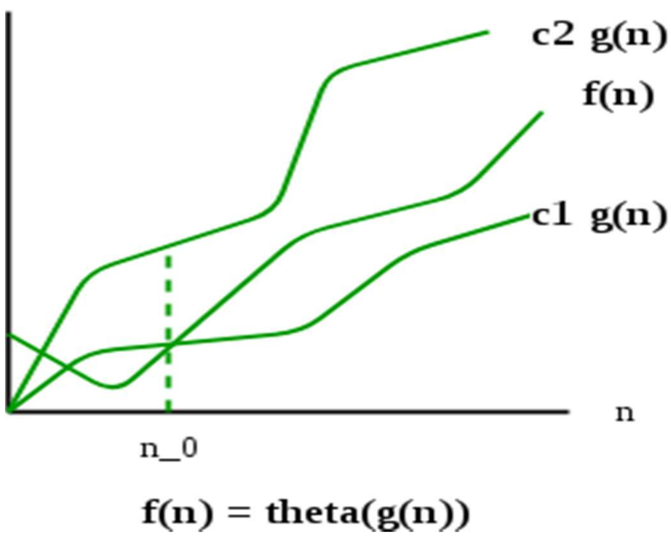
Analysis of Algorithms

- We have discussed Asymptotic Analysis, and Worst, Average, and Best Cases of Algorithms.
- The main idea of asymptotic analysis is to have a measure of the efficiency of algorithms that don'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 the time complexity of algorithms for asymptotic analysis.
- The following 3 asymptotic notations are mostly used to represent the time complexity of algorithms.



1) Θ Notation:

The theta notation bounds a function 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.

For example, consider the following expression. $3n^3 + 6n^2 + 6000 = \Theta(n^3)$

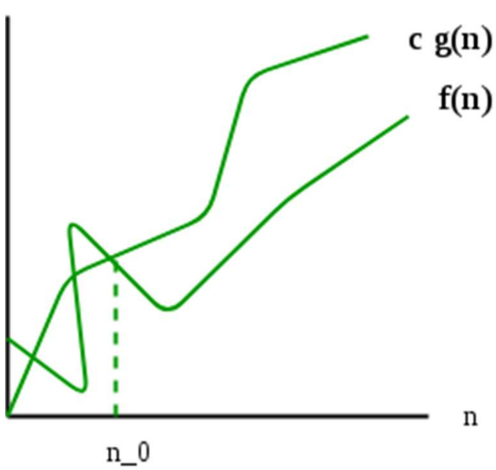
Dropping lower order terms is always fine because there will always be a n0 after which $\Theta(n^3)$ has higher values than $\Theta n^2)$ irrespective of the constants involved.

For a given function g(n), we denote Θ(g(n)) is following set of functions.

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}

The above definition means, if f(n) is theta of g(n), then the value f(n) is always between c1*g(n) and c2*g(n) for large values of n (n >= n0).

The definition of theta also requires that f(n) must be non-negative for values of n greater than n0.



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

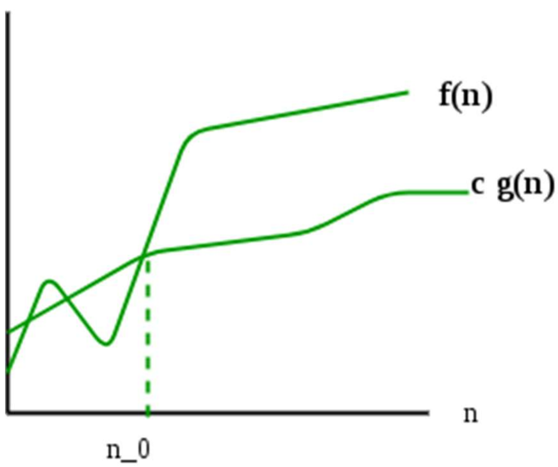
2) Big O Notation:

- 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.
  If we use Θ notation to represent time complexity of Insertion sort, we have to use two statements for best and worst cases:
  1. The worst case time complexity of Insertion Sort is Θ(n^2).
  2. The best case time complexity of Insertion Sort is Θ(n).

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.

O(g(n)) = { f(n): there exist positive constants c and
n0 such that 0 <= f(n) <= c*g(n) for
all n >= n0}



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

3) 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 previous post, 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 n0 such that 0 <= c*g(n) <= f(n) for all n >= n0}.

- 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.

Properties of Asymptotic Notations :

As we have gone through the definition of this three notations let's now discuss some important properties of those notations.

1. General Properties :

If f(n) is O(g(n)) then a*f(n) is also O(g(n)) ; where a is a constant.

Example: f(n) = $2n^2+5$ is $O(n^2)$

then 7*f(n) = $7(2n^2+5) = 14n^2+35$ is also $O(n^2)$.

Similarly this property satisfies for both Θ and Ω notation.

We can say

If f(n) is Θ(g(n)) then a*f(n) is also Θ(g(n)) ; where a is a constant.
If f(n) is Ω (g(n)) then a*f(n) is also Ω (g(n)) ; where a is a constant.

2. Transitive Properties :

If f(n) is O(g(n)) and g(n) is O(h(n)) then f(n) = O(h(n)) .

Example: if f(n) = n, g(n) = n^2 and h(n)=n^3
n is $O(n^2)$ and n^2 is $O(n^3)$

then n is $O(n^3)$

Similarly this property satisfies for both Θ and Ω notation.

We can say

If f(n) is Θ(g(n)) and g(n) is Θ(h(n)) then f(n) = Θ(h(n)) .
If f(n) is Ω (g(n)) and g(n) is Ω (h(n)) then f(n) = Ω (h(n))

3. Reflexive Properties:

Reflexive properties are always easy to understand after transitive.

If f(n) is given then f(n) is O(f(n)). Since *MAXIMUM VALUE OF f(n) will be f(n) ITSELF !*

Hence x = f(n) and y = O(f(n) tie themselves in reflexive relation always.

Example: f(n) = n^2 ; $O(n^2)$ i.e O(f(n))

Similarly this property satisfies for both Θ and Ω notation.

We can say that:

If f(n) is given then f(n) is Θ(f(n)).

If f(n) is given then f(n) is Ω (f(n)).

4. Symmetric Properties:

If f(n) is Θ(g(n)) then g(n) is Θ(f(n)) .

Example: f(n) = n^2 and g(n) = n^2
then f(n) = $\Theta(n^2)$ and g(n) = $\Theta(n^2)$

This property only satisfies for Θ notation.

5. Transpose Symmetric Properties :

If $f(n)$ is $O(g(n))$ then $g(n)$ is $\Omega(f(n))$.

Example: $f(n) = n$, $g(n) = n^2$

then n is $O(n^2)$ and n^2 is $\Omega(n)$

This property only satisfies for O and Ω notations.

6. Some More Properties :

1.) If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ then $f(n) = \Theta(g(n))$

2.) If $f(n) = O(g(n))$ and $d(n)=O(e(n))$

then $f(n) + d(n) = O(\max(g(n), e(n)))$

Example: $f(n) = n$ i.e $O(n)$

$d(n) = n^2$ i.e $O(n^2)$

then $f(n) + d(n) = n + n^2$ i.e $O(n^2)$

3.) If $f(n)=O(g(n))$ and $d(n)=O(e(n))$

then $f(n) * d(n) = O(g(n) * e(n))$

Example: $f(n) = n$ i.e $O(n)$

$d(n) = n^2$ i.e $O(n^2)$

then $f(n) * d(n) = n * n^2 = n^3$ i.e $O(n^3)$
