Types of Asymptotic Notations in Complexity Analysis of Algorithms: Asymptotic notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis.

There are mainly three asymptotic notations:

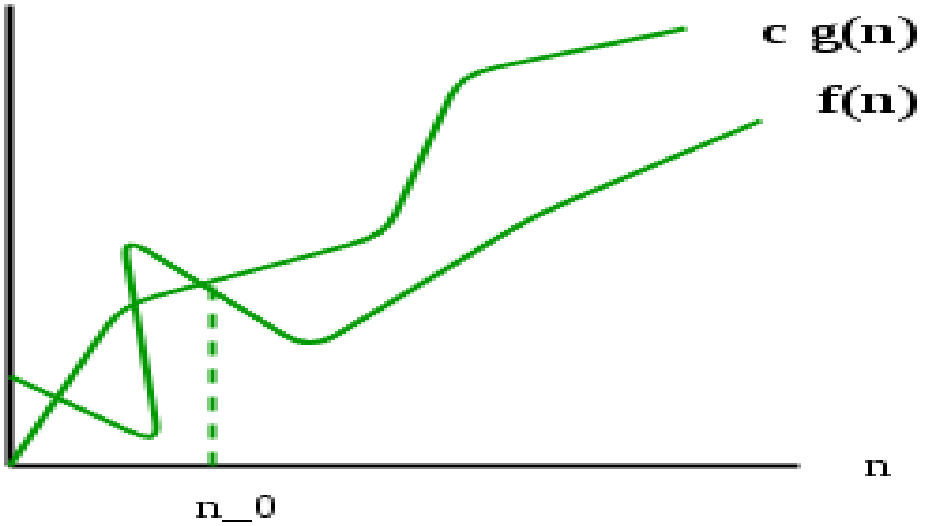
1. ***Big-O Notation (O-notation):*** *upper bound(worst case analysis)*

   Big-O notation represents the upper bound of the running time of an algorithm. Therefore, it gives the worst-case complexity of an algorithm.

   .It is the most widely used notation for Asymptotic analysis.
   .It specifies the upper bound of a function.
   .The maximum time required by an algorithm or the worst-case time complexity.
   .It returns the highest possible output value(big-O) for a given input.
   .Big-Oh(Worst Case) It is defined as the condition that allows an algorithm to complete statement execution in the longest amount of time possible.

If f(n) describes the running time of an algorithm, f(n) is O(g(n)) if there exist a positive constant C and n0 such that, $0 \leq f(n) \leq cg(n)$ for all $n \geq n0$

It returns the highest possible output value (big-O)for a given input. The execution time serves as an upper bound on the algorithm's time complexity.



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

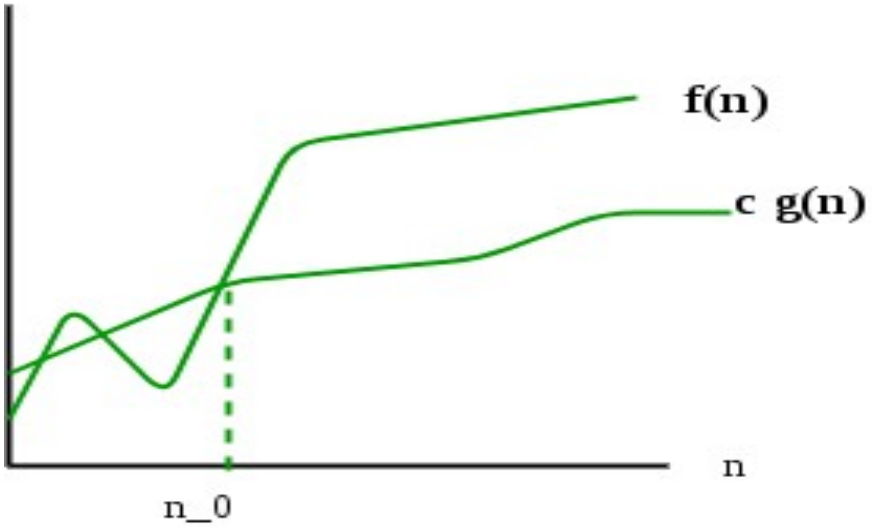
Mathematical Representation of Big-O Notation:

O(g(n)) = { f(n): there exist positive constants c and n0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n0$ }

2. ***Omega Notation (Ω-notation):*** *lower bound(best case analysis)*

Omega notation represents the lower bound of the running time of an algorithm. Thus, it provides the best case complexity of an algorithm.

The execution time serves as a lower bound on the algorithm's time complexity.
It is defined as the condition that allows an algorithm to complete statement execution in the shortest amount of time.
Let g and f be the function from the set of natural numbers to itself. The function f is said to be Ω(g), if there is a constant c > 0 and a natural number n0 such that c*g(n) ≤ f(n) for all n ≥ n0



f(n) = Omega(g(n))

Mathematical Representation of Omega notation :

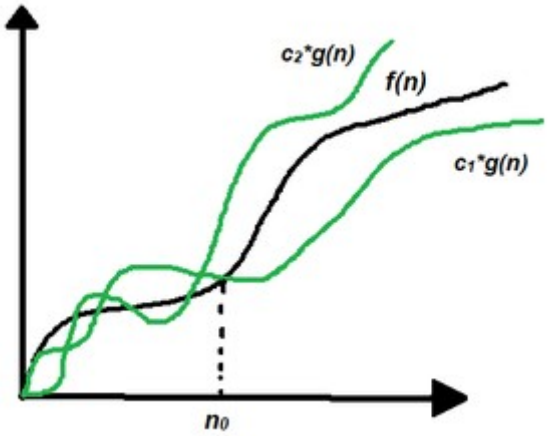
Ω(g(n)) = { f(n): there exist positive constants c and n0 such that 0 ≤ cg(n) ≤ f(n) for all n ≥ n0 }

3. ***Theta Notation (Θ-notation):*** *Average case analysis*

Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the ***average-case*** *complexity of an algorithm.*

.Theta (Average Case) You add the running times for each possible input combination and take the average in the average case.

Let g and f be the function from the set of natural numbers to itself. The function f is said to be Θ(g), if there are constants c1, c2 > 0 and a natural number n0 such that c1* g(n) ≤ f(n) ≤ c2 * g(n) for all n ≥ n0



Theta notation

Mathematical Representation of Theta notation:

*Θ (g(n)) = {f(n): there exist positive constants c1, c2 and n0 such that 0 ≤ c1 * g(n) ≤ f(n) ≤ c2 * g(n) for all n ≥ n0}*