Asymptotic Notations - GATE Bits in PDF

Asymptotic Notations is an important chapter in **Design and Analysis of Algorithms**, which carries over to bigger topics later on. It is useful for all of Algorithms in **GATE CS, BARC, BSNL, DRDO, ISRO**, and other exams. These **GATE Bits** on Asymptotic Notations can be **downloaded in PDF** for your reference any time. Use these GATE Study Notes to help you ace any exam.

When we compare the relative performance of alternative algorithms to solve same problem, we do not need exact time or space respectively. (we are permitted to be little sloppy).

Analysis is meaningful only when the input size is large. Assume two algorithms A1 and A2 are proposed to solve a particular problem. and

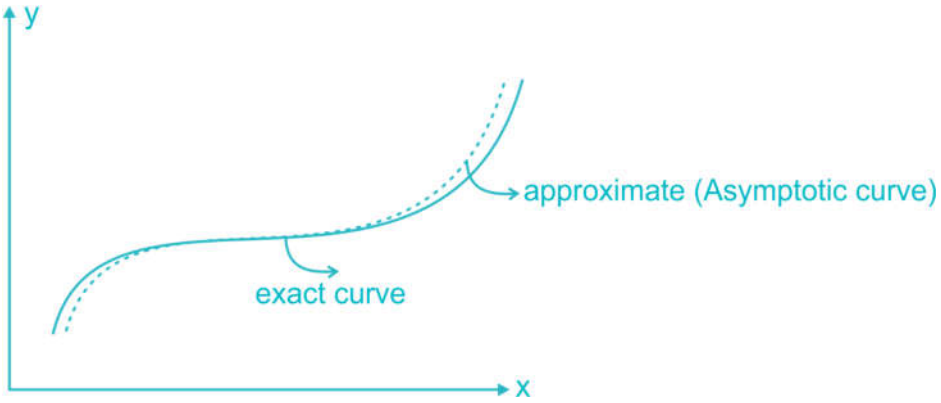
$T_{A1}(n) = 100n^2 + 10^6n$

$T_{A2}(n) = n^3$

Now at first sight it might appear that A2 is better compared to A1, but for large input size (n) only the order of growth is relevant.

Hence A1 wins over A2, because we will say time complexity of A1 is order of n^2 and time complexity of A2 is order of n^3.

To depict time and space complexities of an algorithm asymptotically (approximately) we require asymptotic notations.



Asymptotic Notations

i) Big – O (O)

After analysing an algorithm A, if one says

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

He means that algorithm will be completed within cn^2 time for a sufficiently large n.

Hence big – O given upper bound but this upper bound may or may not be the tightest.

Hence

If $T_A = O(n^2)$ then

$T_A = O(n^3)$

$T_A = O(n^4)$

In fact $T_A = O(n^{2+R^+})$

[but it does NOT mean $O(n^2) = O(n^3) = O(n^4)$ ...]

But first statement is more meaningful

Remember : $n^2 = O(n^3)$

↳ One-way equality:)

In fact, it means $n^2 \in O(n^3)$

Set definition O notation

$O(g(n)) = \{f(n)\ \exists\ c > 0,\ n_0 > 0$ such that $0 \leq f(n) \leq cg(n), \forall\ n \geq n_0\}$

Hence set O(g(n)) is the umbrella under which all asymptotically smaller functions will reside

Example for $O(n^4)$

$n \in O(n^4)$

$n^2 \in O(n^4)$

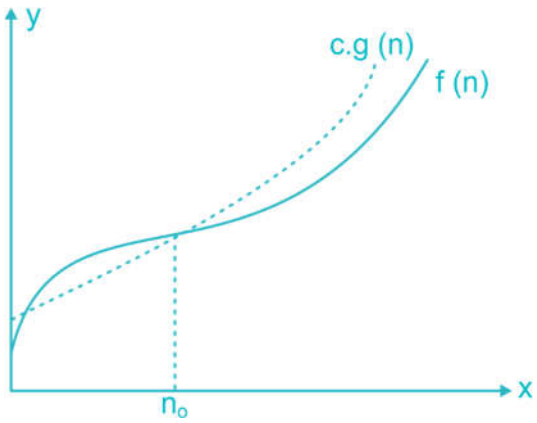
$n^3 \in O(n^4)$ etc..

Also, $f(n) = n^2 + O(n^3)$

means

$f(n) = n^2 + g(n)$

where $g(n) \in O(n^3)$



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

After $n = n_0$ $f(n)$ will never catch $c.g(n)$ where c is a constant

By writing $f(n) = O(g(n))$ we mean that a function $f(n)$ is a member of set $O(g(n))$

Since O notation describes an upper bound, it can be used to bound the worst case running time of an algorithm.

Hence it is meaningful to say that running time of insertion sort is $O(n^2)$, though we can also say $O(n^3)$, because again big –O need not provide tightest upper bound. [It may or may not be tightest].

ii. Small–o

Big –O notation is like $\leq$

$f(n) \leq g(n)$ when $f(n) = O(g(n))$

small – o notation is like $<$

$f(n) < g(n)$ when $f(n) = o(g(n))$

The asymptotic upper bound provided by O – notation may or may not be asymptotically tight.

Small – o is used to denote an upper bound that is NOT asymptotically tight.

$o(g(n)) = \{f(n)$: for any positive constant $c > 0$, $n_0 > 0$, $0 \leq f(n) < cg(n)$ $\forall$ $n > n_0\}$

Example

$3n = o(n^2)$, but $2n^2 \neq o(n^2)$

$\log n = o(n)$, but $\log n \neq o(100 \log n)$

Therefore, when

$f(n) = o(g(n))$ and n approaches infinity

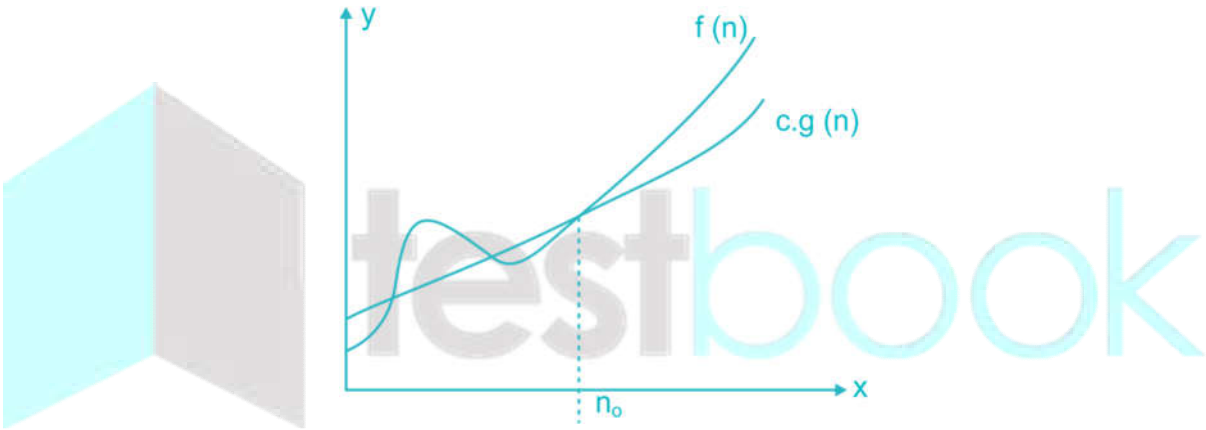
$$n \to \propto \frac{f(n)}{g(n)} = 0$$

iii. Omega (Ω)

Ω notation is used to provide an asymptotic lower bound.

$\Omega(g(n))$ is the set of functions

$\Omega(g(n)) = \{f(n): \exists\ c > 0$ and n_0 such that $0 \leq c.g(n) \leq f(n)$, for all $n \geq n_0\}$



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

As Ω notation describes a lower bound, it is used to bound the best case running time of an algorithm.

Example:

If T(n)denotes the running time of the insertion sort on input – size n, then we can say that $T(n) = \Omega(n)$ as insertion sort takes linear time in best case.

Remember in practice we use O and Ω notations to tight upper and lower bounds respectively otherwise for any function T(n)both the following statements are TRUE.

$T(n) = \Omega(0)$

$T(n) = O(\infty)$

Which means any algorithm will take at least zero time and at most infinity time to compute.

iv. Little –Omega (ω)

ω –nation is used to denote a lower bound that is not asymptotically tight.

Hence ω(g(n)) is the set

ω(g(n)) = {f(n): ∃ positive C, $n_0 > 0$ such that $0 \leq C\ g(n) < f(n)$ for all $n \geq n_0$}

Ω-notation is like ≥ where us ω notation is like >

Hence

$\sqrt{n} = \omega(\lg n) but\ \sqrt{n} \neq \omega(\sqrt{n})$

v. Theta-Notation (θ)

If two functions f(n)and g(n)are of same order, then

f(n) = θ(g(n))

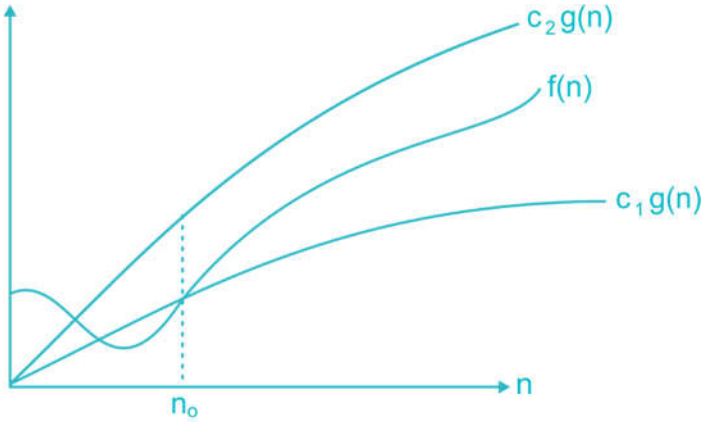
we say that g(n) is an asymptotically tight bound for f(n)

θ(g(n)) the set of functions θ(g(n)) = {f(n): ∃ c_1 , c_2 >0

and n_0 s.t. $0 \leq c_1\ g(n) \leq f(n) \leq c_2\ g(n)$ for all $n \geq n_0$}

which means

θ(g(n)) = O(g(n)) ∩ Ω(g(n))



f(n) = θ(g(n))

Hence depending upon the value of

c, c g(n) can be the tightest upper and lower bound of f(n)

Example:

Assume that in the Best Case, Average case and worst case time required by an algorithm A is 100n, 50nlogn and $10n^2$ respectively and assume we represent time complexity of A as $T_A(n)$

Then All of the following statements are TRUE.

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

$T(n) = \Omega(n)$

$T(n) = o(n^3)$

$T(n) = \omega(\sqrt{n})$

Example:

As we know that no matter what the input is mergesort always takes c.nlogn time.

Assume T(n) represents T.C. of mergesort then

$T(n) = O(nlogn)$

$T(n) = \Omega(nlogn)$

Hence

$T(n) = \theta(nlogn)$

Hence depending upon the behaviour of the algorithm on different inputs we use the most expressive and meaningful notation to represent complexity of an algorithm Following statement about time complexity of mergesort are also correct but do not make much sense:

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

$T(n) = o(n^2)$

$T(n) = \omega(n)$

$T(n) = \Omega(n)$

Now Try It Yourself

1. Which of the following set is empty?
(1) $o\left(g(n)\right)$ $\omega\left(g(n)\right)$
(2) $O\left(g(n)\right)$ $\Omega\left(g(n)\right)$

(3) $o(g(n))$ $O(g(n))$
(4) $\omega(g(n)) \cap \Omega(g(n))$

2. Assume $T(n)$ denotes the time complexity of merge sort algorithm which of the following statements is/are correct?
(i) $T(n) = o(n \log n)$
(ii) $T(n) = O(n \log n)$
(iii) $T(n) = \theta(n \log n)$
(iv) $T(n) = O(n^2)$

(1) i and iii only
(2) ii and iii only
(3) i and iv only
(4) ii, iii and iv only

3. Identify the FALSE statement:
(1) $f(n) = \theta\left(f\left(\frac{n}{2}\right)\right) implies f(n^2) = \theta\left(f\left(\frac{n^2}{2}\right)\right)$
(2) $f(n) = O(g(n)) implies \lg(f(n)) = O\left(In(g(n))\right)$
(3) $f(n) = O(g(n)) implies 2^{f(n)} = O(2^{g(n)})$
(4) $f(n) + g(n) = \theta\left(Max(f(n), g(n))\right)$

4. Given f(n) = ω(n²). Which of the following can never hold?
(1) f(n) = O (n³)
(2) f(n) = Ω (n²)
(3) f(n) = θ (n²)
(4) f(n) = ω (n)

5. Which of the following can never hold?
(1) $n^3 + \Omega(n^2) = O(n^4)$
(2) $n + \theta(n^2) = \Omega(n)$
(3) $n^2 + O(n^2) = \theta(n^3)$
(4) $n^3 + O(n^3) = O(n^4)$

6. Assume f(n) and g(n) are two functions such that f(n) = O (g(n)). Which of the following will always hold?
(1) $f(n) = O\left((f(n))^2\right)$

(2) $f(n) = \Omega\left(\left(f(n)\right)^2\right)$
(3) $g(n) = O\left(\left(f(n)\right)^2\right)$
(4) $g(n) = \Omega\left(g(n)\right)$

7. Which of the following arrangements of functions is in ascending order of growth rate. That is if g(n) follows f(n) than it should be the case that f(n) is O(g(n)).

(1) $\sqrt{\log n}, \log(\log n), 2^{\log n}, n \log n$

(2) $(\log n)^{100}, n^{\frac{1}{100}}, 2^{\log 2n^{\frac{3}{2}}}, n \log n$

(3) $\sqrt{n}, 2^{\sqrt{\log n}}, n^{\frac{1}{4}}.(\log n)^3, n^2$

(4) $2^{\sqrt{\log n}}, \sqrt{n}, n(\log n)^2, n^{\frac{3}{2}}$

(answers & solutions at the end)

Liked this article on Asymptotic Notations? Let us know in the comments. You may also like...

Theory of Computation

Rank of a Matrix & Its Properties

Eigen Values & Eigen Vectors

Linear Algebra Short Quiz

ANSWERS & SOLUTIONS to TRY IT YOURSELF QUESTIONS

Ans 1: 1
Solution: There does not exist any function $f(n)$ such that $f(n)\epsilon o\left(g(n)\right)$ and $f(n)\epsilon\omega\left(g(n)\right)$ at the same time.
Both the sets in option(A) are mutually exclusive.

Ans 2: 4

Solution: we know that time complexity of Merge Sort is $\theta(n \log n)$ hence all statements are valid except i, because little-oh notation is used to give non-tightest upper bound.

Statement (iv) is true because of the fact that if merge sort can be performed in nlogn time than it can definitely be performed in $O(n^2)$,this statement may not be very useful though.

Remember: Upper bound means "cannot be asymptotically worse than". Therefore all algorithms are O(∞). Lower bound means "cannot be asymptotically better than". So all algorithms are Ω(0).

Ans 3: 3
$e.g\ 3n = O(n)\ but\ 2^{3n} \neq O(2^n)$

Ans 4: 3
Solution: Statement C can never hold because if tight bounds are $\Omega(n^2)$ and $O(n^2)$ then $\omega(n^2)$ can never be the non-tightest lower bound.

Ans 5: 3
Solution: Statements A, B and D can hold but Statement C can never hold.

Ans 6: 4
Solution: Assume $f(n) = \frac{1}{n}$ then option 'A' will not hold.

Ans 7: 4