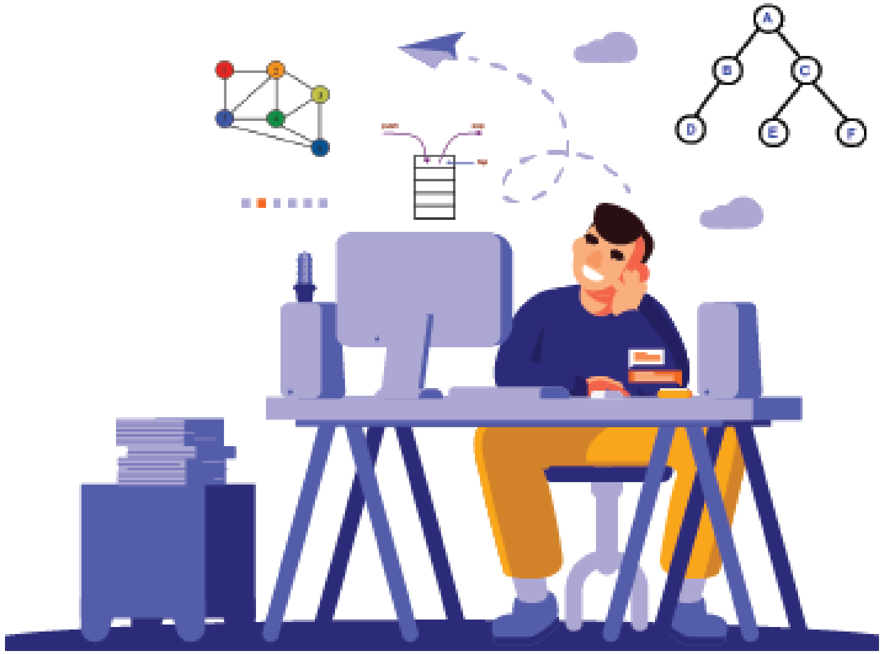
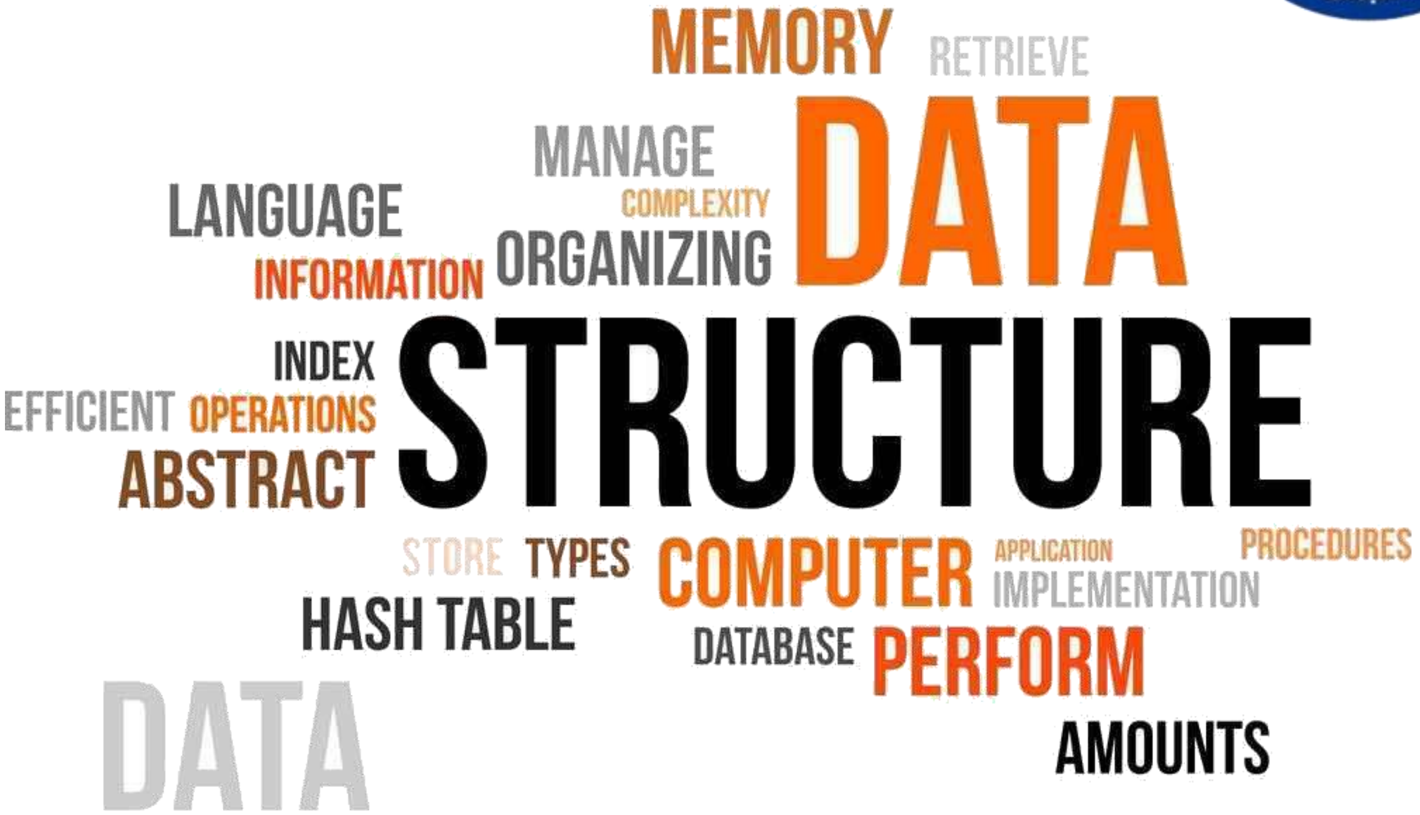


Data Structures

Course code: IT623



Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

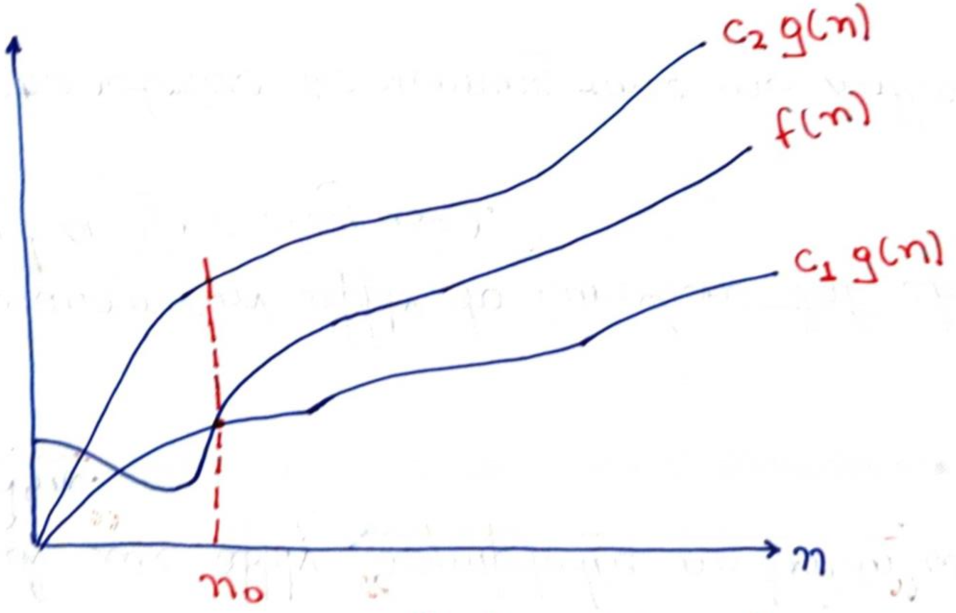
Lectures 7

Algorithm analysis and Run time

Asymptotic Analysis: $\theta - notation$

Let us define what this notation mean. For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions.

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



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

Asymptotic Analysis: $\theta - notation$

Θ - Notation

* A function $\boxed{f(n)}$ belongs to the set $\boxed{\Theta(g(n))}$ if there exist positive constants $\boxed{c_1}$ and $\boxed{c_2}$ such that it can be "sandwiched" between $\boxed{c_1 g(n) \text{ and } c_2 g(n)}$, for sufficiently large $\boxed{n}$.
  * → (read as f of n is theta of g of n)
* Since $\Theta(g(n))$ is a set, we can write "$f(n) \in \Theta(g(n))$" to indicate that $f(n)$ is a member of $\Theta(g(n))$.
* Instead, we will usually write "$f(n) = \Theta(g(n))$" to express the same notion.
* An intuitive picture of functions $\boxed{f(n) \text{ and } g(n)}$, where $f(n) = \Theta(g(n))$.
* For all values of n at and to the right of n_0, the value of $f(n)$ lies at or above $\boxed{c_1 g(n)}$ and at or below $\boxed{c_2 g(n)}$.
* $g(n)$ is an asymptotically tight bound for $f(n)$.

Asymptotic Analysis: $\theta - notation$

The definition of θ(g(n)) require that every member $f(n) \in \theta\left(g_{(n)}\right)$ by asymptotically non-negative, that is, that f**(n)** be non-negative whenever **n** is sufficiently large.

Example → $f(n) = 18n + 9$

since $f(n) \geq 18n$ and $f(n) \leq 27n$,

for $n \geq 1$

- $g(n) = 3n$

$$c_1 3n \leq \underbrace{18n+9}_{f(n)} \leq c_2 3n$$

$c_1 = 6 \quad c_2 = 9$

Asymptotic Analysis: $\theta - notation$

* Let us briefly justify this intuition by using the formal definition to show that

$$\boxed{\tfrac{1}{2}n^2 - 3n = \Theta(n^2)}$$

* To do so, we must determine positive constants c_1, c_2, and n_0 such that

$$\boxed{c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2}$$

for all $n \geq n_0$. Dividing by n^2 yields.

$$\boxed{c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2}$$

* We can make the right-hand inequality hold for any value of $\boxed{n \geq 1}$ by choosing any constant $\boxed{c_2 \geq 1/2}$.

* We can make the left-hand inequality hold for any value of $\boxed{n \geq 7}$ by choosing any constant $\boxed{c_1 \le 1/14}$.

Asymptotic Analysis: $\theta - notation$

* By choosing $c_1 = 1/14$, $c_2 = 1/2$, and $n_0 = 7$, we can verify that $\boxed{\frac{1}{2}n^2 - 3n = \Theta(n^2).}$

* Certainly, other choices for the constants exist, but the important thing is that some choice exists.

* These constants depend on the function $\boxed{\frac{1}{2}n^2 - 3n}$; a different function belonging to $\boxed{\Theta(n^2)}$ would usually requires different constants.

* We can also use the formal definition to verify that $\boxed{6n^3 \neq \Theta(n^2).}$ Suppose for the purpose of contradiction that $\boxed{c_2}$ and $\boxed{n_0}$ exist such that

$$\boxed{6n^3 \leq c_2 n^2} \quad \text{for all } n \geq n_0.$$

* But then dividing by $\boxed{n^2}$ yield $\boxed{n \leq c_2/6}$, which cannot possibly hold for arbitrarily large n, since c_2 is constant.

Asymptotic Analysis: $\theta - \boldsymbol{notation}$

* The lower-order terms of an asymptotically positive function can be ignored in determining asymptotically tight bounds because they are insignificant for large n.

* When n is large, even a tiny fraction of the highest-order term suffices to dominate the lower-order terms.

* Thus, setting c_1 to a value that is slightly smaller than the coefficient of the highest-order term and setting c_2 to a value that is slightly ~~smaller than the coefficient~~ larger permits the inequalities in the definition of Θ-notation to be satisfied.

$$\boxed{f(n) = an^2 + bn + c}$$

where $a, b,$ and c are constants and $a > 0$.

* The lower-order terms and ignoring the constant yields $f(n) = \Theta(n^2)$.

* To show the same thing, we take the constants $\boxed{c_1 = a/4,\ c_2 = 7a/4,\ \text{and}\ n_0 = 2}$.

$$\boxed{n_0 = 2 \cdot \max\left(|b|/a, \sqrt{|c|/a}\right)}$$

Asymptotic Analysis: $\theta - \boldsymbol{notation}$

We can verify that $0 \le c_1 n^2 \le an^2 + bn + c \le c_2 n^2$ for all n>= n0.

* In general, for any polynomial

$\boxed{p(n) = \sum_{i=0}^{d} a_i n^i,}$ where the a_i are constants and $a_d > 0$

* We have $p(n) = \Theta(n^d)$.

* Since any constant is a degree-0 polynomial, we can express any constant function as $\Theta(n^0)$, or $\Theta(1)$.

* We shall often use the notation $\Theta(1)$ to mean either a constant or a constant function with respect to some variable.

Asymptotic Analysis: $O - notation$

* The Θ-notation asymptotically bounds a function from above and below.

* When, we have only an asymptotic upper bound, we use O-notation

* For a given function g(n), we denote by O(g(n)) (pronounced "big-oh of g of n" or sometimes just "oh of g of n") the set of functions.

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

* Precise definition: "Schaum's outlines"

Suppose $f(n)$ and $g(n)$ are functions defined on positive integers with the property that $f(n)$ is bounded by some multiplier of $g(n)$ for almost all n. That is, suppose there exist a positive integer n_0 and a positive number c such that, for all $n > n_0$, we have

Asymptotic Analysis: $O - notation$

$$\boxed{|f(n)| \leq c\,|g(n)|}$$

This is also written as: $\boxed{f(n) = O(g(n))}$

It is read as "f(n) is of order g(n)." For any polynomial P(n) of degree m, we show in solved ~~that~~ $P(n) = O(n^m)$; e.g.;

$$\boxed{8n^3 - 576n^2 + 832n - 28 = O(n^3)}$$

We can also write

$$\boxed{f(n) = h(n) + O(g(n)) \text{ when } f(n) - h(n) = O(g(n))}$$

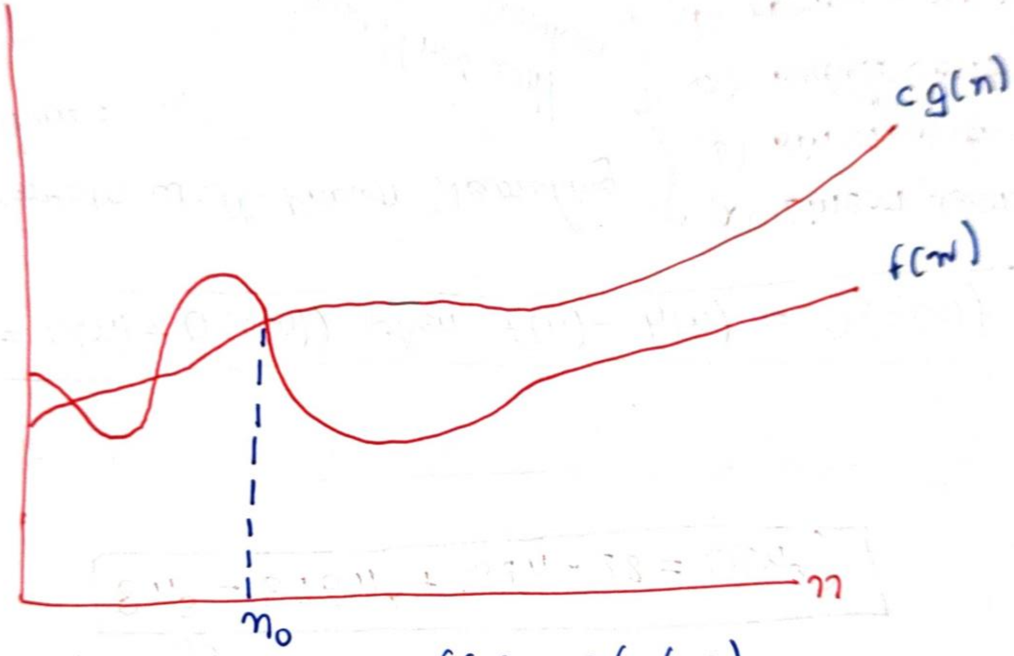
The complexity of certain well-known searching and sorting algorithms:

(a) Linear search: $O(n)$
(b) Binary search: $O(\log n)$
(c) Bubble sort: $O(n^2)$
(d) Merge sort: $O(n \log n)$.

These algorithms will be discussed in further lectures.

Asymptotic Analysis: $O - notation$

* We use O-notation to give an upper bound on a function, to within a constant factor.



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

* For all values n at and to the right of n_0, the value of the function $f(n)$ is on or below $c\,g(n)$.

Asymptotic Analysis: $O - notation$

* We write $\boxed{f(n) = O(g(n))}$ to indicate that a function $\boxed{f(n)}$ is a member of the set $\boxed{O(g(n))}$.

* $\boxed{f(n) = \Theta(g(n))}$ implies $\boxed{f(n) = O(g(n))}$, since Θ-notation is a stronger notation than O-notation.

* Written set-theoretically, we have $\boxed{\Theta(g(n)) \subseteq O(g(n))}$.

* Our proof that any quadratic function $\boxed{an^2+bn+c}$ where $a>0$ is in $\Theta(n^2)$ also shows that any such quadratic function is in $O(n^2)$.

* A surprizing aspect:

  when $a>0$, any linear function $\{an+b\}$ is $O(n^2)$

* It is easily verified by taking $c = a + |b|$ and $\boxed{n_0 = \max(1, -b/a)}$.

Asymptotic Analysis: $O - notation$

* Using O-notation, we can often describe the running time of an algorithm merely by inspecting the algorithm's overall structure.

* For example, the doubly nested loop structure of insertion sort algorithm immediately yields an $O(n^2)$ upper bound on the worst-case running time:

* Since O-notation describes an upper bound, when we use it to bound the worst-case running time of an algorithm, we have a bound on the running time of the algorithm on every input.

* $O(n^2)$ bound on worst-case running time of insertion sort also applies to its running time on every input.

* When we say "the running time is $O(n^2)$", we mean that there is a function $f(n)$ that is $O(n^2)$ such that for any value of n, no matter what particular input of Size n

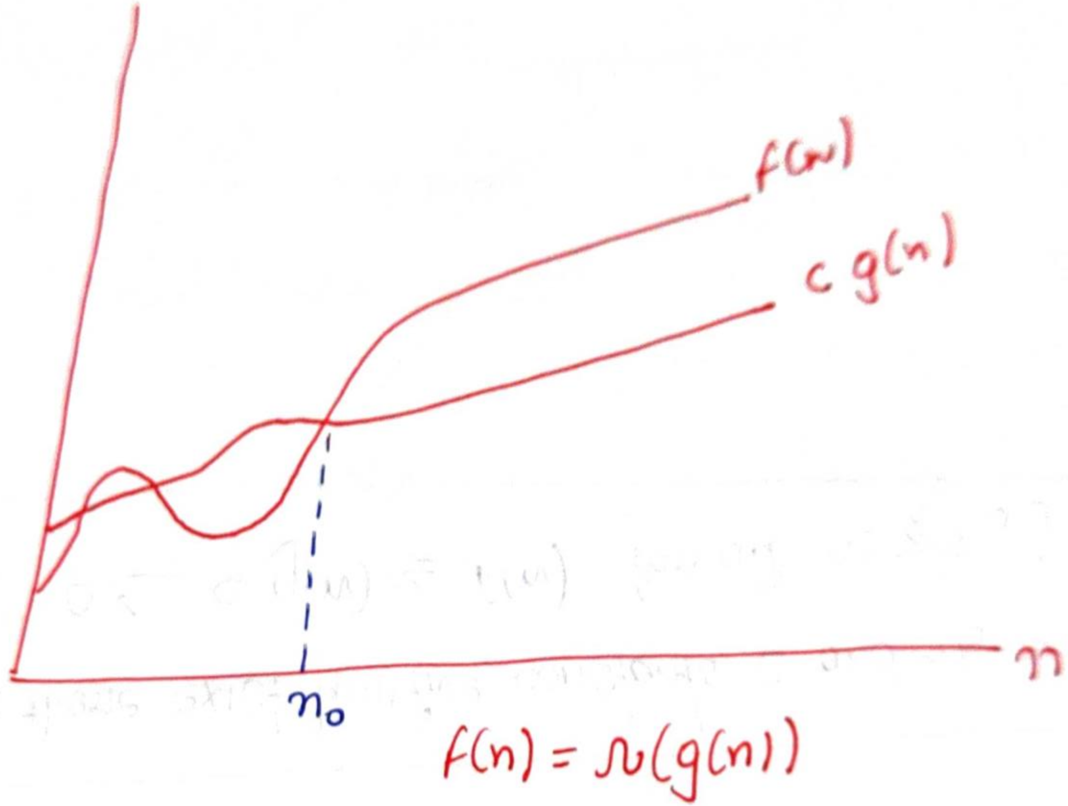
Asymptotic Analysis: $\Omega - \boldsymbol{notation}$

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

For a given function $\boxed{g(n)}$, we denote by $\boxed{\Omega(g(n))}$ (pronounced "big-omega of g of n" or sometimes just "omega of g of n") the set of functions.

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

Asymptotic Analysis: $\Omega - notation$



For all values n at or to the right of n_0, the value of f(n) is on or above cg(n).

Asymptotic Analysis: $\Omega - notation$

Definition:

$f(n) = \Omega(g(n))$ (read as f of n is omega of g of n), iff there exists a positive number n_0 and a positive number M such that $|f(n)| \geq M|g(n)|$, for all $n \geq n_0$.

C

For $\boxed{f(n) = 18n+9, \; f(n) > 18n}$ for all n, hence $f(n) = \Omega(n)$. Also, for $\boxed{f(n) = 90n^2 + 18n + 6, \; f(n) > 90n^2}$ for $\underline{n \geq 0}$ and therefore $\boxed{f(n) = \Omega(n^2)}$

* For $f(n) = \Omega(g(n))$, $g(n)$ is a lower bound function and there may be several such functions.

* However, it is appropriate that the function which is almost as large a function of n as possible such that the definition of Ω is satisfied, is chosen as $g(n)$.

Asymptotic Analysis: $\Omega - \boldsymbol{notation}$

Theorem 3.1

For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if

$$\boxed{f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))}$$

⇒ The earlier proof that $an^2 + bn + c = \Theta(n^2)$ for any constant $a, b,$ and c, where $a > 0$, immediately implies that $an^2 + bn + c = \Omega(n^2)$ and $an^2 + bn + c = O(n^2)$.

★ In practice, rather than using the theorem to obtain asymptotic upper and lower bounds we usually use it to prove asymptotically tight bounds.

" When we say that the running time of an algorithm is $\Omega(g(n))$, we mean that no matter what particular input of size n is chosen for each value of n, the running time on that input is at least a constant time $g(n)$. "

Asymptotic Analysis: $\Omega - notation$

★ Asymptotic notation in equations and inequalities

→ when the asymptotic notations stands alone (that is, not within a larger formula) on the right-hand side of an equation, as in $n = O(n^2)$

→ The equality sign set membership: $n \in O(n^2)$.

→ However, when asymptotic notation appears in a formula, we interpret it as standing for some anonmous function:

→ For example, the formula $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means that $2n^2 + 3n + 1 = 2n^2 + f(n)$ where $f(n)$ is some function in the set $\Theta(n)$.

★ Thus, we can chain together a number of such relationships, as in

$$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$$
$$= \Theta(n^2)$$

Asymptotic Analysis: o – *notation*

o-notation "little-oh of g of n":

* The asymptotic upper bound provided by O-notation may or may not be asymptotically tight.
* The bound $2n^2 = O(n^2)$ is asymptotically tight, but the bound $2n = O(n^2)$ is not.

* We use o-notation as the set

$$o(g(n)) = \{ f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0 \}$$

[For example: $2n = o(n^2)$ but $2n^2 \neq o(n^2)$

* The definitions of O-notation and o-notation are similar. The main difference is that in $f(n) = O(g(n))$, the bound $\boxed{0 \leq f(n) \leq c(g(n))}$ holds for some constants $c > 0$, but in $f(n) = o(g(n))$, the bound $\boxed{0 \leq f(n) < cg(n)}$ holds for all constants $c > 0$.

* In o-notation, the function $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity: $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

Asymptotic Analysis: ω – *notation*

ω-notation:

By analogy, ω-notation is to Ω notation as o-notation is to O-notation. We use ω-notation to denote a lower bound that is not asymptotically tight.

$f(n) \in \omega(g(n))$ if and only if $g(n) \in o(f(n))$.

We define $\omega(g(n))$ ("little-omega of g of n") as the set

$\omega(g(n)) = \{ f(n) :$ for any positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \le c\,g(n) < f(n)$ for all $n \geq n_0 \}$

For example, $n^2/2 = \omega(n)$, but $n^2/2 \neq \omega(n^2)$. The relation $f(n) = \omega(g(n))$ implies that

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

if the limit exists. That is $f(n)$ becomes arbitrarily large relative to $g(n)$ as n approaches infinity

Asymptotic Analysis: *numerical example*

1. **Big O notation:** Consider the function f(n) = n^3 + 2n^2 + 5. The big O notation for f(n) is O(n^3) because the highest-order term in the polynomial is n^3. As n gets larger, the contribution of the other terms becomes relatively insignificant compared to the dominant n^3 term.

2. **Big Omega notation:** Consider the function f(n) = n^2 + 10n. The big Omega notation for f(n) is Ω(n^2) because the lowest order term in the polynomial is n^2. As n gets larger, the contribution of the other term (10n) becomes relatively insignificant compared to the dominant n^2 term.

3. **Big Theta notation:** Consider the function f(n) = 3n^2 + 4n + 2. The big Theta notation for f(n) is Θ(n^2) because the highest and lowest order terms in the polynomial are n^2. As n gets larger, the contribution of the other term (4n + 2) becomes relatively insignificant compared to the dominant n^2 term.

Asymptotic Analysis: *numerical example*

4. Little O notation: Consider the function f(n) = n^2 + 2log(n). The little O notation for f(n) is o(n^2) because the contribution of the log(n) term becomes negligible as n approaches infinity compared to the dominant n^2 term.

5. Little Omega notation: Consider the function f(n) = n^2 + nlog(n). The little Omega notation for f(n) is ω(n^2) because the contribution of the nlog(n) term grows faster than n^2 as n approaches infinity.

Properties of Asymptotic Notations

Asymptotic notation, which includes ***Big O ("O"), Big Omega ("Ω"), Big Theta ("θ"), Little o ("o"), and Little Omega ("ω"),*** have several properties that make them useful in analyzing and comparing the computational complexity of algorithms and functions.

1. **Reflexivity:** Any function is asymptotically equivalent to itself. This means that $f(n) = \Theta(f(n))$, $f(n) = O(f(n))$, and $f(n) = \Omega(f(n))$ are always true.

2. **Transitivity:** If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$. Similarly, if $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n))$, then $f(n) = \Omega(h(n))$. This property allows us to compare the asymptotic behavior of different functions.

3. **Symmetry:** If $f(n) = \Theta(g(n))$, then $g(n) = \Theta(f(n))$. This property allows us to interchange the roles of f(n) and g(n) when analyzing the complexity of algorithms.

Properties of Asymptotic Notations

4. Addition: If $f(n) = \Theta(h(n))$ and $g(n) = \Theta(k(n))$, then $f(n) + g(n) = \Theta(\max(h(n), k(n)))$. This property allows us to analyze the complexity of an algorithm that involves multiple functions.

5. Multiplication: If $f(n) = O(h(n))$ and $g(n) = O(k(n))$, then $f(n)g(n) = O(h(n)k(n))$. This property allows us to analyze the complexity of an algorithm that involves multiple functions.

6. Transpose: If f(n) is a non-negative monotonic increasing function, then $f^{-1}(n) = \Theta(f(n))$, where f^{-1} is the inverse function of f. This property allows us to analyze the complexity of the inverse of a function.

These properties of asymptotic notation allow us to analyze the complexity of algorithms and functions and compare their efficiency in terms of time and space complexity.

Questions on Asymptotic Notations

1. What is the Big O notation for the function ***f(n) = 2n^2 + 3n + 1?***

2. What is the Big Omega notation for the function ***g(n) = 4n + 2log(n)?***

3. If ***f(n) = 3n^2 + 2nlog(n) and g(n) = 5nlog(n)***, which function has a higher order of growth as n approaches infinity?

4. If ***f(n) = 2^n and g(n) = n!***, which function has a higher order of growth as n approaches infinity?

5. What is the Big Theta notation for the function ***h(n) = n^3 + 5n^2 + 3n + 1?***

Questions and Answers on Asymptotic Notations

1. What is the Big O notation for the function f(n) = 2n^2 + 3n + 1? **Answer:** The Big O notation for f(n) is O(n^2) because the highest order term in the polynomial is n^2.

2. What is the Big Omega notation for the function g(n) = 4n + 2log(n)? **Answer:** The Big Omega notation for g(n) is Ω(n) because the lowest order term in the polynomial is n.

3. If f(n) = 3n^2 + 2nlog(n) and g(n) = 5nlog(n), which function has a higher order of growth as n approaches infinity? **Answer:** To compare the growth rate of the two functions, we need to compare their dominant terms. The dominant term of f(n) is n^2 and the dominant term of g(n) is nlog(n). Therefore, as n approaches infinity, f(n) has a higher order of growth than g(n).