Asymptotic Notations

Asymptotic Notations give us an idea about how good a given algorithm is compared to some other algorithm
Let us see the mathematical defination of "Order of" now

Primarily there are three types of widely used asymptotic notations.

(1) Big Oh notation (O)
(2) Big Omega notation (Ω)
(3) Big theta notation (Θ) $\longrightarrow$ widely used One!

Big oh notation

Big oh notation is used to describe asymptotic upper bound
Mathematically, if $f(x)$ describes running time of an algorithm; $f(x)$ is $O(g(x))$ iff there exist positive constants c and n_0 such that

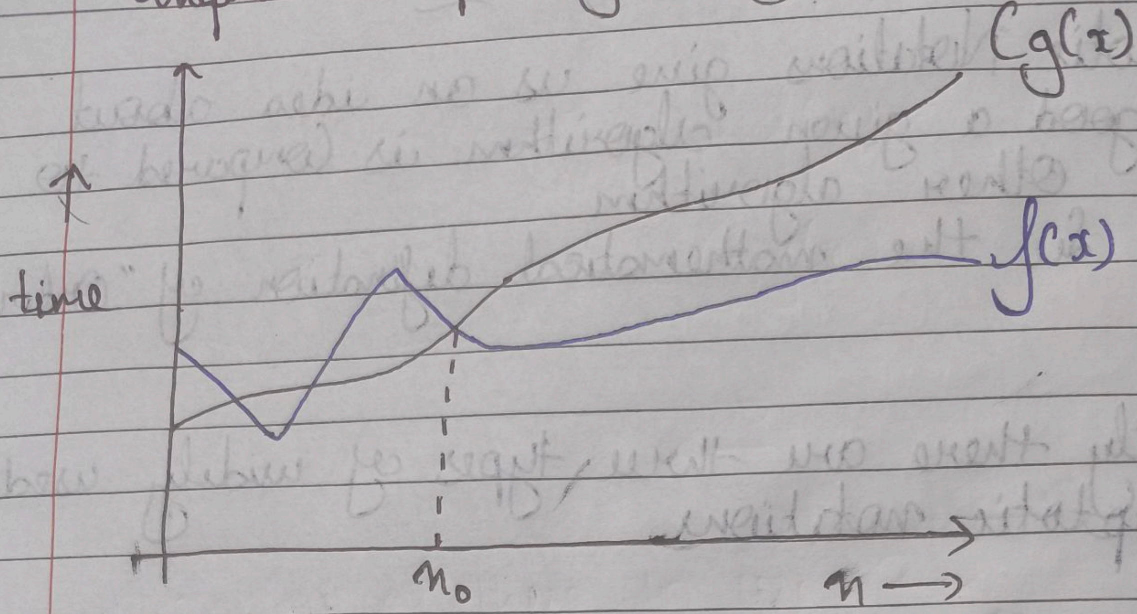
$$0 \leq f(x) \leq c\,g(x) \text{ for all } n \geq n_0$$

$\Downarrow$

used to give upper bound on a funᶜ

if a function is $O(n)$, it is automatically $O(n^2)$ as well!

Graphic Example for Big oh (O)



Big Omega notation

Just like O notation provides an asymptotic upper bound, Ω notation provides asymptotic lower bound. Let f(n) define running time of an algorithm;

f(n) is said to be Ω g(n) if there exists positive constants C and n_0 such that

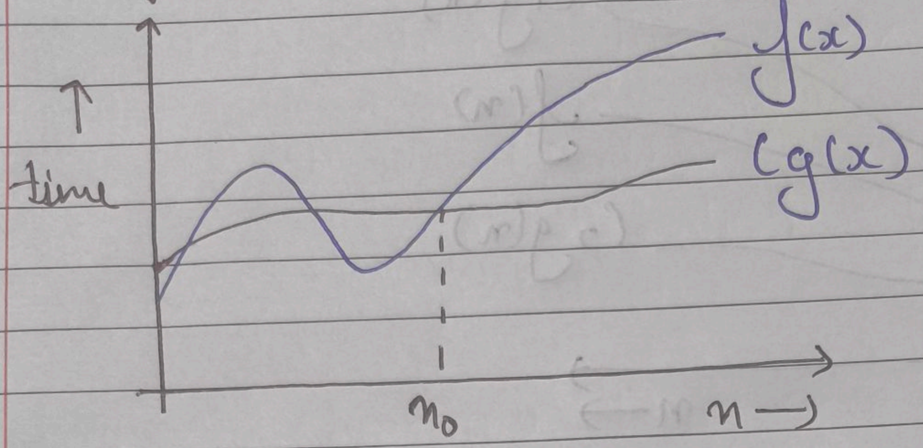
$0 \leq Cg(n) \leq f(n)$ for all $n \geq n_0$

⇓

used to give lower bound on a function

if a function is $O(n^2)$ it is automatically $O(n)$ as well

Graphic Example for Big Omega (Ω)



Big theta notation let $f(x)$ define running time of an algorithm

$f(x)$ is said to be $\Theta(g(n))$ iff $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$

Mathematically,

$$0 \le f(n) \le C_1 g(n) \quad \forall\ n \ge n_0$$

Sufficiently large value of n

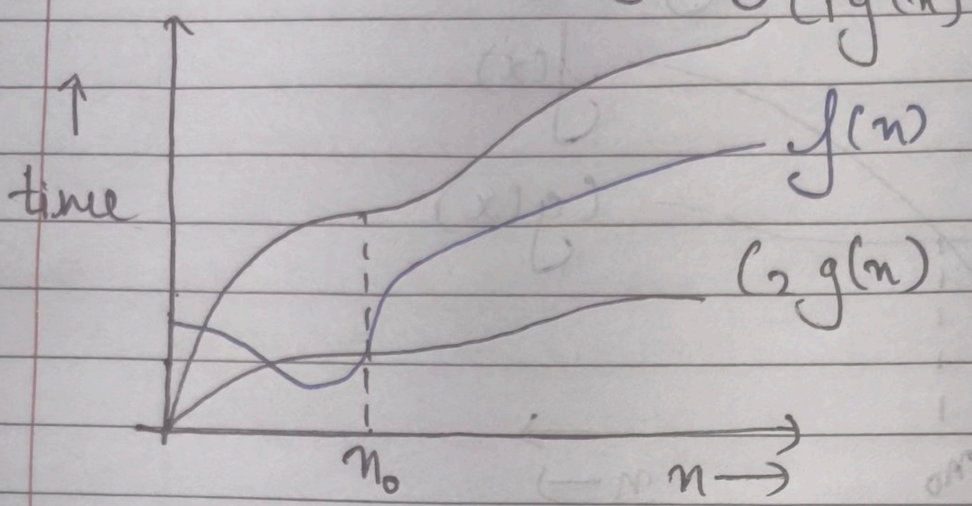
$$0 \le C_2 g(n) \le f(n) \quad \forall\ n \ge n_0$$

Merging both the eq^n, we get

$$0 \le C_2 g(n) \le f(n) \le C_1 g(n) \quad \forall\ n \ge n_0$$

The eq^n simply means there exists positive constants C_1 and C_2 such that $f(n)$ is sandwich between $C_2 g(n)$ and $C_1 g(n)$

Graphic Example of Big theta



Which One of these to use?
Since Big theta gives a better picture of runtime for a given algorithm, most of the interviewers expect you to provide an answer in terms of Big theta when they say "Order of"

Increasing Order of Common runtimes

$1 < \log n < n < n \log n < n^2 < n^3 < 2^n < n^n$

Better

worse

Common runtime from better to worse $\longrightarrow$