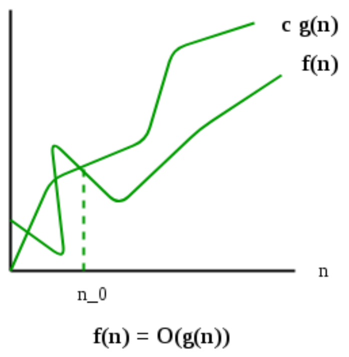
Text:

1. Asymptotic Notations

The main idea of asymptotic analysis is to have a measure of efficiency of algorithms that doesn't depend on machine specific constants, and doesn't require algorithms to be implemented and time taken by programs to be compared. Asymptotic notations are mathematical tools to represent time complexity of algorithms for asymptotic analysis. The following three asymptotic notations are mostly used to represent time complexity of algorithms.

1.1. The Big O Notation:

The *Big O* notation defines an upper bound of an algorithm, it bounds a function only from above. For example, consider the case of *Insertion Sort*. It takes linear time in best case and quadratic time in worst case. We can safely say that the time complexity of Insertion sort is $O(n^2)$. Note that $O(n^2)$ also covers linear time.



The Big *O* notation is useful when we only have upper bound on time complexity of an algorithm. Many times we easily find an upper bound by simply looking at the algorithm.

For a given function *g(n)*, we denote by *O(g(n))* the set of functions.

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

Example 7.1:

$f(n) = 2n + 3$

$2n + 3 \le 10n \quad \forall\ n \ge 1$
Here, $c=10$, $n_0=1$, $g(n)=n$
$\Rightarrow f(n) = O(n)$

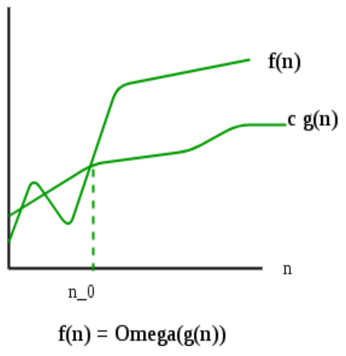
Also, $2n + 3 \le 2n + 3n$
$2n + 3 \le 5n \quad \forall\ n \ge 1$

And, $2n + 3 \le 2n^2 + 3n^2$
$2n + 3 \le 5n^2$
$\Rightarrow f(n) = O(n^2)$

$$O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(3^n) < O(n^n)$$

1.2. The Omega (Ω) notation:

Just as *Big O* notation provides an asymptotic upper bound on a function, *Ω* notation provides an asymptotic lower bound.



Ω notation can be useful when we have lower bound on time complexity of an algorithm. As discussed in the previously, the best case performance of an algorithm is generally not useful, the omega notation is the least used notation among all three.

For a given function *g(n)*, we denote by *Ω(g(n))* the set of functions.

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

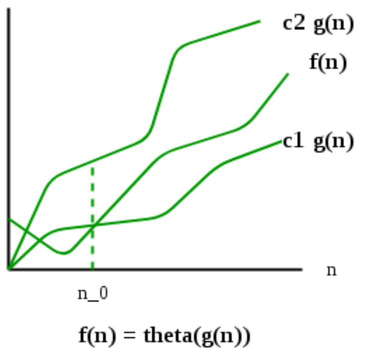
Let us consider the same *insertion sort* example here. The time complexity of *insertion sort* can be written as *Ω(n)*, but it is not a very useful information about insertion sort, as we are generally interested in worst case and sometimes in average case.

Example 7.2:

f(n) = 2n + 3

2n + 3 **≥** *n ∀ n ≥ 1*
Here, c=1, n_0=1, g(n)=n
=> f(n) = Ω(n)
Also, f(n) = Ω(log n)
f(n) = Ω(√n)

1.3. The Theta (Θ) notation:



The theta notation bounds a functions from above and below, so it defines the exact asymptotic behaviour. A simple way to get theta notation of an expression is to drop low order terms and ignore leading constants. For example, consider the following expression.

$3n^3 + 6n^2 + 6000 = \Theta(n^3)$

Dropping lower order terms is always fine because there will always be a n_0 after which $\Theta(n^3)$ has higher values than $\Theta(n^2)$ irrespective of the constants involved. For a given function *g(n)*, we denote *Θ(g(n))* as the following set of functions.

Θ(g(n)) = {f(n): there exist positive constants c_1, c_2 and n_0 such that $0 \le c_1 \times g(n) \le f(n) \le c_2 \times g(n)$ for all $n \ge n_0$}

The above definition means, if *f(n)* is theta of *g(n)*, then the value *f(n)* is always between $c_1 \times g(n)$ and $c_2 \times g(n)$ for large values of n ($n \geq n_0$). The definition of theta also requires that *f(n)* must be non-negative for values of n greater than n_0.

Example 7.4:

$f(n) = 2n + 3$

$1 * n \leq 2n + 3 \leq 5n \quad \forall\ n \geq 1$

Here, $c_1=1$, $c_2 = 5$, $n_0=1$, $g(n)=n$

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

Example 7.5:

$f(n) = 2n^2 + 3n + 4$

$2n^2 + 3n + 4 \leq 2n^2 + 3n^2 + 4n^2$

$2n^2 + 3n + 4 \leq 9n^2$

$f(n) = O(n^2)$

also, $\quad 2n^2 + 3n + 4 \geq 1 * n^2$

$f(n) = \Omega(n^2)$

$\Rightarrow 1 * n^2 \leq 2n^2 + 3n + 4 \leq 9n^2 \quad \forall\ n \geq 1$

Here, $c_1=1$, $c_2 = 9$, $n_0=1$, $g(n)= n^2$

$\Rightarrow f(n) = \Theta(n^2)$

Example 7.6:

$f(n) = n^2 \log n + n$

$n^2 \log n \leq n^2 \log n + n \leq 10\, n^2 \log n$

$\Omega(n^2 \log n) \quad \Theta(n^2 \log n) \quad O(n^2 \log n)$

Example 7.7:

$f(n) = n!$

$= 1 \times 2 \times 3 \times 4 \times \ldots \times n$

$1 \times 1 \times 1 \times \ldots \times 1 \leq 1 \times 2 \times 3 \times 4 \times \ldots \times n \leq n \times n \times n \times \ldots \times n$

$1 \leq n! \leq n^n$

$\Omega(1) \quad O(n^n)$ (Here we cannot find the average or tight bound Θ)