Asymptotic Notations

- The efficiency of an algorithm depends on the amount of time, storage and other resources required to execute the algorithm. The efficiency is measured with the help of asymptotic notations.
- An algorithm may not have the same performance for different types of inputs. With the increase in the input size, the performance will change.
- The study of change in performance of the algorithm with the change in the order of the input size is defined as asymptotic analysis.

Why to use Asymptotic notations?

Problem1: Algorithm A complexity is n

Algorithm B complexity is n+c

Algorithm C complexity is n-c

Then how to compare which algorithm is best and balancing the complexity with Asymptotic notations.

another example: $n^2-6n+50$

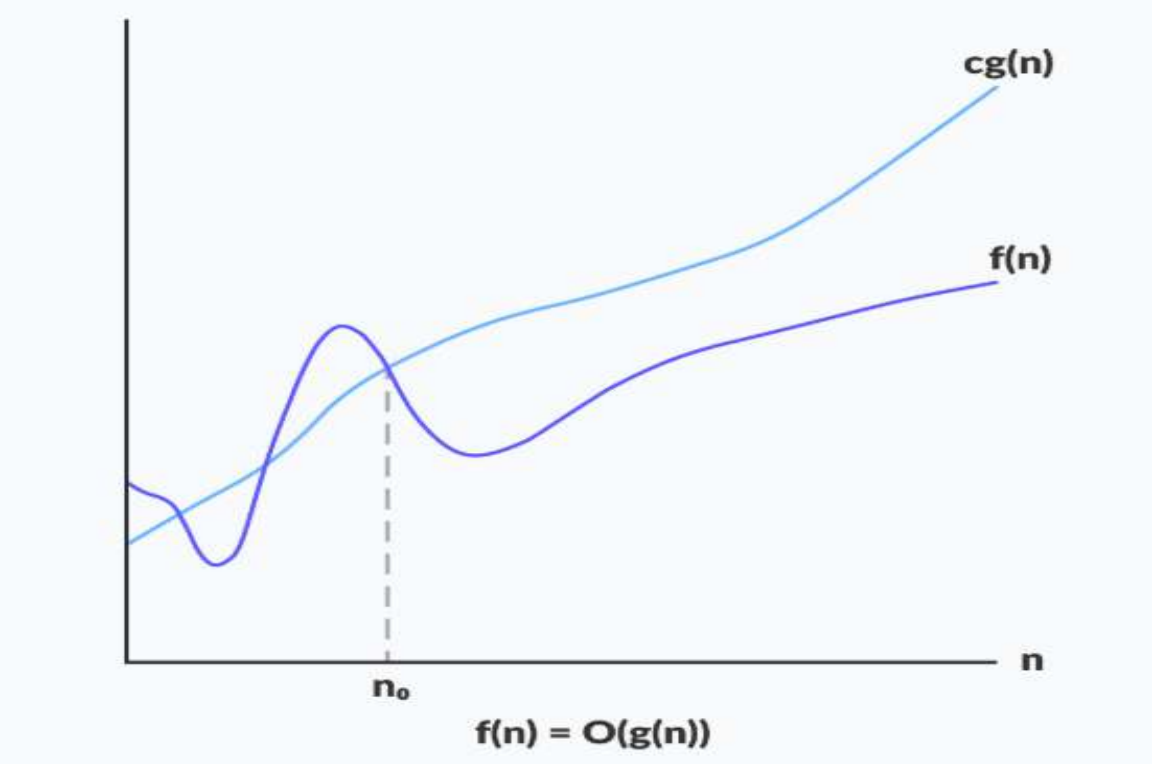
$n^4-3n^3+2n^2+1$

Types of Asymptotic Notations

1. Big O (O)
2. Big Omega (Ω)
3. Big Theta (Θ)

Big O Notation

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



Big O Notation Examples

Example:

$n-6$ in $O(n)$ here $f(n)=n-6$, $g(n)=n$

$n-c$ in $O(n)$

n^3-2n-6 in $O(n^3)$

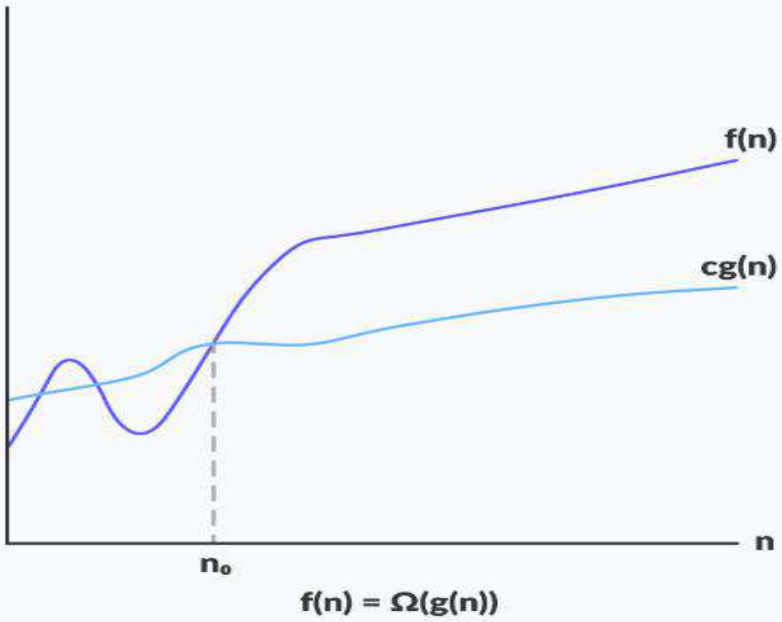
$n^3+6n-16$ in $O(n^3)$

n^3-6 in $O(n^3)$

The biggest advantage of the notation is that complicated expressions can be dramatically simplified.

Big Omega (Ω)

$\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0 \}$



Big Omega (Ω) Notation Examples

Example:

$n-6$ in $\Omega(1)$ here $f(n)=n-6$, $g(n)=1$

$n+c$ in $\Omega(n)$, $n-c$ in $\Omega(1)$

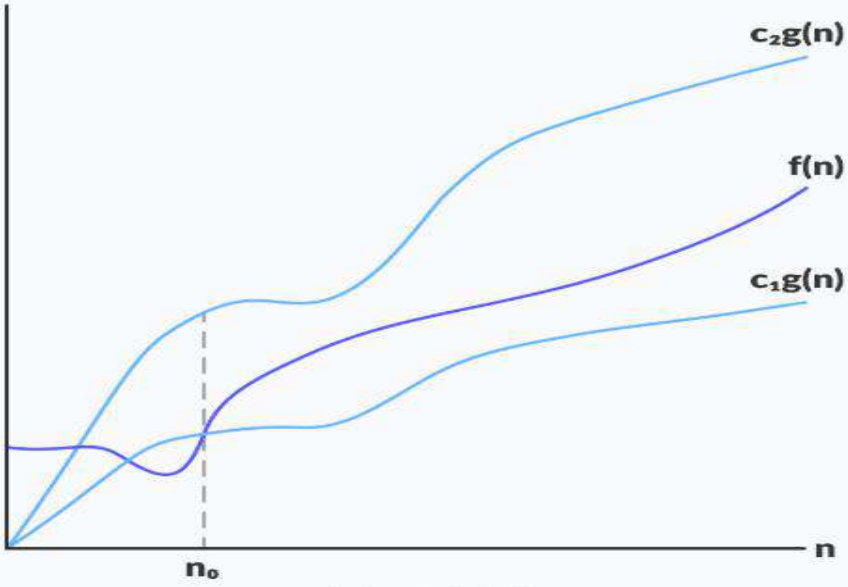
n^3+2n+6 in $\Omega(n^3)$

n^3+6n-1 in $\Omega(n^3)$

n^3+6 in $O(n^3)$

Theta (Θ)

$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0 \}$



Theta (Θ) Notation Examples

Example:

$n-c$ in $\Theta(n)$ here $f(n)=n-c$, $g(n)=n$ and

$c<<<<n$ (c is negligible to n)

n^5+6 in $\Theta(n^5)$

n^3+or-c in $\Theta(n^3)$