CS & IT ENGINEERING

Algorithms

Analysis of Algorithms

Lecture No.- 02

By- Dr. Khaleel Khan Sir

Recap of Previous Lecture

- **Topic** Introduction to Course
- **Topic** Algorithm Concept
- **Topic** Algorithm Lifecycle Steps
- **Topic**
- **Topic**

Topics to be Covered

Topics

Types of Analysis

Asymptotic Notations

Apriori Analysis

↓

Math Fn (w.r.to Input Size)

Poly	Expo

(Slow rate of Growth) Poly n^x $x \geq 0$

↓

Runs fast

$\langle n, c, n^2, n^3, n^4$ $(\log n, \log\log n) \rangle$

Expo a^n $[a > 1]$ (Faster rate of Growth)

Runs Slow

$\langle 2^n, 4^n, n^n \ldots \rangle$

Topic: Analysis of Algorithms

$\underline{\underline{Ex}}$: (Poly) n^2 ; 2^n (Expo)

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

Types of Analysis / Behaviour of Algorithm :

For a Fixed value of 'n' (Input Size) the Algo can have different behaviour, for different input classes (arrangements)

I : (Size n)

Input classes

I_1 (Inc) I_2 (Dcr) I_3 (Random) I_k (   )

(i) Best Case : The I/p class for which the Algo does Minimum work & thereby taking Min. Time,

Ex: (i) Linear Search

Key : x

Elem is found @ first position: $O(1)$

(ii) Quicksort : when elements are not in Sorted order;

$O(n \cdot \log n)$ ✓

(ii) <u>Worst Case</u>: The I/p class for which Algo does Max work & hence takes Max Time;

<u>Ex</u>: 1) Linear Search;

	1	2	3			n
A					..	○

'x' : $O(n)$

2) <u>Quicksort</u> : Elements of the array are in Sorted order

$O(n^2)$

(iii) Average case:

$$A(n) = \sum_{i=1}^{K} P_i * t_i$$

1) determine all Input classes
$\langle I_1, I_2 \ldots I_K \rangle$

2) determine the Time for each I/P class
$\langle \underbrace{I_1, I_2 \ldots I_K}_{t_1 \; t_2 \; \ldots \; t_K} \rangle$

3) Assoc. the Prob with each I/P class
$\langle \underbrace{I_1, I_2 \ldots I_K}_{P_1 \; P_2 \; \ldots \; P_K} \rangle$

Topic: Analysis of Algorithms

If $B(n)$: Best Case Time
$A(n)$: Avg. Case "
$W(n)$: worst Case "

$$\boxed{B(n) \leq A(n) \leq W(n)}$$

1) $B(n) = A(n) = W(n)$: Merge Sort ; Sel. Sort ;

2) $\boxed{B(n) = A(n)} < W(n)$: Quicksort $(n\log n, n^2)$

3) $B(n) < (A(n) = W(n))$: Linear Search $(1, n)$

4) $B(n) < A(n) < W(n)$:

Topic: Analysis of Algorithms

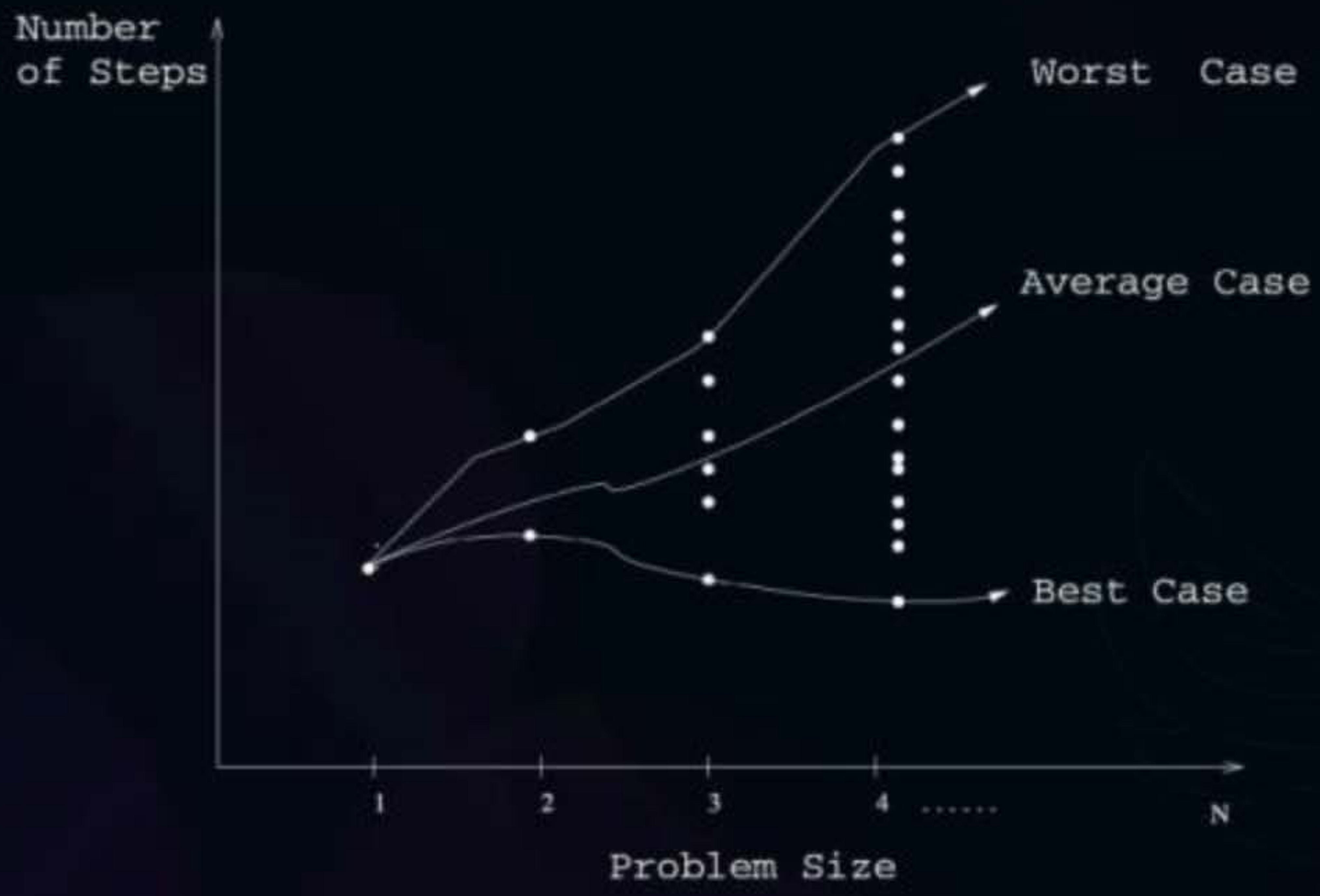
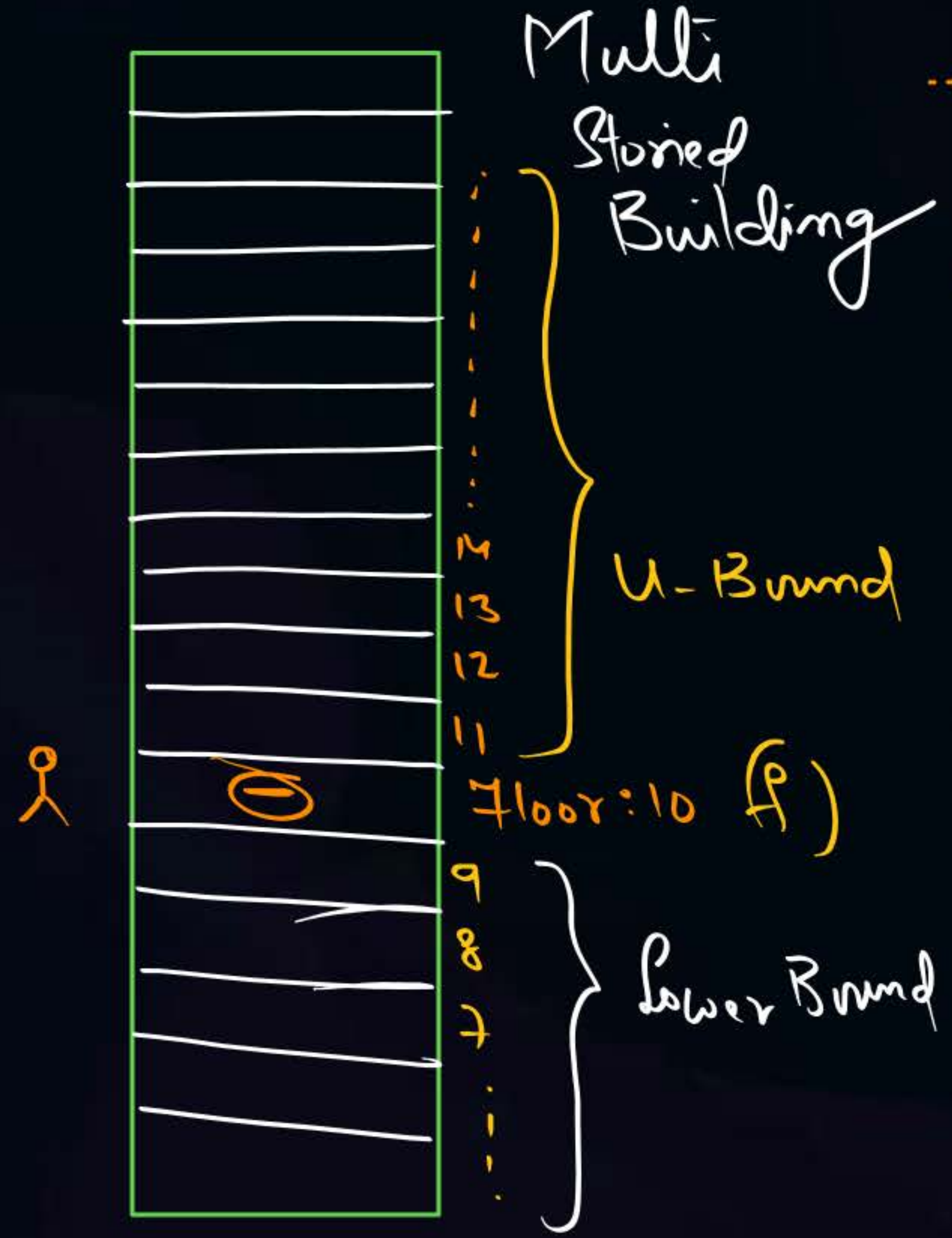


Figure 2.1: Best, worst, and average-case complexity

Topic: Analysis of Algorithms

- The Worst-case Complexity of the Algorithm is the function defined by the maximum number of steps taken in any instance of size n. This represents the curve passing through the highest point in each column.
- The Best-case Complexity of the Algorithm is the function defined by the minimum number of steps taken in any instance of size n. This represents the curve passing through the lowest point of each column.
- The Average-case complexity of the Algorithm, which is the function defined by the average number of steps over all instances of size n.

Topic: Analysis of Algorithms



Asymptotic Notations (ASN)

Repr. Bounds of fn's

- Upper
- Lower
- Tight

A.S.N

BIG

- UB 1) Big-oh (O)
- LB 2) Big-omega (Ω)
- TB 3) Theta (Θ)

SMALL | LITTLE

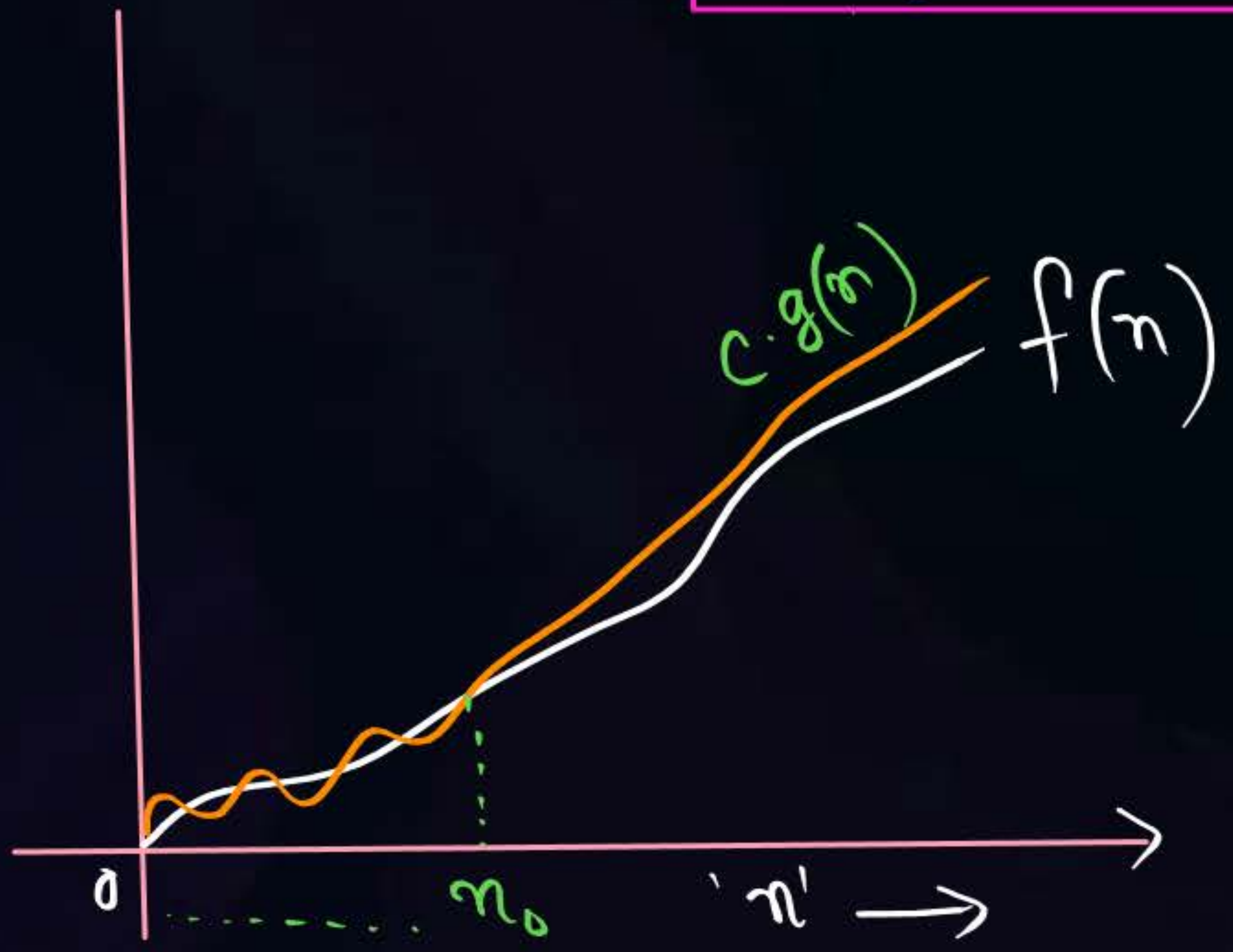
- Proper UB 4) Small-oh (o)
- 5) Small omega (ω) Proper LB

Topic: Analysis of Algorithms

Let f & g be functions from Integers or reals to real no's;

1) Big-oh (O) : upperBound of a fn';

f(n) is O(g(n)), iff there exists constants 'c' & n_0 Such that $\boxed{f(n) \le c \cdot g(n)}$, whenever $n > n_0$;



Ex:

1) $f(n) = 1 + n + n^2 \Rightarrow O(n^2)$

$$\boxed{\begin{aligned} 1+n+n^2 &< n^2+n^2+n^2 \\ &< 3 \cdot n^2 \quad , n>1 \end{aligned}}$$

$\underbrace{1+n+n^2}_{f(n)} \le \underbrace{3}_{c}\underbrace{n^2}_{g(n)}, \; n > \underbrace{1}_{n_0}$ $\therefore f(n) = O(n^2)$

1) $\underline{1+n+n^2} \le 3 \cdot n^2 \therefore O(n^2)$ ✓

2) $1+n+n^2 \le c_1 \cdot n^3 \therefore O(n^3)$ ✓

3) $1+n+n^2 \le c_2 \cdot n^4 \therefore O(n^4)$ ✓

$\vdots$

$O(n^{10})$ ✓

$\underline{Ex}$:

2) $f(n) = n$

$O(n)$ ✓

$n \le 2 \cdot n, n > 1$

$\le c_1 \cdot n^2, n > 1$

$\vdots$

3) $f(n) = \underline{100}$

$= C$

$O(1)$ ✓

$$\boxed{\underset{f}{100} \le \underset{c}{200} \cdot \underset{g}{1}}$$

Ex:

P_1

P_2

→ 30 minutes

→ 2 hrs ✓

→ 1 day

2)

Plot : 300 Sqyd

G+1

< 5 ; K ; vert . . . >

$f(n) \sim 53L$

1) Max. Amount

G.T <100L

2) Min. Amount

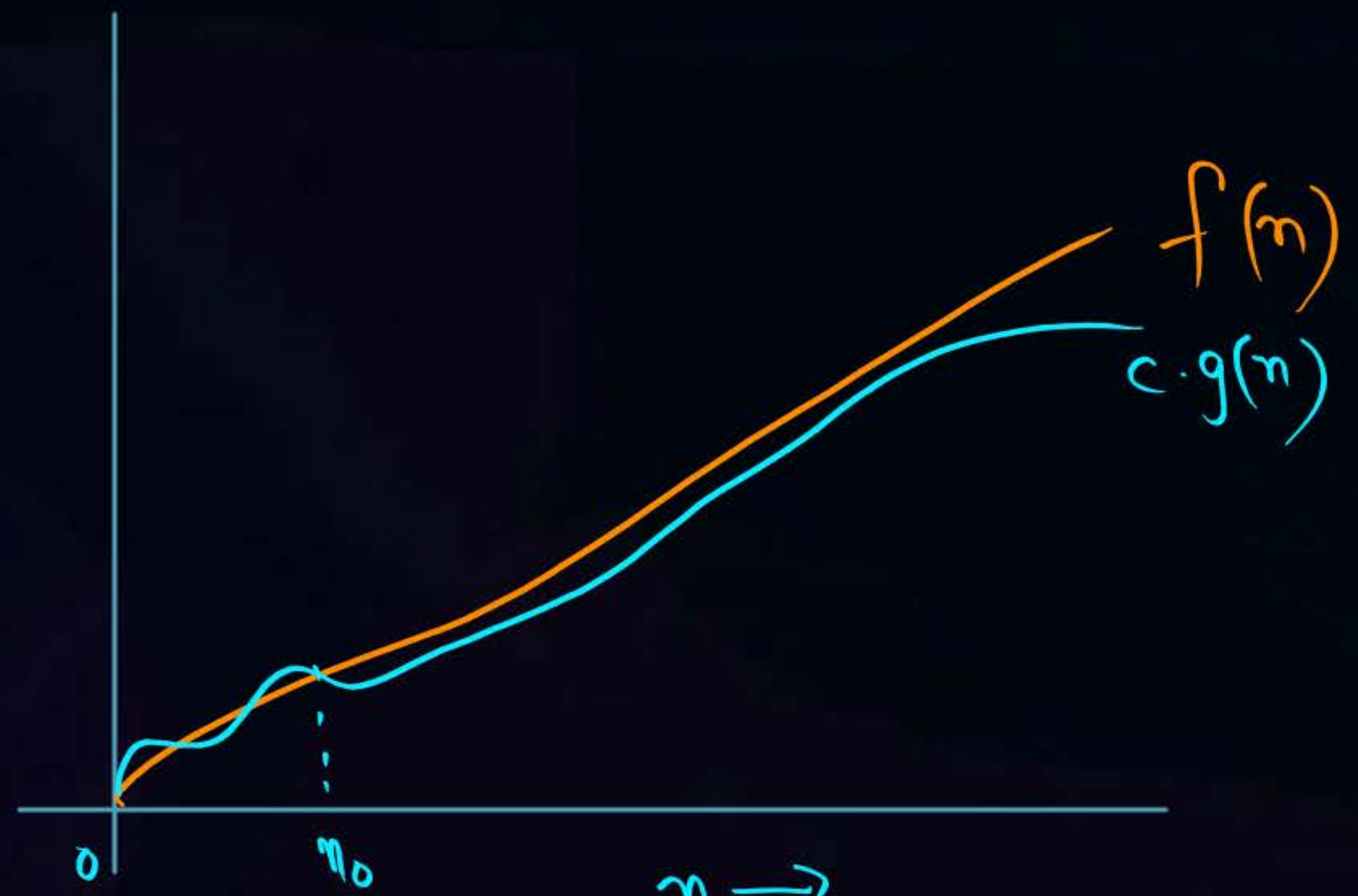
>(52L)

>5000

2) Big-omega (Ω): Lower Bound

$f(n)$ is $\Omega(g(n))$ iff there exists constants c & n_0 $[c>0;\ n_0>0]$

Such that $\boxed{f(n) \geq c \cdot g(n)}$, whenever $n > n_0$



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

$1 + n + n^2 \geq 1$, $n > 1$

$f(n)$ is $\Omega(1)$

$1 + n + n^2 \geq n$, $n > 1$

$f(n)$ is $\Omega(n)$

$\underbrace{1 + n + n^2}_{f} \geq \underset{c}{1} \cdot \underset{g}{n^2}$, $n > 1$

$f(n)$ is $\underline{\underline{\Omega(n^2)}}$

2) $f(n) = n$ $\begin{cases} O(n) \\ \Omega(1) ; \Omega(n) \end{cases}$ $\underline{n \leq 2 \cdot n}$

$n \geq 1 \cdot 1$

$n \geq c_1 \cdot n$

$n \geq \frac{1}{2} n, \ n > 1$

c

3) $f(n) = \underset{(c)}{100}$ $\begin{cases} O(1) \\ \Omega(1) \end{cases}$

$100 \leq 200 \cdot 1$

$100 \geq 5 \cdot 1$

$$f(n) = n + \log_2 n \quad \begin{cases} O(n) \\ \Omega(n) \end{cases}$$

$$n + \log n < n + n < 2n$$

$$n + \log n > 1 \cdot 1 \quad \therefore \Omega(1)$$

$$n + \log n > c \cdot \log n \quad \therefore \Omega(\log n)$$

$$n + \log n > 1 \cdot n \quad \therefore \Omega(n)$$

$$f(n) = \log n + \sqrt{n} \begin{cases} O(\sqrt{n}) & \log n + \sqrt{n} < 2\cdot\sqrt{n} \\ \Omega(\sqrt{n}) & \log n + \sqrt{n} > 1\cdot\sqrt{n} \end{cases}$$

$$1+n+n^2 \le 3\cdot n^2$$

$$1+n+n^2 \ge 1\cdot n^2$$

1) $f(n) = 1+n+n^2 \begin{cases} O(n^2) \\ \Omega(n^2) \end{cases}$

2) $f(n) = n \begin{cases} O(n) \\ \Omega(n) \end{cases}$

3) $f(n) = 100 \begin{cases} O(1) \\ \Omega(1) \end{cases}$

4) $f(n) = n + \log n \begin{cases} O(n) \\ \Omega(n) \end{cases}$

5) $f(n) = \log + \sqrt{n} \begin{cases} O(\sqrt{n}) \\ \Omega(\sqrt{n}) \end{cases}$

$$\sqrt{n} + \log n \le 2\cdot\sqrt{n}$$

$$\sqrt{n} + \log n \ge 1\cdot\sqrt{n}$$

3) Theta (Θ) : Tight Bound

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

$$\boxed{c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)}$$

$(c_1, c_2) > 0$

$n > n_0$

1) $1+n+n^2 \begin{cases} O(n^2) \\ \Omega(n^2) \end{cases} \therefore \Theta(n^2)$

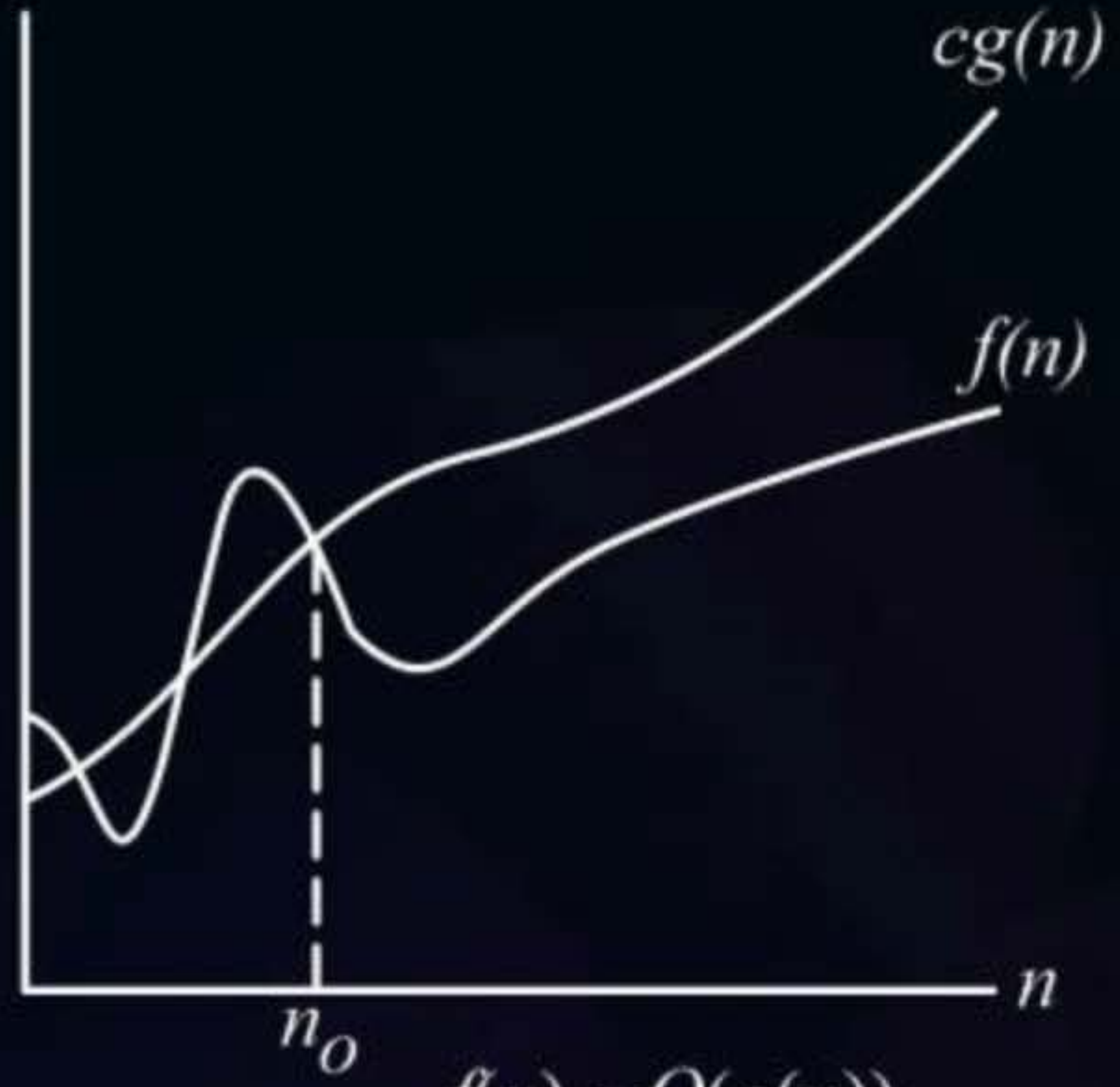
2) $100 \begin{cases} O(1) \\ \Omega(1) \end{cases} \therefore \Theta(1)$

3) $n+\log n \begin{cases} O(n) \\ \Omega(n) \end{cases} \Theta(n)$

The formal definitions associated with the Big Notation are as follows:

- $f(n) = O(g(n))$ means c . g(n) is an upper bound on f(n). Thus there exists some constant c such that f(n) is always $\leq$ c.g(n), for large enough n (i.e., n $\geq$ no for some constant n_0).
- $f(n) = \Omega(g(n))$ means c.g(n) is a lower bound on f(n). Thus there exists some constant c such that f(n) is always $\geq$ c. g(n), for all n $\geq$ no.

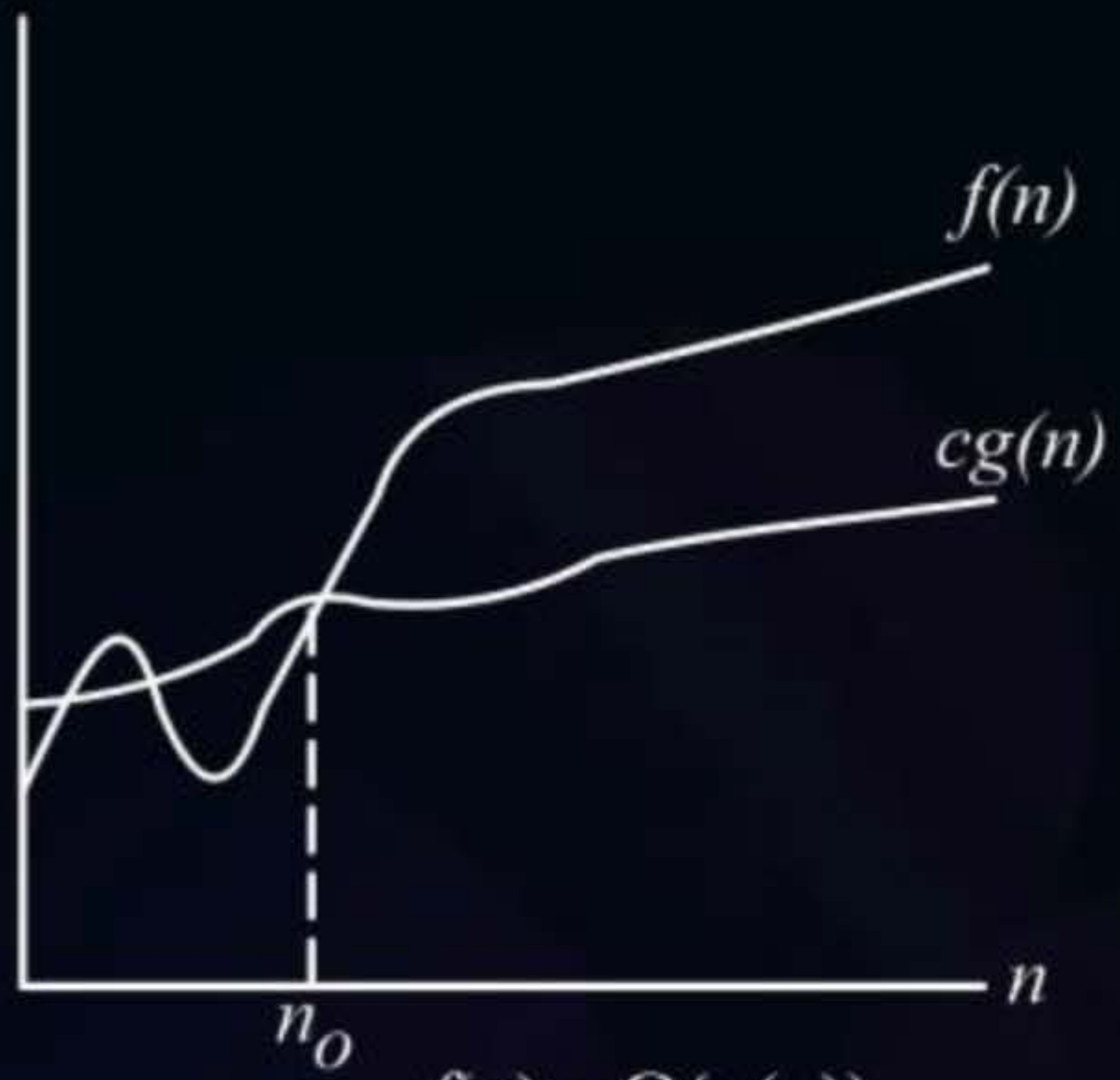
Topic: Analysis of Algorithms



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

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

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 notion than O-notation. Written set-theoretically, we have

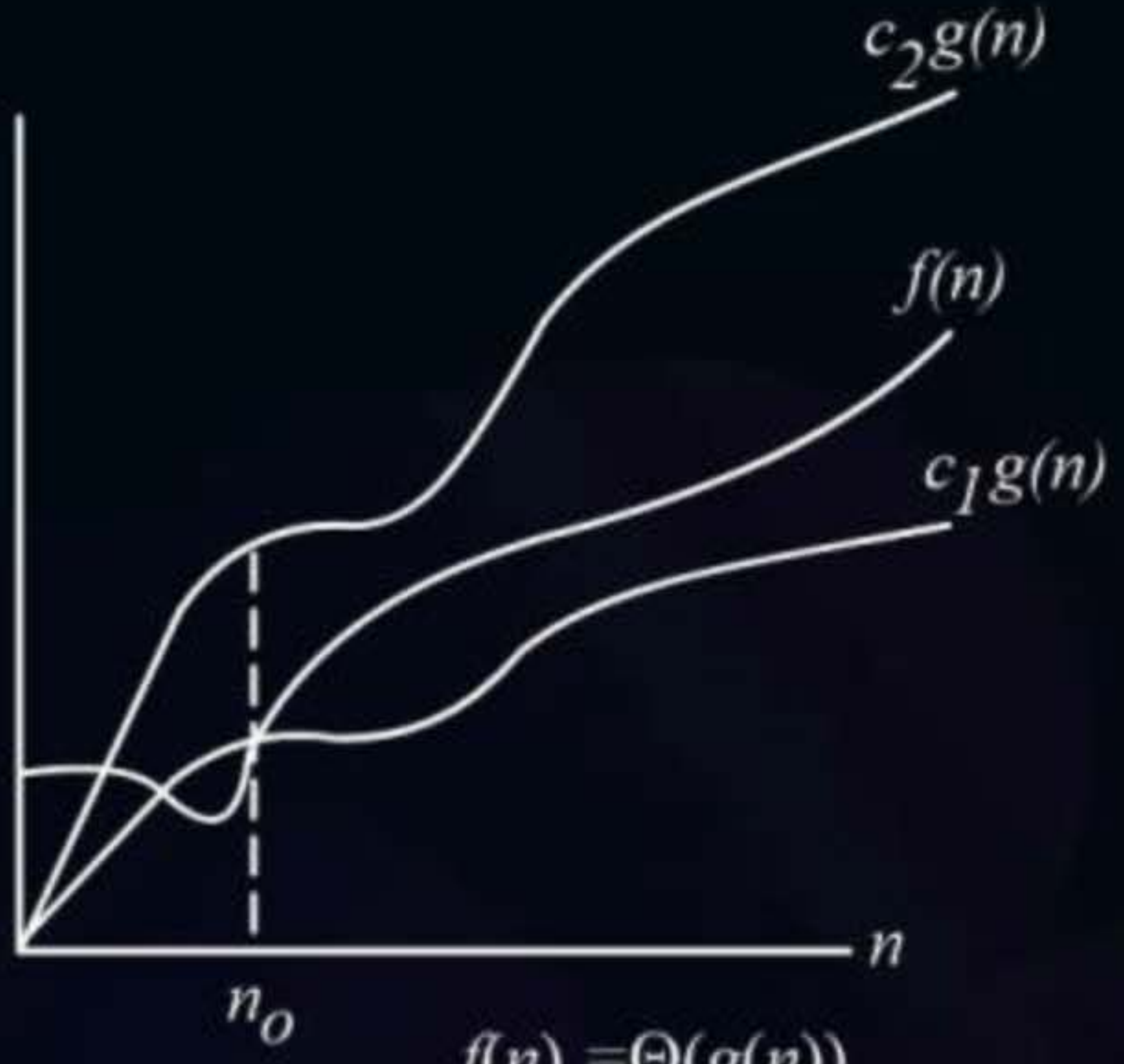


$f(n) = \Omega(g(n))$

Topic: Analysis of Algorithms

- $f(n) = \Theta(g(n))$ means $c_1 . g(n)$ is an upper bound on $f(n)$ and $c_2 . g(n)$ is a lower bound on $f(n)$, for all $n \geq no$. Thus there exist constants c_1 and c_2 such that $f(n) \leq c_1 . g(n)$ and $f(n) \geq c_2 . g(n)$. This means that $g(n)$ provides a nice, tight bound on $f(n)$.

Topic: Analysis of Algorithms



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

For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. ■

1) $f(n) = (n+c)^m, \ (c, m) > 0$ — $O(n^m)$ — $\Omega(n^m)$ $\Theta(n^m)$

$= (n+4)^2 = n^2 + 8n + 16$ $\begin{cases} O(n^2) \\ \Omega(n^2) \end{cases}$ $\Theta(n^2)$

2) $f(n) = \left(\frac{1}{n} + \log n\right)$ $\begin{cases} O(\log n) \\ \Omega(\log n) \end{cases}$ $\Theta(\log n)$

3) $f(n) = 1 - \frac{1}{n}$ $\begin{cases} O(1) \\ \Omega(1) \end{cases}$

4) $f(n) = \left[2^{\log_2 n}\right]$ $\qquad a^{\log_c b} = b^{\log_c a}$

$n^{\log_2 2} = n$ $\begin{cases} O(n) \\ \Omega(n) \end{cases}$ $\therefore \Theta(n)$

5) $f(n) = \left[2^{\sqrt{n} \cdot \log_2 n}\right]$

$= 2^{\log_2 (n^{\sqrt{n}})} = n^{\sqrt{n} \log_2 2}$

$= n^{\sqrt{n}}$ ✓

6) $f(n) = 2^{2n} \sim O(2^n)$ ✗

$O(4^n)$ ✓

7) $f(n) = \sum_{i=1}^{n} 1 = n$

$= O(n)$

8) $f(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$

9) $f(n) = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$

10) $f(n) = \sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 = O(n^4)$

G.P

11) $f(n) = \sum_{i=1}^{n} 2^i$

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

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

$O(2^n)$

12) $f(n) = \sum_{i=1}^{n} i \cdot 2^i = \boxed{(n-1) \cdot 2^{n+1} + 2}$

$O(n \cdot 2^n)$

13) $f(n) = \sum_{x=1}^{n} 1/x$ H.P

$\sim \int f(x)$

$\sim \int_1^n 1/x = [\log x]_1^n = (\log n)$

1) $2^{n+1} = 2 \cdot 2^n = O(2^n)$ ✓

2) $2^{2n} = (2^2)^n = \boxed{4^n > 2^n}$

$\longrightarrow O(4^n)$ ✓

10) $f(n) = \sum_{i=1}^{n} \sqrt{i} \sim \int_{i=1}^{n} i^{1/2} = \left[\frac{i^{3/2}}{3/2}\right]_1^n$

$$\int x^n = \frac{x^{n+1}}{n+1}$$

$$= \left[n^{3/2} - c\right]$$

$$= O\left(n^{3/2}\right) = O\left(n^{1.5}\right)$$

$$= O\left(n \cdot \sqrt{n}\right) ✓$$

11) $f(n) = \sum_{i=1}^{n} i^{-1/2}$

H/W

$$f(n) = \prod_{i=1}^{n} 1 = (1)^n = 1 = O(1) \checkmark$$

$$f(n) = \prod_{i=1}^{n} i = (1 \cdot 2 \cdot 3 \cdot 4 \cdots\cdots n) = n!$$

$$f(n) = n! = [n \cdot (n-1) \cdot (n-2) \cdots 1]$$

$$n \cdot (n-1) \cdot (n-2) \cdots 1 \leq n \cdot n \cdot n \cdots n$$

$$\leq n^n$$

$$\boxed{\therefore\ n! \text{ is } O(n^n)}$$

$$f(n) = \sum_{i=1}^{n} i = \cancel{O}(\Sigma i)$$

Can we say $n!$ is $\Omega(n^n)$? ✗

1) $n!$ is $O(n^n)$ ✓

2) $n! \neq \Omega(n^n)$

3) $n! = \Omega(n)$ ✓

$n! = \Omega(n^2)$ ✓

$n! = \Omega(2^n)$ ✓

1) <u>Big-oh</u>

$$f(n) \leq c \cdot g(n)$$ [f is Less than g]

(Smaller fn is always in the order of Bigger fn)

a) $f(n) = \log n$; $g(n) = n$

$f(n)$ is $O(g(n))$

b) $f(n) = n^4$; $g(n) = n^2$

$g(n)$ is $O(f(n))$

c) $f(n) = \sqrt{n}$; $g(n) = \log n$ $\therefore g(n)$ is $O(f(n))$

$$\log \sqrt{n} \qquad \log \log n$$

$$\log n^{1/2} \qquad \log \log n$$

$$\frac{1}{2} \cdot \log(n) > \log(\log n)$$

$$f(n) = n^2 \;<\; ; \; g(n) = n^3 \qquad [n^2 < n^3]$$

$$\log_2 n^2 \qquad \log_2 n^3$$

$$\underline{\underline{2}} \cdot \cancel{\log_2} n \qquad \underline{\underline{3}} \cdot \cancel{\log_2} n \qquad \underline{2 < 3}$$

$$O(1) \sim O(1) \quad \times$$

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

$$f(n) = \left[n^2 \cdot \log n\right] \quad ; \quad g(n) = \left(n \cdot \log^{10} n\right)$$

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

$$= \underline{\underline{n}} \qquad\qquad = \log^9 n = \underline{\underline{(\log n)^9}}$$

$$\log n \qquad \underset{A}{>} \qquad \log\left((\log n)^9\right)$$

$$n = 4 \qquad\qquad = 9 \cdot \log\log n \qquad n > n_0$$

$$\log_2 4 = 2$$

$f(n) = \underset{\text{Poly}}{\underline{n^2}}$; $g(n) = \underset{\text{Expo}}{\underline{2^n}}$

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

$= \log n^2 \qquad \log 2^n$

$= 2 \cdot \log n \qquad n \cdot \log_2 2$

$= O(\log n) < O(n)$

$n^2 < 2^n,\ n > 4$

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

Rows 1–4: Turbulent Behaviour

Rows 5–7: Non Turbulent $n^2 < 2^n$

Topic: Analysis of Algorithms

$\log x^y = y \cdot \log x$ ①

$\log xy = \log x + \log y$ ②

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

$\boxed{a^{\log_b^x} = x^{\log_b^a}}$ $*$

$2^{\log_2^n} = n^{\log_2^2} = n$

$\log n = \log_{10}^n$

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

$\log \frac{x}{y} = \log x - \log y$ $*$

$\log_b^x = \frac{\log_a^x}{\log_a^b}$ $*$

$a = b^{\log_b a}$

$\log_c(ab) = \log_c a + \log_c b$

$\log_b a^n = n \cdot \log_b a$

$\log_b a = \frac{\log_c a}{\log_c b}$

$\boxed{\log_b 1/a = -\log_b a}$

$\boxed{\log_b a = \frac{1}{\log_a b}}$

$a^{\log_b c} = c^{\log_b a}$

2 mins Summary

Topic One

Topic Two

Topic Three

Topic Four

Topic Five

THANK - YOU