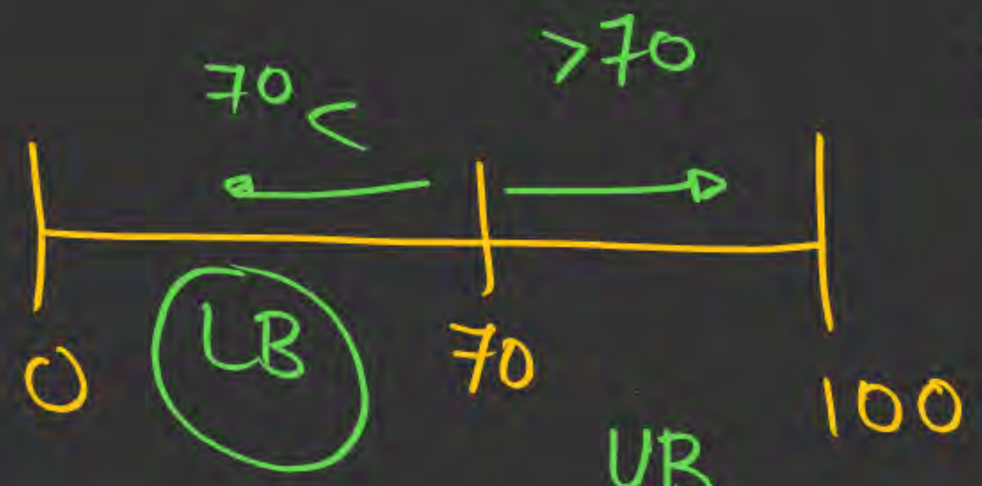
upper limit (max) ← 1. Upper Bound

Lower limit (min) ← 2. Lower Bound

1. Lose Bound
2. Tight Bound

Class $\rightarrow$ Marks

50 $\rightarrow$ LB



UB

71, 72, 73 ---- $\rightarrow$ UB

72 $\rightarrow$ UB

Loose LB

Loose UB

1 . . . . . 66 67 68 69 70 71 72 73 100

Lower Bound

(LB)

tight
Lower Bound

Upper Bound

(UB)

Tight Upper
Bound

* Asymptotic Notations

① Big Notations

① → UB : Big Oh → O

② → LB : Big Omega → Ω

② Small / Little Notations

③ → UB → Small Oh → o

④ → LB → Small omega → ω

$\Big\}$ → Lose Bound

⑤ Theta : Θ → Tight Bound

Topic : Asymptotic Notations

Big Oh → O

→ Let ‘f’ and ‘g’ be functions from the set g integers/real to reals number;

1. **Big-Oh(O):** Upper bound (UB)

f(n) is O(g(n)) if there exists some constant $c > 0$ and $n_0 > 0$ such that $f(n) \le c.g(n)$, whenever $n \ge n_0$

f, g → +ve functions

c, n_0 → positive constants

$$f(n) = O(g(n))$$

if

$$f(n) \le c * g(n), \quad c > 0$$

$$n > n_0$$

example :-

prev lecture Algo AJ( )

→ 1 order of Magnitude

$\underline{\underline{f(n)}} \Rightarrow n^2+n+1$

$1+n \le n^2+n^2$

$n=2 \quad 1+2 \le 2^2+2^2$ ✓

$n=3 \quad 1+3 \le 3^2+3^2$ ✓

$1 \le n^2$

$1+\underline{\underline{n}} \le n^2+\underline{\underline{n^2}}$

$1+n+n^2 \le \underline{n^2}+\underline{n^2}+\underline{n^2}$, $n \ge 2 \rightarrow n_0$

$\underbrace{1+n+n^2}_{f(n)} \le \underbrace{3}_{c} * \underbrace{n^2}_{g(n)}$

$f(n) \le c * g(n)$?

$\Downarrow$

Hence $f(n) = O(g(n))$

$1+n+n^2 = O(n^2)$

Tight UB

$1+n+n^2 \rightarrow \boxed{O(n^2)}$

$1+n+n^2 \rightarrow O(n^3)$?

$1+n+n^2 \rightarrow O(n^4)$

(n^5)

$\vdots$

Loose UB

$1+n+n^2 \leq c * n^3$

$\downarrow$

$1+n+n^2 \leq 3 * n^3$ ✓

Topic : Asymptotic Notations

Imp

Whenever we determine the upper bound and lower bound, we should find that function 'g' which is closest to the given function

Tight

Flight (Non-Stop)

Nagpur ⟶ Delhi

Loose Upper Bound ⟵ P1 ⟶ < 1 year

P2 ⟶ < 1 week

UB

Tight Upper Bound ⟵ most useful info ⟵ $\boxed{P3 \longrightarrow < 5\,hr}$

eg2 : Tight Lower Bound

Tarzan

Friends

Loose lower Bound

F1 : > 50 Rs ✓

F2 : > 5000 Rs ✓

F3 : > 5L Rs ✓

Closest lower Bound ← Tight lower Bound

Shortcut

$$f(n) = \underbrace{n^2 + n + 1} \rightarrow F(n) = O(n^2)$$

dominating term → term with Highest rate of growth

$$f(n) = 5n^3 + 8n + 7$$

$$f(n) = O(5n^3) \rightarrow \boxed{O(n^3)}$$

✓

Startup : UnicornX

2024 → 2 users/customers

2027 → 100 "

2028 → 10,000 "

2035 → 10,0000000 "

insignificant

dominating

≈ 10 cr users

Algo AJSir(n)

{

}

① Step-count method

$T(n) = 4n^2 + 8n + 8$

$O(4n^2) \rightarrow O(n^2)$

② Order of magnitude

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

$O(n^2)$

Topic : Asymptotic Notations

2. **Big-Omega(Ω):** Lower Bound

some constant

f(n) in Ω(g(n)) iff there exists ~~consist~~ 'c' and 'n_0' such that

$f(n) \geq c.g(n)$, whenever $n \geq n_0$

$1 > 0$ ✓

$$\rightarrow \quad f(n) = \Omega(g(n))$$

$$\text{iff}, \quad f(n) \geq c * g(n), \quad n \geq n_0$$

$$f(n) = 8n^2 + 3n + 5$$

Tight lower Bound

$$f(n) = 1 + n + n^2 \geq 1 * n^2 \longrightarrow \boxed{\Omega(n^2)}$$

$$1 + n + n^2 \geq 1 * n \longrightarrow \Omega(n)$$

$$1 + n + n^2 \geq 1 * \sqrt{n} \longrightarrow \Omega(\sqrt{n})$$

$$1 + n + n^2 \geq 1 * 1 \longrightarrow \Omega(1)$$

Loose LB

$$f(n) = n^2 + n + 1$$

$$\underbrace{n^2 + n + 1}_{f(n)} \geq \underbrace{1}_{c} * \underbrace{n^2}_{g(n)}, \quad n \geq 2$$

$$f(n) \geq c * g(n)$$

$$f(n) = \Omega(g(n))$$

Hence $\boxed{1 + n + n^2 = \Omega(n^2)}$

UB

72

Tight ← 71

70

69 → Tight

68

⋮

LB

Topic : Asymptotic Notations

3. **Theta(θ):** Tight Bound:

$f(n)$ is $\theta(g(n))$ iff $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$

$C_1.g(n) \le f(n) \le C_2.g(n)$

$f(n) \le C_1 * g(n)$

if $f(n) = O(g(n))$

and

$f(n) \ge C_2 * g(n)$ $f(n) = \Omega(g(n))$

$\longrightarrow f(n) = \Theta(g(n))$

eg:- : $f(n) = n^2+n+1$

$n^2+n+1 \leq 3*n^2$

$f(n) = O(n^2)$

$c_1 = 3$

$n^2+n+1 \geqslant 1*n^2$

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

$$\underbrace{1*n^2}_{c_1,\ g(n)} \leqslant \underbrace{1+n+n^2}_{f(n)} \leqslant \underbrace{3*n^2}_{c_2,\ g(n)}$$

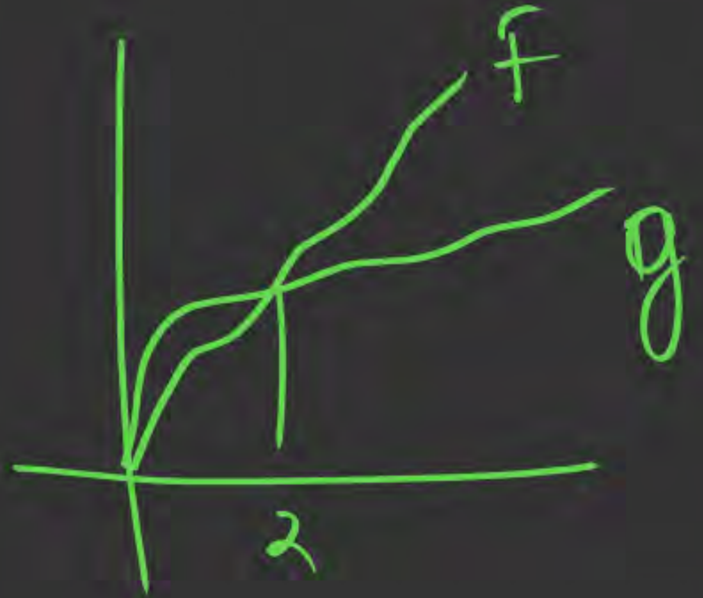
$f(n) = O(n^2)$

and

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

$\boxed{f(n) = \Theta(n^2)}$

Tight Bound



$n^2+n+1 \geq 1 * n^2 \; , \quad n \geq 2$

2^2+2+1 $\qquad n=2$

$4+2+1$

$=7 \geq 4$ ✓

$3^2+3+1 \geq 9$ $\qquad n=3$

$9+4$

$13 \geq 9$ ✓

eg: n^2 vs 2^n

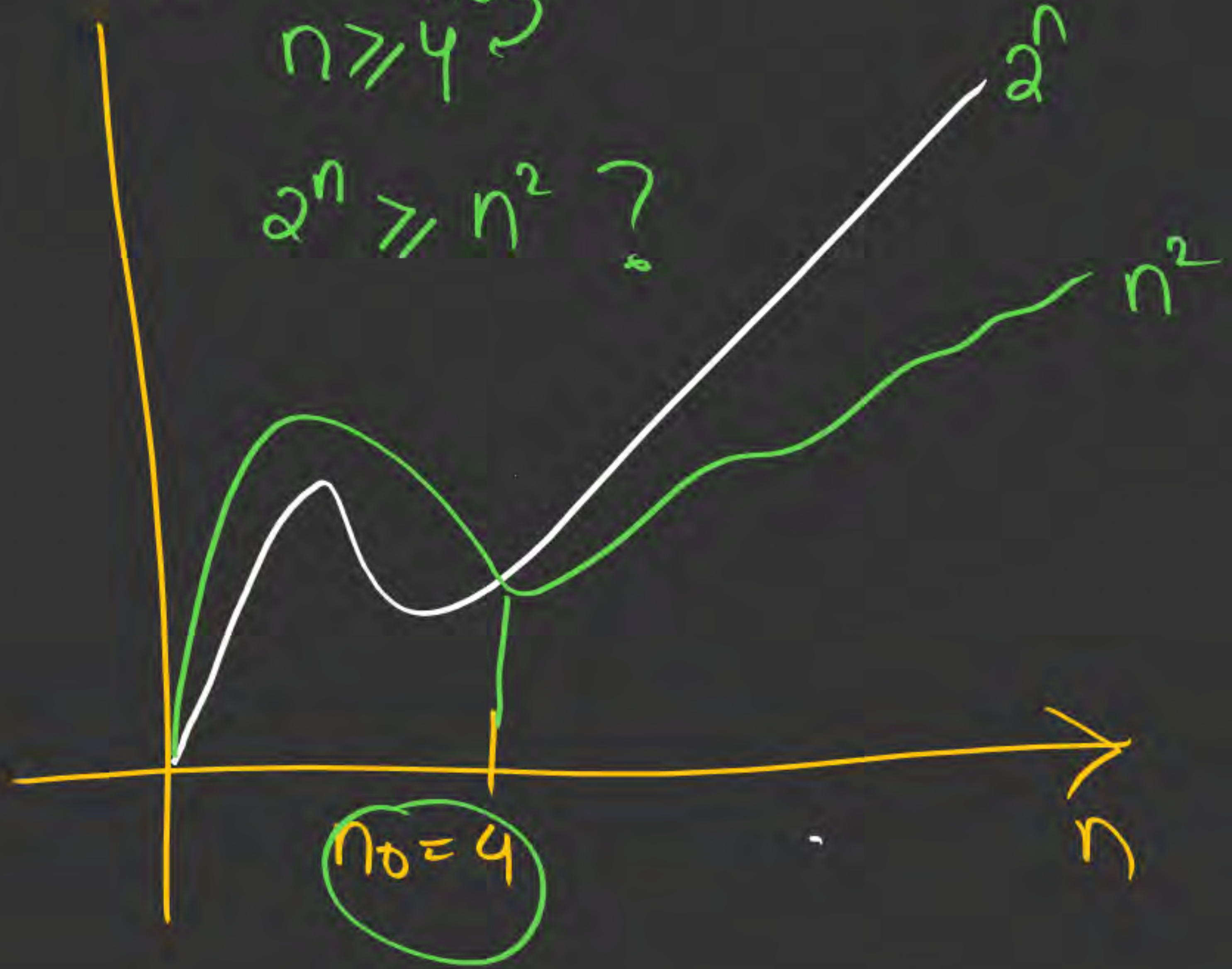
n	n^2	2^n
1	1	2 ✓
2	4	= 4
3	9 ✓	8
4	16	16
5	25	< 32
6	36	64
7	49	128

$n_0 = 1$ cr
$n_0 = 10$ cr
$n_0 = 1$B

understanding

For all $n \geq \overset{n_0}{4}$?

$2^n \geq n^2$?



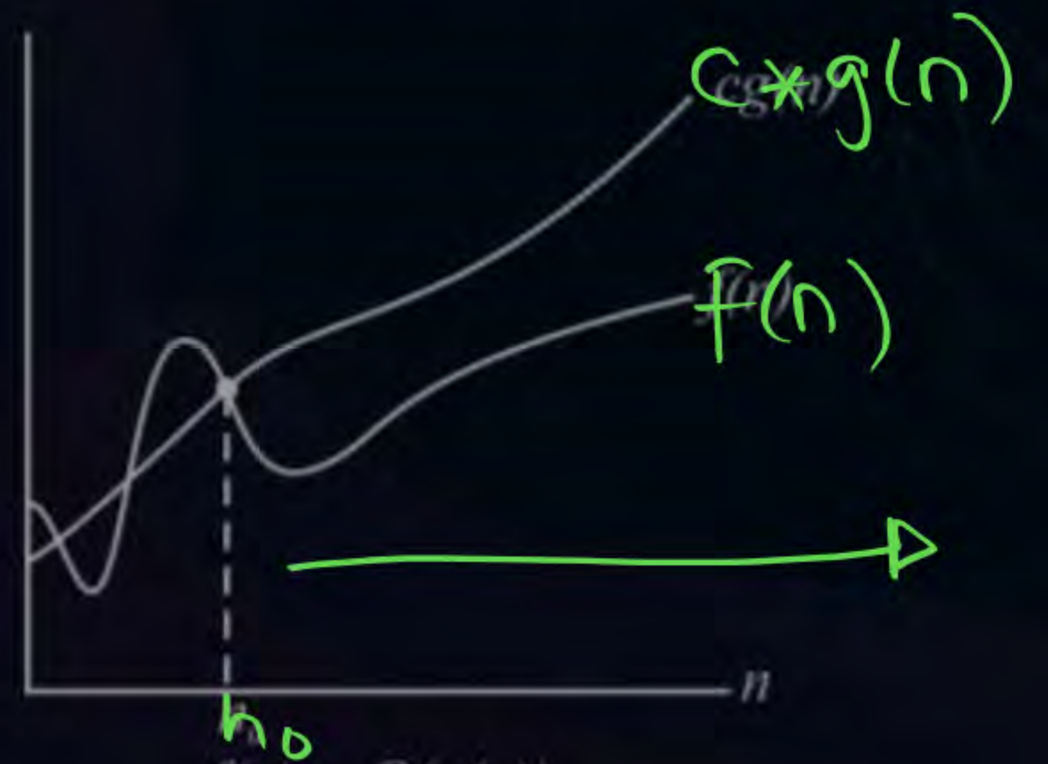
Topic : Analysis of Algorithms

$O(g(n)) = \{f(n)$: There exist positive constant c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$.

We write $f(n) = O(g(n)$ to indicate that a function $f(n)$ is a member of the set $O(g(n))$. Note that $f(n) = \theta(g(n))$ implies $f(n) = O(g(n))$. Since θ-notation is a stronger notation that O-notation. Written set-theoretically.

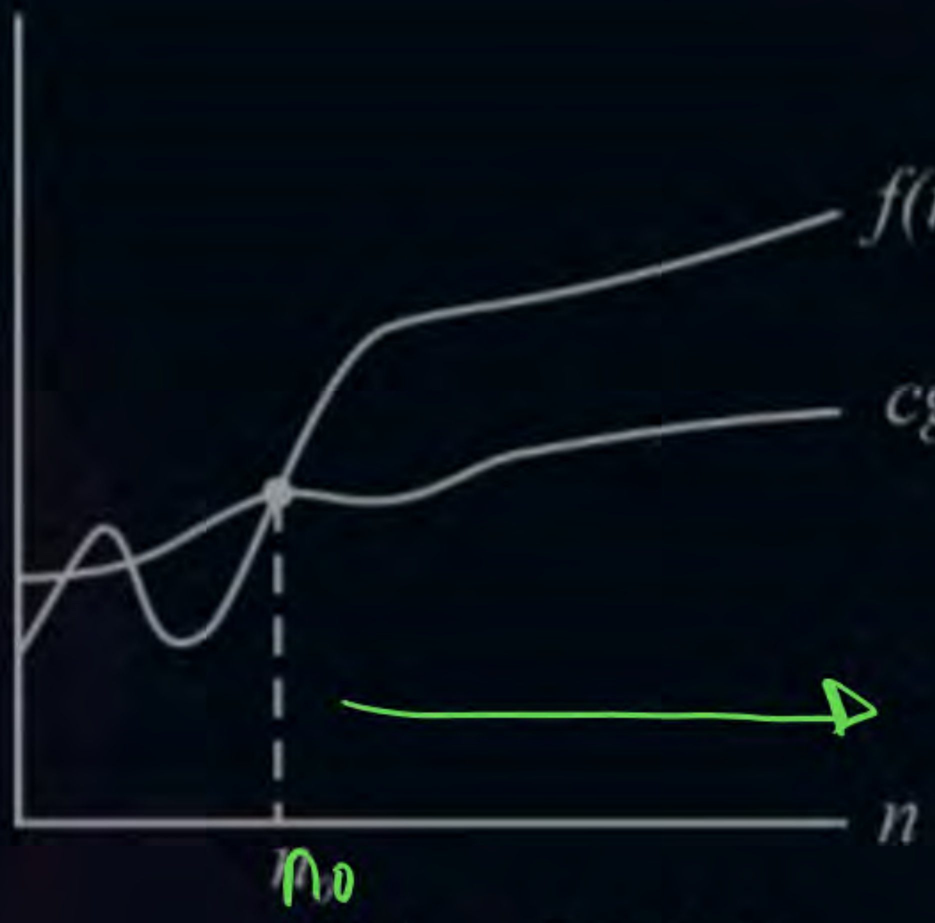
$f(n) = O(g(n))$

$n \geq n_0$



$f(n) = O(g(n))$

Topic : Analysis of Algorithms



$f(n) = \Omega(g(n))$

$f(n) \rightarrow F(n)$

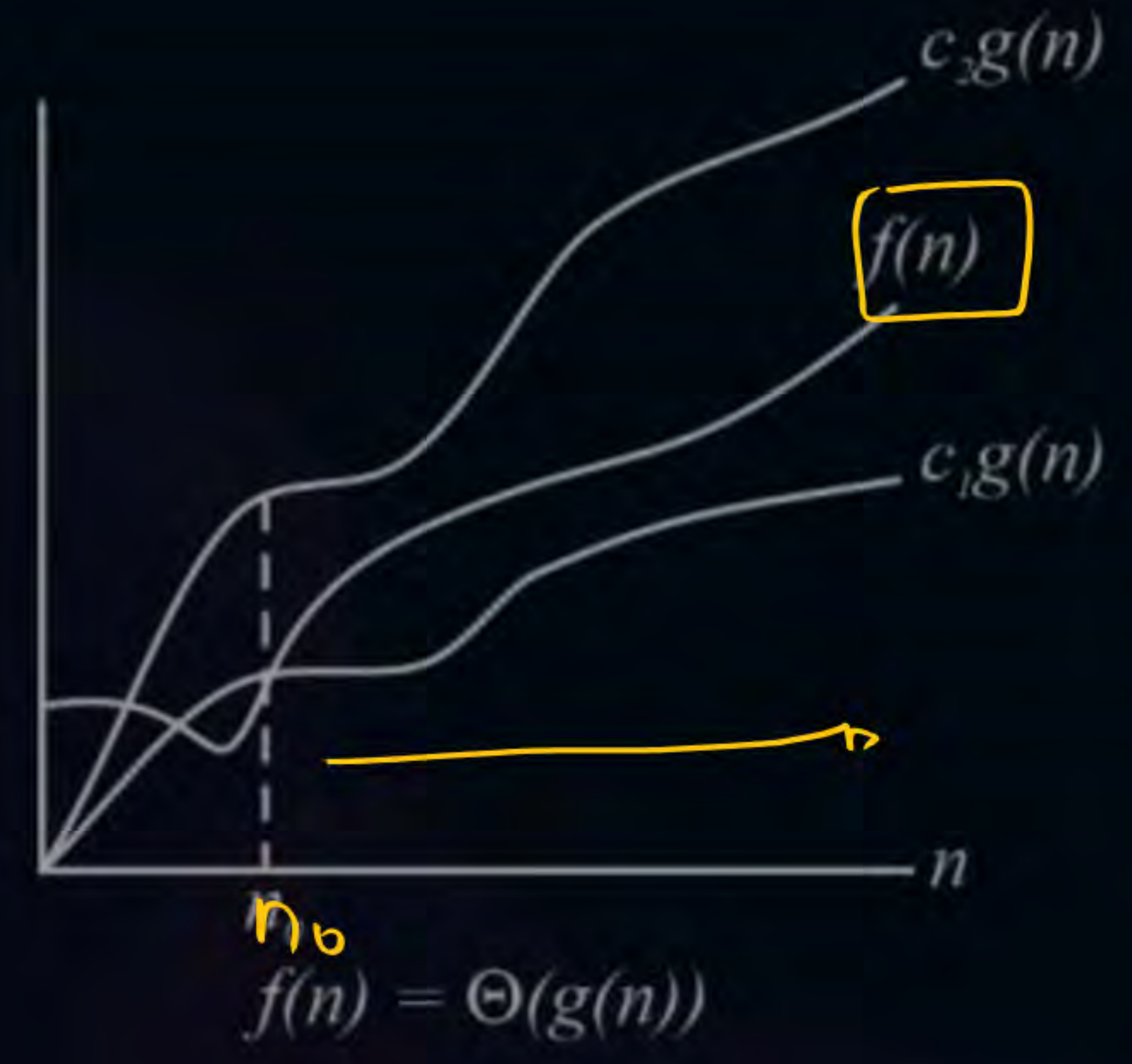
$cg(n) \rightarrow c * g(n)$

$F(n) \geq c * g(n)$

for all $n \geq n_0$

$f(n) = \Omega(g(n))$

Topic : Analysis of Algorithms



$$c_1 \times g(n) \le f(n) \le c_2 * g(n)$$
$$\Downarrow$$
$$\boxed{f(n) = \Theta(g(n))}$$

Topic : Analysis of Algorithms

→ Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.

→ Big O is a member of a family of notations invented by Paul Bachmann, Edmund Landau, and others, collectively called Bachmann-Landau notation or asymptotic notation. The letter O was chosen by Bachmann to stand for Ordnung, meaning the order of approximation.

In computer science, big O notation is used to classify algorithms according to how their run time or space requirements grow as the input size grows.

→ In analytic number theory, big O notation is used to express a bound on the difference between an arithmetical function and a better understood approximation; a famous example of such a difference is the remainder term in the prime number theorem.

Topic : Analysis of Algorithms

approx

- Big O notation is also used in many other fields to provides similar estimates.
- Big O notation characterizes functions according to their growth rates: different functions with the same asymptotic growth rate may be represented using the same O notation. The letter O is used because the growth rate of a function is also referred to as the order of the function. A description of a function in terms of big O notation usually only provides an upper bound on the growth rate of the function.

Associated with big O notation are several related notations, using the symbols O, Ω, ω and θ, to describe other kinds of bounds on asymptotic growth rates.

✓ O ✓ Ω o ω Θ ✓

next lec

$f(n) = 8n^2 + 2n \rightarrow O(n^2)$

$g(n) = 5n^2 + 6n \rightarrow O(n^2)$

Summary:

Asymptotic Notations

Big

Small

O, Ω Θ o, ω

THANK - YOU