Asymptotic Notations and Apriori Analysis

Previous Page Next Page

In designing of Algorithm, complexity analysis of an algorithm is an essential aspect. Mainly, algorithmic complexity is concerned about its performance, how fast or slow it works.

The complexity of an algorithm describes the efficiency of the algorithm in terms of the amount of the memory required to process the data and the processing time.

Complexity of an algorithm is analyzed in two perspectives: **Time** and **Space**.

Time Complexity

It’s a function describing the amount of time required to run an algorithm in terms of the size of the input. "Time" can mean the number of memory accesses performed, the number of comparisons between integers, the number of times some inner loop is executed, or some other natural unit related to the amount of real time the algorithm will take.

Space Complexity

Asymptotic Notations

Execution time of an algorithm depends on the instruction set, processor speed, disk I/O speed, etc. Hence, we estimate the efficiency of an algorithm asymptotically.

Time function of an algorithm is represented by **T(n)**, where **n** is the input size.

Different types of asymptotic notations are used to represent the complexity of an algorithm. Following asymptotic notations are used to calculate the running time complexity of an algorithm.

- **O** − Big Oh
- **Ω** − Big omega
- **θ** − Big theta
- **o** − Little Oh
- **ω** − Little omega

O: Asymptotic Upper Bound

'O' (Big Oh) is the most commonly used notation. A function ***f(n)*** can be represented is the order of ***g(n)*** that is ***O(g(n))***, if there exists a value of positive integer **n** as $\mathbf{n_0}$ and a positive constant **c** such that −

$$f(n)\leqslant c.g(n) \text{ for } n > n_{0} \text{ in all case}$$

Hence, function ***g(n)*** is an upper bound for function ***f(n)***, as ***g(n)*** grows faster than ***f(n)***.

Example

Let us consider a given function, $f(n) = 4.n^3 + 10.n^2 + 5.n + 1$

Considering $g(n) = n^3$,

$$f(n)\leqslant 5.g(n) \text{ for all the values of } n > 2$$

Example

Let us consider a given function, $f(n) = 4.n^3 + 10.n^2 + 5.n + 1$.

Considering $g(n) = n^3$, $f(n)\geqslant 4.g(n)$ for all the values of $n > 0$.

Hence, the complexity of ***f(n)*** can be represented as $\Omega (g(n))$, i.e. $\Omega (n^3)$

θ: Asymptotic Tight Bound

We say that $f(n) = \theta(g(n))$ when there exist constants $\mathbf{c_1}$ and $\mathbf{c_2}$ that $c_{1}.g(n) \leqslant f(n) \leqslant c_{2}.g(n)$ for all sufficiently large value of **n**. Here **n** is a positive integer.

This means function **g** is a tight bound for function **f**.

Example

Let us consider a given function, $f(n) = 4.n^3 + 10.n^2 + 5.n + 1$

Considering $g(n) = n^3$, $4.g(n) \leqslant f(n) \leqslant 5.g(n)$ for all the large values of **n**.

Hence, the complexity of ***f(n)*** can be represented as $\theta (g(n))$, i.e. $\theta (n^3)$.

O - Notation

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

We use **o-notation** to denote an upper bound that is not asymptotically tight.

We formally define ***o(g(n))*** (little-oh of g of n) as the set ***f(n) = o(g(n))*** for any positive constant $c > 0$ and there exists a value $n_{0} > 0$, such that $0 \leqslant f(n) \leqslant c.g(n)$.

Let us consider the same function, $f(n) = 4.n^3 + 10.n^2 + 5.n + 1$

Considering $g(n) = n^{4}$,

$$\lim_{n \rightarrow \infty}\left(\frac{4.n^3 + 10.n^2 + 5.n + 1}{n^4}\right) = 0$$

Hence, the complexity of ***f(n)*** can be represented as $o(g(n))$, i.e. $o(n^4)$.

ω – Notation

We use **ω-notation** to denote a lower bound that is not asymptotically tight. Formally, however, we define ***ω(g(n))*** (little-omega of g of n) as the set ***f(n) = ω(g(n))*** for any positive constant ***C > 0*** and there exists a value $n_{0} > 0$, such that $0 \leqslant c.g(n) < f(n)$.

For example, $\frac{n^2}{2} = \omega (n)$, but $\frac{n^2}{2} \neq \omega (n^2)$. The relation $f(n) = \omega (g(n))$ implies that the following limit exists

$$\lim_{n \rightarrow \infty}\left(\frac{f(n)}{g(n)}\right) = \infty$$

That is, ***f(n)*** becomes arbitrarily large relative to ***g(n)*** as **n** approaches infinity.

Example

Let us consider same function, $f(n) = 4.n^3 + 10.n^2 + 5.n + 1$

Considering $g(n) = n^2$,

$$\lim_{n \rightarrow \infty}\left(\frac{4.n^3 + 10.n^2 + 5.n + 1}{n^2}\right) = \infty$$

Hence, the complexity of ***f(n)*** can be represented as $o(g(n))$, i.e. $\omega (n^2)$.

Apriori and Apostiari Analysis

Apriori analysis means, analysis is performed prior to running it on a specific system. This analysis is a stage where a function is defined using some theoretical model. Hence, we determine the time and space complexity of an algorithm by just looking at the algorithm rather than running it on a particular system with a different memory, processor, and compiler.

computer; however, asymptotically they are the same.

Previous Page Next Page