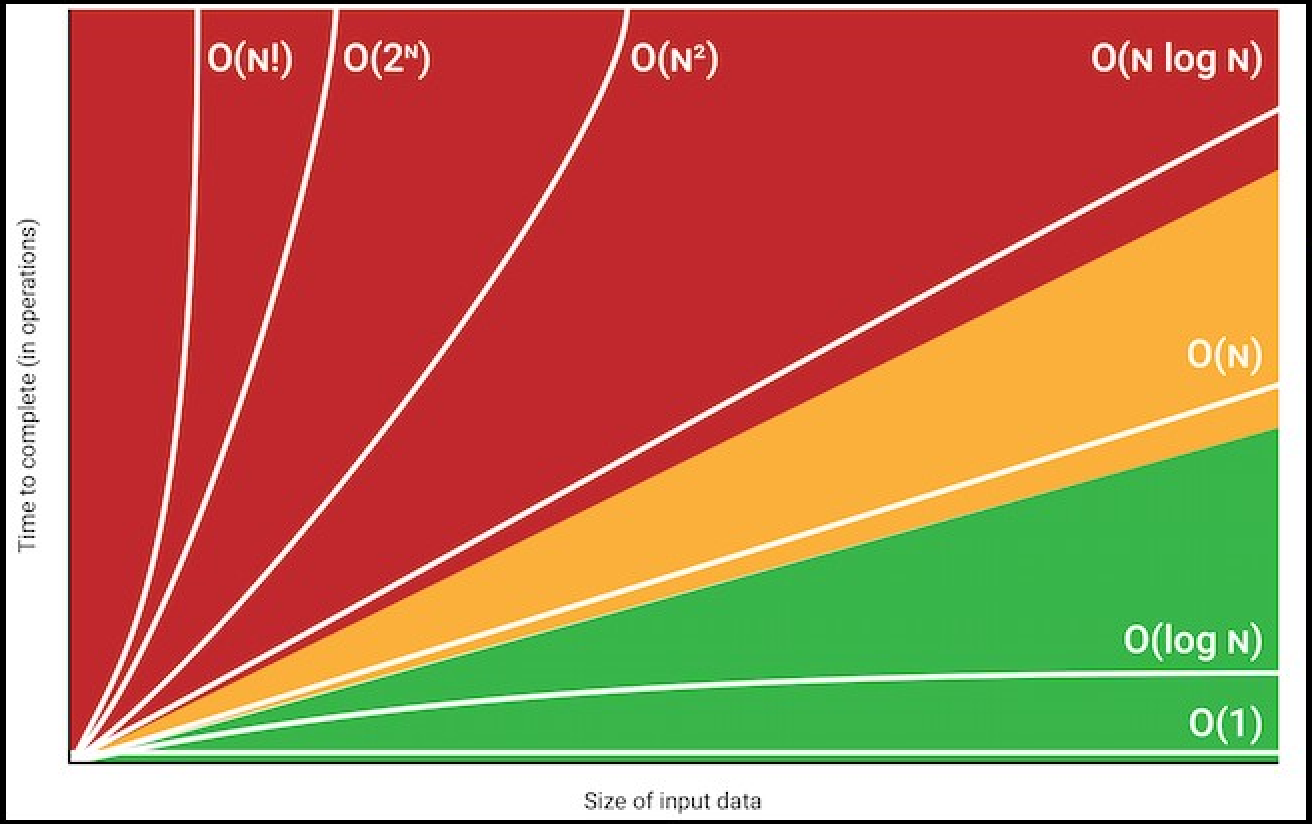


Asymptotic Notation

for i = 1 to n ← $\boxed{n+1}$

· Analysis

· Introduction

time Complexity · Step Count

Theoritical : Priori Analysis

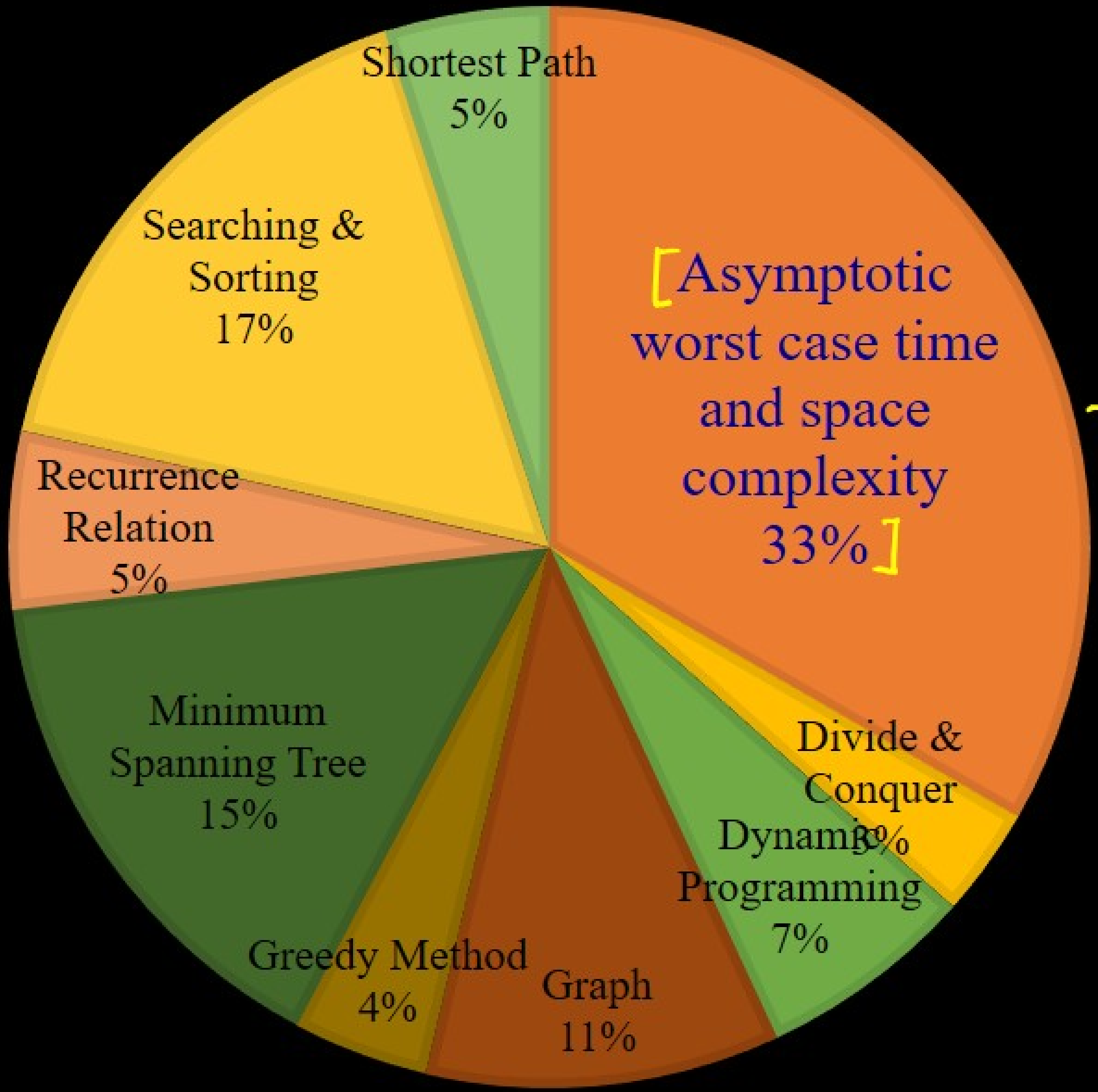
Posteriori Analysis -

RAM model -

Steps ①

1. Logical / Arithmetic a = b+c,
   - 1 steps
2. Loops & Subroutine - multiple count
3. Memory Access - 1 step count

Asymptotic Notation



RAM

- Running time of program is function of Input size

+ Data structure
+
operation

Asymptotic Notation

program/Algorithm
+
functions

n^2
n^3
$\log n$
2^n

- Asymptotic Notation is a "mathematical tool" that allows to find Running time of an algorithm as function of Input size.
- Asymptotic Notation also helps us in bounding the Running time of Algorithm -
  - Lower Bound
  - upper Bound
  - Exact Bound

Asymptotic Notation

Asymptotic Notation is *mathematical tool* allow us to analyze an algorithm's running time by identifying its behavior as the *input size* for the algorithm increases. This allows us to bound the running time of algorithm

Demonstration of function

https://www.desmos.com/calculator

$2^n, x^2, x^3, x, \log x \quad e^x$

About comparison of function (mathematical function)

function may Represent Running time of the program

Asymptotic Analysis

Smaller value of n

Focus on large $n \to \infty$

Behavior of the function

Compare the function

Asymptotically Large value

$\{0 - 1\}$

$n \to \infty$

Asymptotic Analysis

Focus on large $n \to \infty$

- Asymptotic analysis means that we consider trends for large values of n. Thus, the fastest growing function of n is the only one that needs to be considered.

Asymptotic Analysis $\boxed{n \to \infty}$

n	$F(n) = 4n + 4$
10	44
1000	4004
1000000	4000004
1000000000	4000000004 +4

Step count $4n + 4$

Bigger function - start dominating

$4n + 4$

$n \to \infty$

Ignore

$4n$

Asymptotic Analysis

$n \to \infty$

Example: 2

$2n^2 + 2n + 2 - 2n^2$

$n \to \infty$

$4n^2 + 4n + 4$

$4n^2$

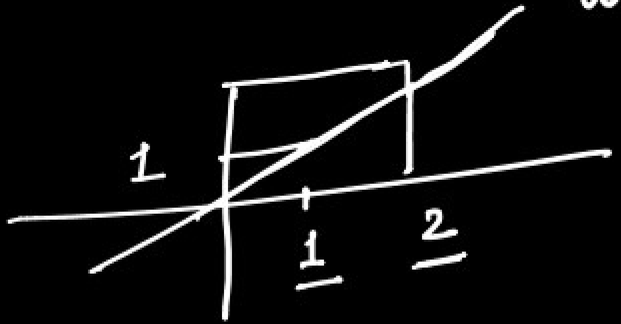
n	$F(n) = 4n^2 + 4n + 4$
1000000	4000000000000+4000000+4

$n \to \infty$

$\{4n^2 + 4n + 4\}$

$4n^2$

Monotonically Increasing Function



we have a function $f(x)$

x_1 x_2 such $x_1 > x_2 \Rightarrow f(x_1) > f(x_2)$

$f(x) = x^2$, $x_1 = 2$, $x_2 = 3$

$x_2 > x_1$

$f(x_2) > f(x_1)$

$9 > 4$

Monotonically Increasing Function

- Running time of a program is a monotonically increasing function is one that increases as x does for all real x.

$$x_1 > x_2 \rightarrow f(x_1) > f(x_2)$$

GATE 2020, Question Number 1

Consider the functions.

I. e^{-x} x 2.7

2^{-x}

$1/2^x$ ←

$x=2$ $1/2^2 = 1/4 = 0.25$

$x=3$ $1/2^3 = 1/8 = 0.125$

II. $x^2 - \sin x$

III. $\sqrt{x^3+1}$ – $\sqrt{x^3+1}$

Which of the above function is/are increasing everywhere in ?

Sinx

Increasing

(A) III only [A]

x^2+1

$x^2 - \sin x$

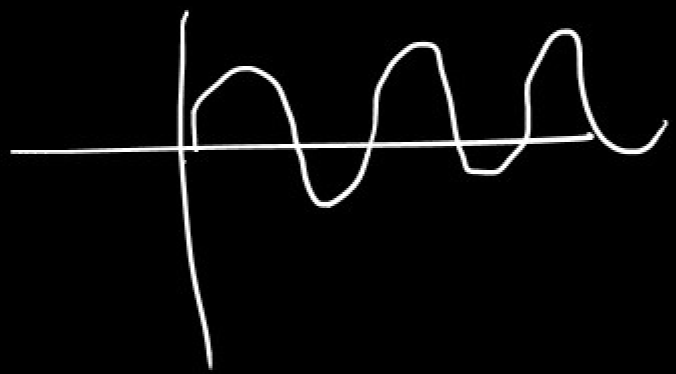
$(-1, 1 \quad -1)$ function

(B) II only

x^2-1

(C) II and III only

(D) I and III only



Trichotomy Properties

two No.

a_1, a_2

- Ordering Properties

$a < b, a = b, a > b$

Asymptotic

function

Notation

$f_1(x)$, $f_2(x)$ equal

$a_1 > a_2$

$a_2 > a_1$

$a_1 = a_2$

$f_1(x) > f_2(x)$

$f_2(x) > f_1(x)$

[Big "oh"] *O Notation*

'O'

$\le$

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

$$f(n) \le c \cdot g(n)$$

$(0-1)$

$n > n_0$, $n > 1$

$c > 0$ is a postive constant

O Notation

$\leq c \cdot$

- [Big "oh"] The function $f(n)$ $\cancel{= O}(g(n))$ (read as f of n is big oh of g of n) iff (if and only if) there exist positive constants $c > 0$ and n_0 such that

$$f(n) \leq c \times g(n) \; for\ all\ n,\ n \geq n_0$$

$f(n)$ lesser or equal

Example

$f(n) = 3n + 2$, and $g(n)$ is n Can we say $f(n) = O(g(n))$

	$3n+2$		$\boxed{4}.n$	
$n=1$	5	$\leq$	4 ✗	$\boxed{c=4}$
$n=2$	8	$\leq$	8 ✓	
$n=3$	11	$\leq$	12 ✓	
$n=4$	14	$\leq$	16 ✓	

$c=4$, $n_0=1$

$n>1$

Example

	$f(n) = 3n + 2$	$g(n) = \mathbf{c}\,n$
$\boldsymbol{n}$**=1**		
$\boldsymbol{n}$**=2**		
$\boldsymbol{n}$**=3**		
$\boldsymbol{n}$**=4**		

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

Example

	$f(n) = 3n + 2$		$g(n) = 4n$
n=1	5		4
n=2	8	$\leq$	8
n=3	11	$\leq$	12
n=4	14	$\leq$	16

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

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

One c, n_0 $c > 0$, $n > n_0$

c, n_0 positive

$c = 4$ $n_0 = 1$

$c = 5$, $n > 0$

Example

$f(n) \leq c.g(n)$

$f(n) = 3n + 2$, and $g(n)$ is n Can we say $g(n) = O(f(n))$

c_1 $c = 1$, $n > 0$

$n = 0$ $n \leq 3n+2$

$0 \leq 2$

$n = 1$ $1 \leq 5$

$n = 2$ $2 \leq 8$

$n = 0$

$n \geq 0$

Example

	$f(n) = 3n + 2$	$g(n) = n$
n=1		
n=2		
n=3		
n=4		

Example

	$f(n) = 3n + 2$	$g(n) = n$
n=**1**	5	1
n=**2**	8	2
n=**3**	11	3
n=**4**	14	4

$\boxed{n > 0}$

Example

$f(n)$ $3n+2$ $g(n)=n$

$f(n) = n^2$, and $g(n)$ is 2^n

	$f(n) = n^2$	$g(n) = 2^n$	
n=1	1	2	
n=2	4	4	
n=3	9	8	
n=4	16	16	

Example

$f(n) = n^2$, and $g(n)$ is 2^n

	$f(n) = n^2$	$g(n) = 2^n$	
n=**1**	1 $\leq$	2	
n=**2**	4 $\leq$	4	
n=**3**	9 $\geq$	8	
n=**4**	16 $\leq$	16	

$\leftarrow$ $\underline{n_0}$

Example

$f(n) = n^2$, and $g(n)$ is 2^n

	$f(n) = n^2$	$g(n) = 2^n$	
n=5	25 $<$	32	
n=6	36 $<$	64	
n=7	49 $<$	128	
n=8	64 $<$	256	

$f(n) \cancel{\in} O(g(n))$

$C=1$. ~~$n>4$~~ $n>3$

No Compare the function

$8. < 108$

$f_1(n) < f_2(n)$

Example

$$100n^{1} + 6 = \overset{102}{O(n^{1})}$$

$$\left\{ \begin{array}{l} c = 101 \\ n_0 = 5 \end{array} \right\} \; n > 5$$

100n + 6 — 101n

As $100n + 6 \leq 101n$ *for all* $n \geq 6.$

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

$f(n) \leq c \cdot g(n)$

equal

$n=1$	106		101
$n=2$	206		202
$n=5$	506		505
$n=6$	606	$\leq$	606
$n=7$	706	$\leq$	707

Example

$f(n) \le c \cdot g(n)$

$C = 20$

$\boxed{n_0}$

$10n^2 + 4n + 2 = O(n^2)$ yes

$10n^2 + 4n + 2 \le 10n^2 +$

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

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

$as\ 10n^2 + 4n + 2 \le 11n^2\ for\ all\ n > 5.$

$10n^2 + 4n + 2 \qquad 11n^2$

$n = 1 \qquad 16 \qquad 11$

$n = 2 \qquad 50 \qquad 44$

$n = 3 \qquad 104 \qquad 11 \times 9 = 99$

$n = 4 \qquad 178 \qquad 11 \times 16 = 176$

$272 \qquad 275$ ✓

$n = 5$

Example

$6 * 2^n + n^2 = O(2^n)$

Example

$6 * 2^n + n^2 = O(2^n)$

$As\ 6 * 2^n + n^2 < 7 * 2^n for\ n > 4.$

Example

$n \le n^2 \le n^3 \le n^4 \ldots$

$3n + 3 = O(n^2)$

Whatever big constant c can be taken

n^2 ~~lesser~~ function than n

no

$f(n) \le c \cdot g(n)$

no

n^2 is $O(\boxed{c}\ n)$

$n^2 \le 100\,n$

$n=1$ $\quad 1 \le 100 \quad 1000$

$n=10$ $\quad 100 \le 1000$

$n=100$ $\quad 100 \times 100 \le 100 \times 100$

$n=101$ $\quad 101 \times 101 \le 100 \times 100$ ✗

Example

$3n + 3 = O(n^2)$

$As\ 3n + 3 < 3n^2 for\ n \geq 2.$

Example

$10n^2 + 4n + 2 \neq O(n).$

O Notation

- $3n + 3 = O(n)$ as $3n + 3 < 4n$ for all $n > 3$.
- $100n + 6 = O(n)$ as $100n + 6 \leq 101n$ for all $n \geq 6$.
- $10n^2+4n+2 = O(n^2)$ as $10n^2+4n+2 \leq 11n^2$ for all $n > 5$.
- $6*2^n + n^2 = O(2^n)$ as.
- $3n+3 = O(n^2)$ as

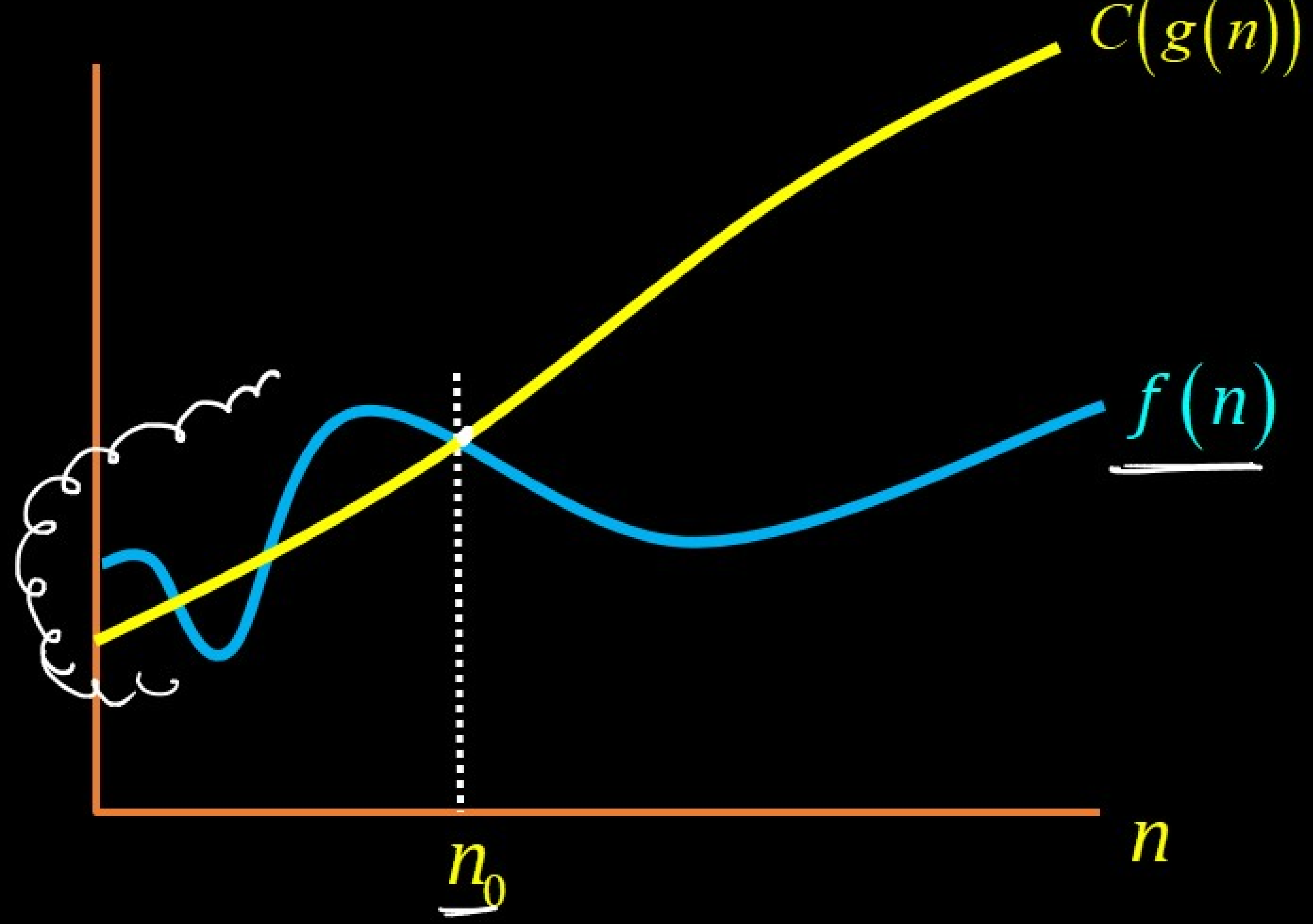
O Notation

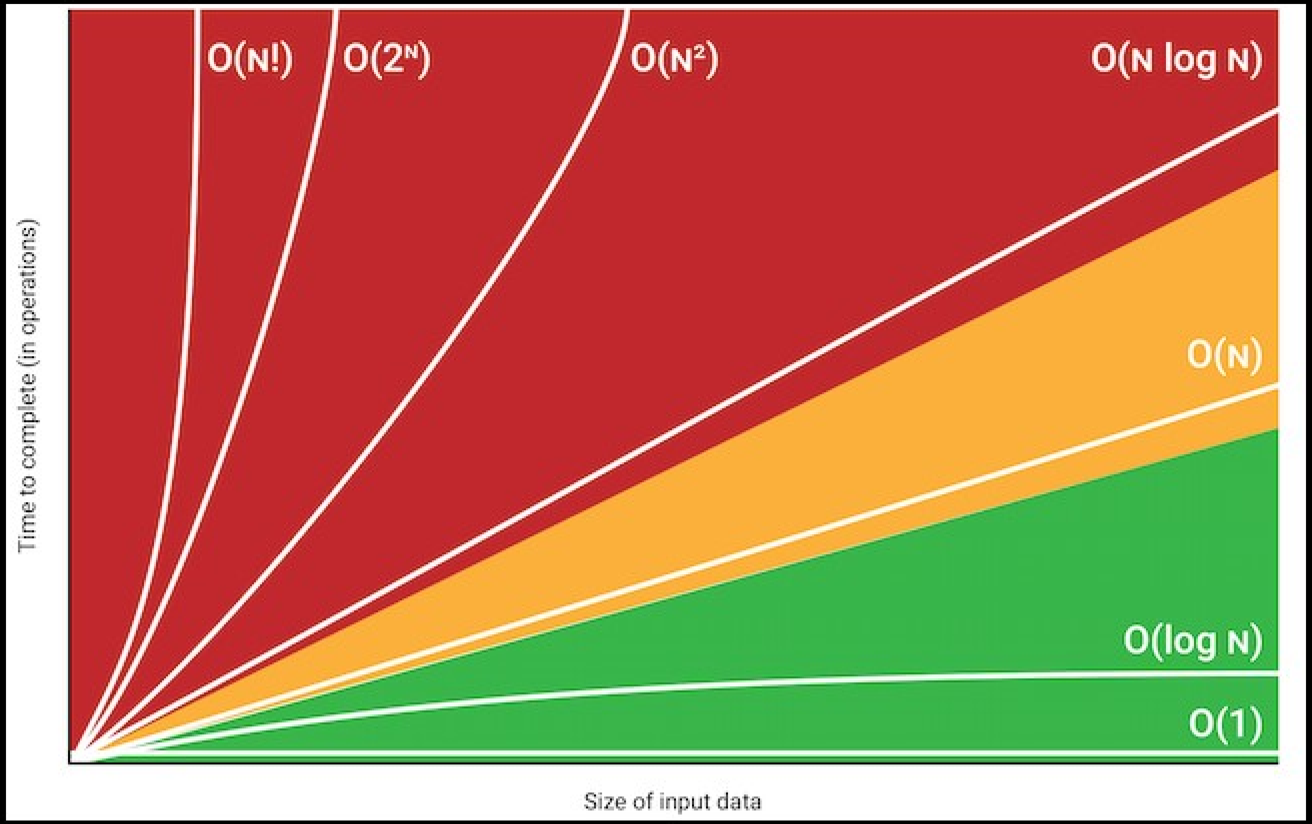
- $3n + 3 = O(n)$ as $3n + 3 < 4n$ for all $n > 3$.
- $100n + 6 = O(n)$ as $100n + 6 \leq 101n$ for all $n \geq 6$.
- $10n^2+4n+2 = O(n^2)$ as $10n^2+4n+2 \leq 11n^2$ for all $n > 5$.
- $1000n^2 + 100n - 6 = O(n^2)$ as $1000n^2 + 100n - 6 < 1001n^2$ for $n \geq 100$.
- $6*2^n + n^2 = O(2^n)$ as $6*2^n + n^2 < 7*2^n$ for $n > 4$.
- $3n+3 = O(n^2)$ as $3n + 3 < 3n^2$ for $n \geq 2$.

Graphical Representation $\left[f(n) = O(g(n))\right]$

$f(n) \leq c \cdot g(n)$

$n \to \infty$





Asymptotic Notation

Can I take bigger value of c and make n greater function than n^2?

Natural ordering $\quad n < n^2 < n^3 < n^4$

Example

$F(n) = 10n^2+2$ and $g(n)$ is n Can I say $f(n) = O(g(n))$

For $c = 1000$ ✓

Example

	$10n^2+2$	$1000n$
n=1		
n=10		
N=99		
n=100		
n=101		

Example

	$10n^2+2$	1000n
n=1	12	1000
n=10	1002	10000
N=99	98,012	99000
n=100	100002	100000
n=101	102,012	101,000

Control c

$2n^2$ $O(n^2)$

n_0

Example

https://www.desmos.com/calculator

Example

$F(n) = 10n^2 + 2$ *and* $g(n)$ *is* n

Can I say $f(n) = O(g(n))$

For $c = 1000$

	$10n^2 + 2$	$1000\ n$
n =1	12	1000
n=10		
n=99		
n=100		
n=101		

Example

$n^2 + 2n$ $\qquad n^2$

n^2 $\qquad n^2$

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

$n \to \infty$

$f(n)$ – Mathamatically equal funtion

– $f(n)$ Mathemathically lesser function

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

- $f(n)$ will be equal function
- $f(n)$ Mathematically smaller function

n is $O(n^2)$ ✓ Asymtotic

n is $O(n)$ ✓

$\leq$

Practical Meaning

Many upper Bound : tightest upper

O - worst case time
- upper bound

1. Linear Search is $O(n)$ —

- time complexity of Linear Search is $O(n)$

$$T(n) \leq c.n$$

$T(n)$

- n is upper bound for Running time of Linear Search

- O - Notation Bounds the Running from above

Practical Meaning

The Big O notation defines an *upper bound* of an algorithm, it bounds a function only from above.

tightest upper bound
is unique

Tight Bound

Mute Stop Video Participants Chat Q&A Polls New Share Pause Share Annotate More

You are screen sharing Stop Share

* 1. Linear Search (time Complexity) is $O(n^2)$ true

$\leq$

True/False

for i=1 to n → 2n+1

if (A[i] == x) → n

return i

retur 0; ← 1

~~else~~

$n+1+n+1$

$2n+2$

① $2n+2$ is $O(n)$

$2n+2$ is $\leq c_1 n$

Ⅱ $2n+2 \leq c_1 n^2$ false

tightest upper

only upperbound unique ✗

$1<2<3<4$ $n<n^2$ $n<n^3$ $n<n^4$

ACE Live Class 1

Comparison of The Function

- *The bigger function will dominate*

Asymptotic

$10n^2 + 4n + 4$ $\quad n \to \infty$ - n^2

$f(n) = 1\cancel{00}\,n$

$g(n) = \cancel{c1} \cdot n$

$C100$

$C = 1000000$

$10n^2$ $\quad n^2$

n^2 $\quad n^2$

Ans.

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

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

Comparison of The Function

- *Ignore constant factors:*

$100n \quad n$

$100n \quad \underline{c} - 101$

Constant value

can nullify using $\underline{c}$

$100n \leq 101n$ ✓

. Never compare the function based on constant.

Comparison of The Function

- *The bigger function will dominate*
  - $n^2 + n$ will be treated as n^2
- *Ignore constant factors:*
  - Multiplicative constant factors are ignored. For example, $347n\ is\ n$.

Theorem

- If $f(n) = a_m n^m + \ldots\ldots. + a_1 n + a_0$, $then\ f(n) =$

$f(n)$ is $O(n^m)$

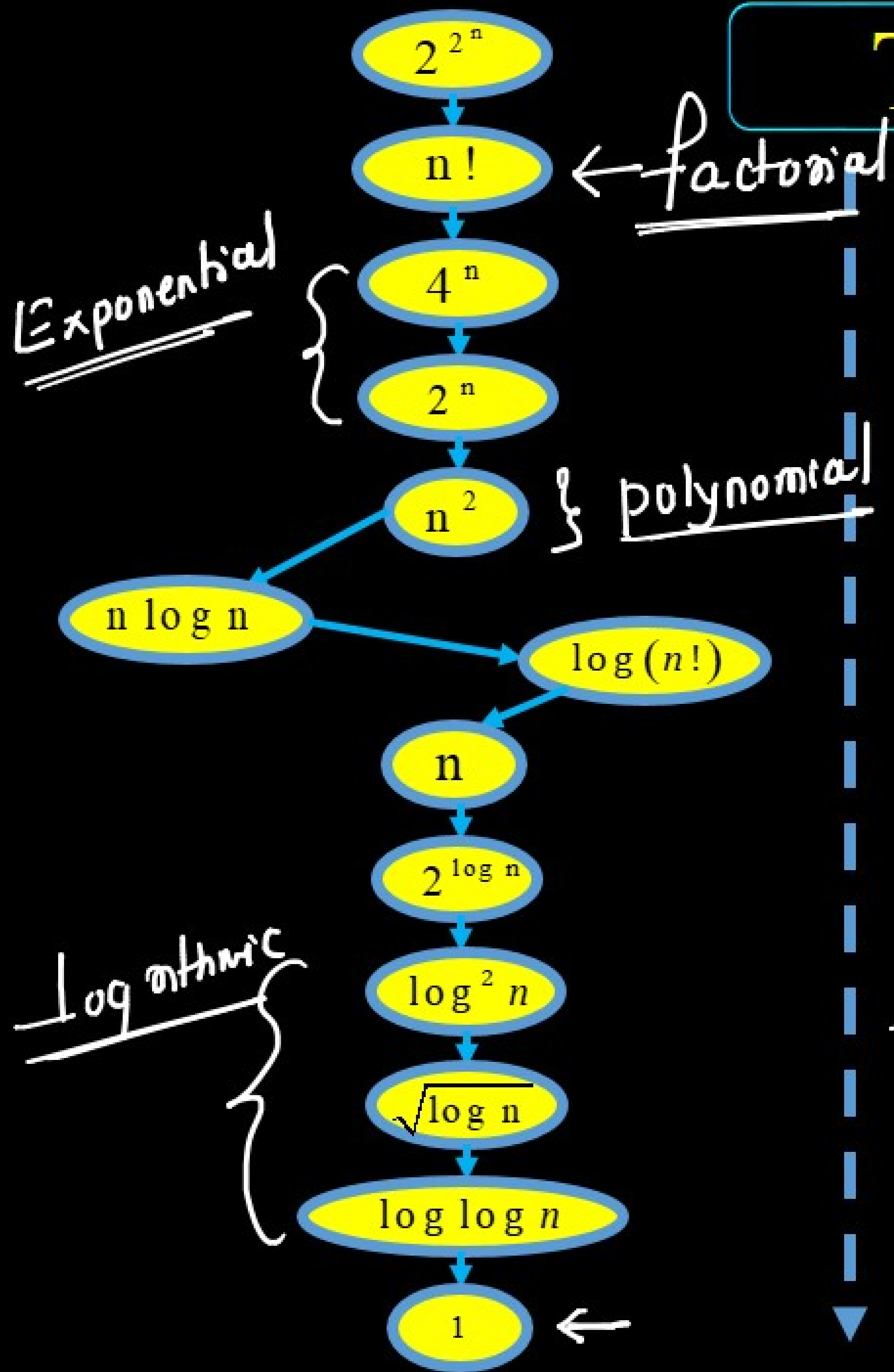
Theorem

- If $f(n) = a_m n^m + \ \dots\dots. \ + \ a_1 n \ + \ a_0, \ then\, f(n) = \ O(n^m)$.

Comparison of The Function

Note: Constant factors appearing exponents cannot be ignored. For example, 2^{3n} is not $O(2^n)$.

The Growth of a Function



1 — O(1) — Not 1 operation

O(1) — Constant No. of operation

for (i=0; i<100; i++) ← 1, 101, 100

— a++ ← 100

0 — 99

100

for (i=0; i<n; i++) — depends upon n

100 times } constant value

1000 time } Natural order

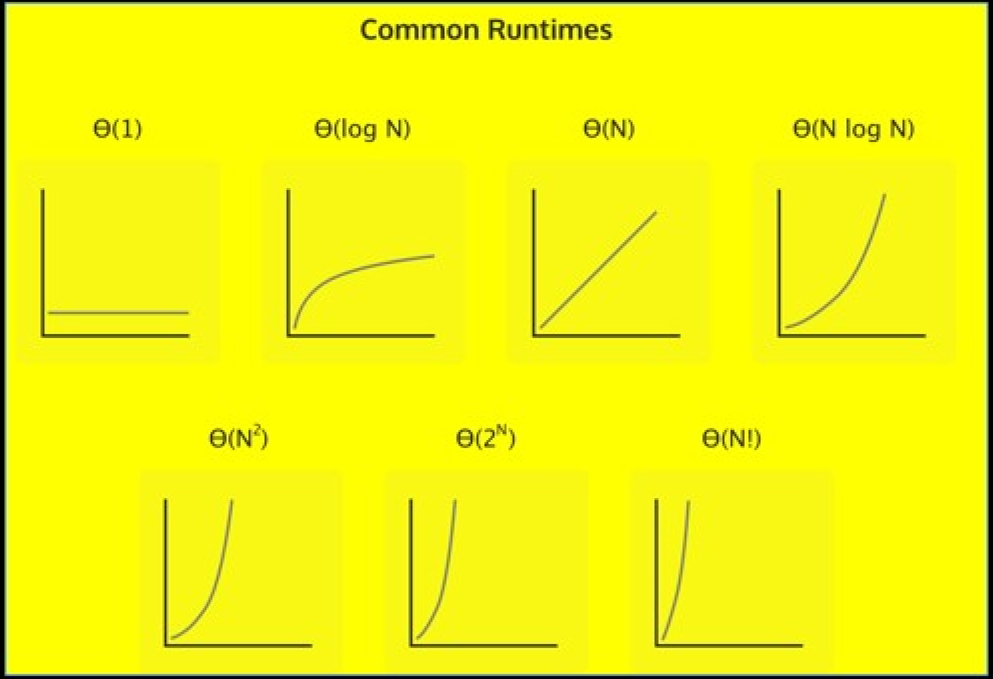
~~too~~ constant < Logarithmic < polynomial < exponential

n \ f(n)	logn	n	n logn	n^2	2^n	n!
10	0.003 μs	0.01 μs	0.033 μs	0.1 μs	1 μs	3.63 ms
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	1 ms	77.1 years
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	1 sec	8.4×10^{15} yrs
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	18.3 min	8.4×10^{15} year
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	13 days	

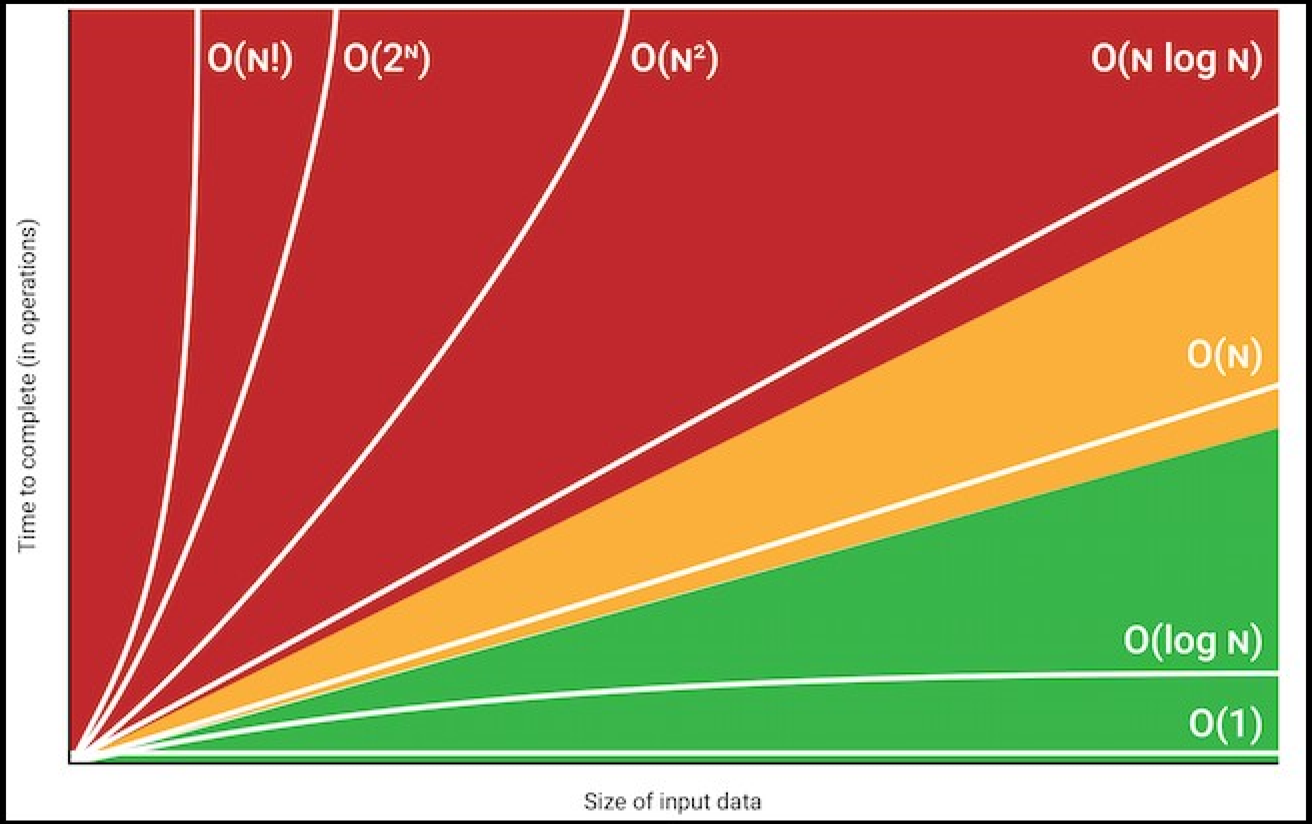
40! – within L30

30! / Execute programming

30! steps



Asymptotic Notation



Asymptotic Notation

Ω Notation

upperbound $\leq$ O Notation

Lowerbound $\geq$ Ω

Θ

o

ω

$f(n)$ is $\Omega(g(n))$ iff c, n_0

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

Ω *Notation*

- [Omega] The function $f(n) = \Omega(g(n))$ (read as f of n is omega of g of n) iff there exist positive constants c and n_0 such that

$$f(n) \geq c \times g(n) \; for \; all \; n, n \geq n_0$$

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

drop

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

$\geq$

$3n+2$

$f(n)$

$g(n) - n$

- $f(n)$ will be *equal* function
- $f(n)$ Mathematically *bigger* function

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

$3n+2 \geq n$

$f(n) - n^2$ $\quad g(n) = n$

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

$n^2 \geq n$

Ω Notation

- $3n + 3 = \omega(n)$ as $3n + 3 > 3n$ for $n \geq 1$.
- $100n+6 = \omega(n)$ as $100n + 6 > 100n$ for $n > 1$.
- $10n^2 + 4n + 2 = \omega(n^2)$ as $10n^2 + 4n + 2 > n^2$ for $n > 1$.
- $6 * 2^n + n^2 = \omega(2^n)$ as $6 * 2^n + n^2 > 2^n$ for $n > 1$.
- Observe also that $3n + 3 = \omega(1)$,
- $10n^2 + 4n + 2 = \omega(n)$, — true
- $10n^2 + 4n + 2 = \omega(1)$, — true
- $6 * 2^n + n^2 = \omega(n^{100})$,

$3n+2$ $\quad f(n)$ is $\Omega\, g(n)$

$\geq$

Constant function

$3n+3$ is $\Omega(1)$ — true

$3n+2$ is $\Omega(n^2)$ — false

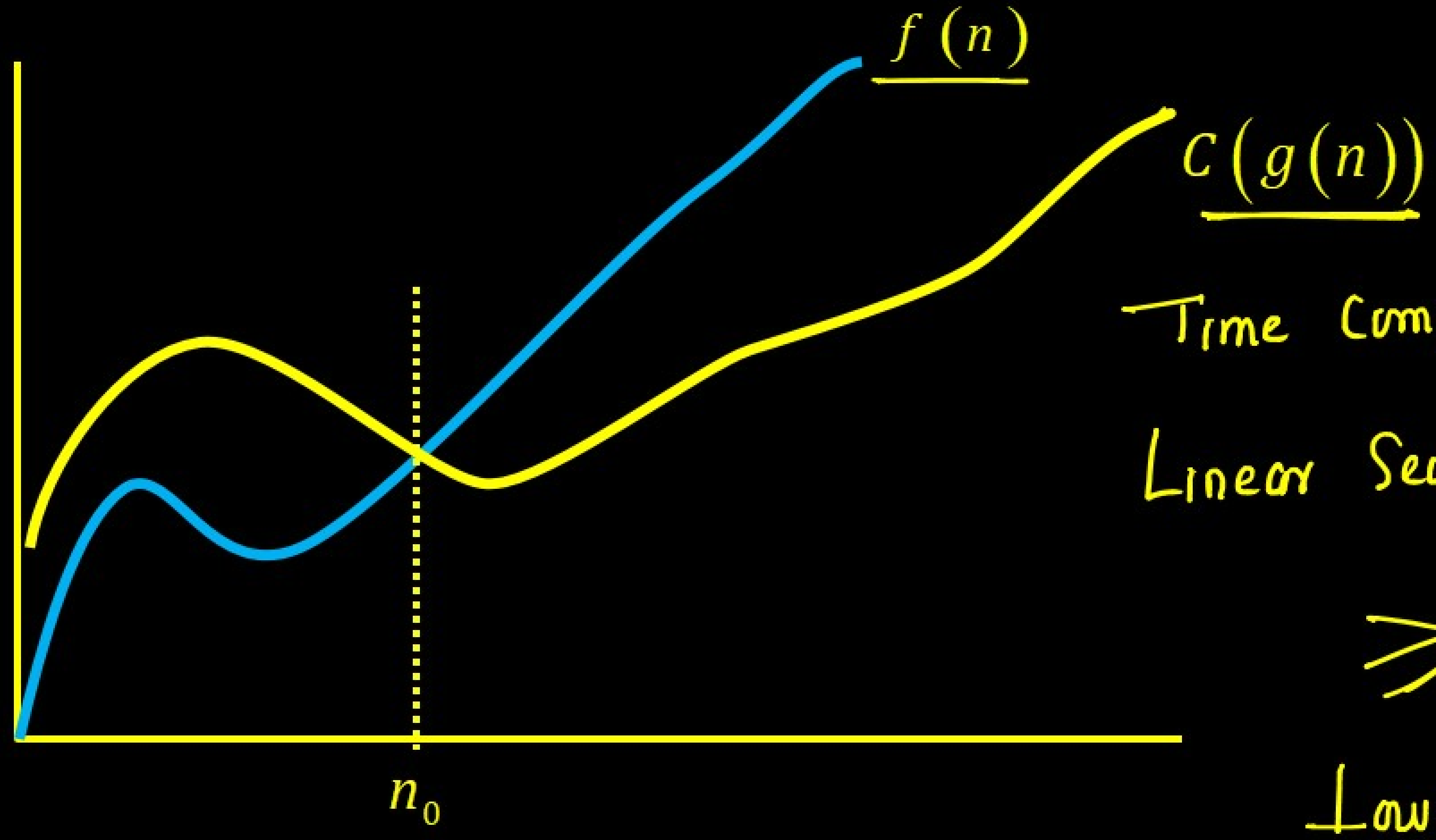
Graphical Representation $f(n) = \Omega(g(n))$

$C > 0$

Fraction

$O(n)$

Run



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

Time complexity of Linear Search $\Omega(1)$ – Tw

$\geq$ Constant

Lower Bound / Best case time complexity

Practical Meaning

Practical Meaning

We say that the running time is "big-Ω of f(n)." We use big-Ω notation for *asymptotic lower bounds*, since it bounds the growth of the running time from below for large enough input sizes.

O - upper bound

Ω Lower bound

Best case

Ω

Difference between Big O and Big Ω

Difference between Big O and Big Ω

The difference between Big O notation and Big Ω notation is that Big O is used to describe the *worst case running time for an algorithm*. But, Big Ω notation, on the other hand, is used to describe the *best case running time for a given algorithm*.

Θ Notation

O - Notation
×

Exact bound :

Best case = worst
= Average

upperbound = Lower bound

Add (A, B, C, n, n) { $2n^2+2n+1$

for i = 1 to n {   n+1

for j = 1 to n {   n x (n+1)

C[i,j] = a[i,j] + b[i,j]

} n^2

}

Θ Notation

$c_1 g(n) \leq f(n) \leq c_2 g(n)$ *for all* $n, n \geq n0$

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

$f(n)$ and $g(n)$

equal funtion

$f(n)$ $\quad$ $g(n)$

$2n+2 \geq n$

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

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

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

$f(n)$ - equal / lesser

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

$f(n)$ equal / greater

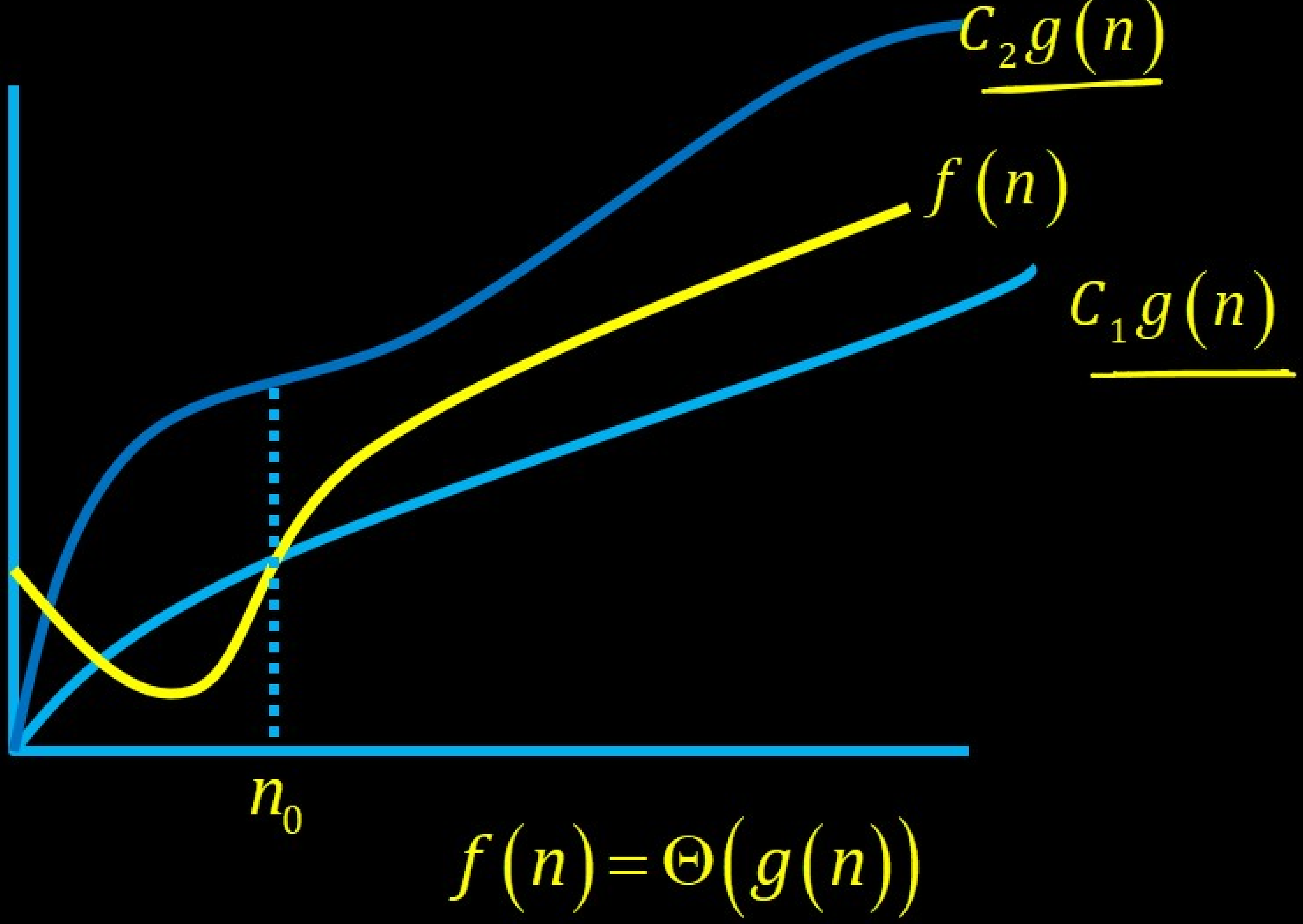
$2n+2 \leq 100n$

Θ Notation: Practical Meaning

Θ Notation: Practical Meaning

- The theta notation bounds a *functions* from above and below, so it defines *exact asymptotic behavior*.
- A simple way to get Theta notation of an expression is to drop low order terms and ignore leading constants.

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



Θ Notation

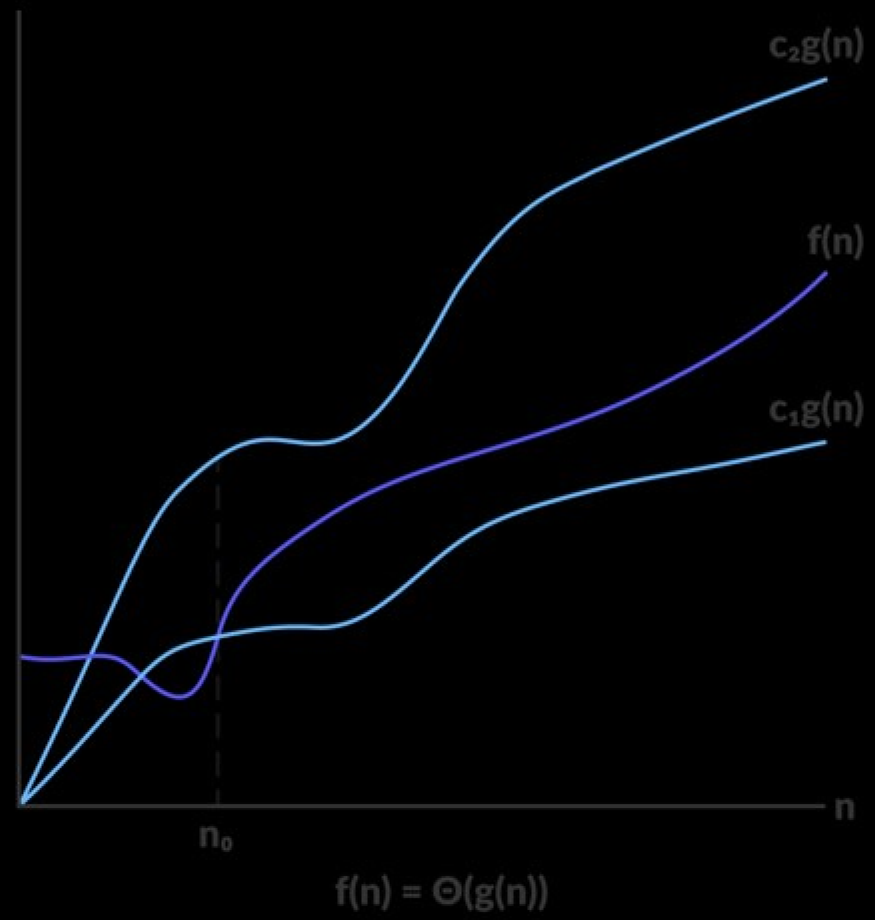
- Example 1.13 The function $3n + 2 = \Theta(n)$ as $3n + 2 \geq 3n$ for all $n > 2$ and $3n + 2 < 4n$ for all $n \geq 2$,

$\Theta(n)$

- so $c1 = 3$, $c2 = 4$, and $n0 = 2$.
- $10n^2 + 4n + 2 = \Theta(n^2)$,
- $6 * 2^n + n^2 = \Theta(2^n)$, and
- $10 * \log n + 4 = \Theta(\log n)$.
- $3n + 2 \neq \Theta(1)$,
- $3n + 3 \neq \Theta(n^3)$,
- $10n^2 + 4n + 2 \neq \Theta(n)$,

$10n^2 + 4n + 2$ is $\Theta(1000 n^2)$

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



GATE 2015 Set-III | 1-Mark Question

Consider the equality $\sum_{i=0}^{n} i^3 = X$ and the following choices for X

$0^3 + 1^3 + 2^3 + \cdots + n^3$

$\left[\frac{n(n+1)}{2}\right]^2$

$\frac{1}{4}\left[n^2(n^2+1+2n)\right]$

$\frac{1}{4}\left[n^4 + n^2 + 2n^3\right]$

n^4

O ≤

Ω ≥

Θ =

I. $\theta(n^4)$ — yes

II. $\boxed{\theta(n^5)}$ | No

III. $O(n^5)$ — yes

IV. $\Omega(n^3)$ — yes

n^4 is $\Omega(n^3)$

$\boxed{n^4 \geq c_1 n^3}$

The equality above remains correct if X is replaced by

(A) only I — A ✗

(B) Only II

(C) I or III or IV but not II — [C]

(D) II or III or IV but not I

n^4

n^4 is $\Theta(n^4)$

n^4 is $\Theta(n^5)$

n^4 is $O(n^5)$

$\boxed{n^4} \leq c_1 n^5$