Asymptotic Analysis

Asymptotic analysis of an algorithm refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case, and worst-case scenario of an algorithm.

When it comes to analyzing the complexity of any algorithm in terms of time and space, we can never provide an exact number to define the time required and the space required by the algorithm, instead, we express it using some standard notations, also known as Asymptotic Notations.

Usually, the time required by an algorithm falls under three types −

* Best Case − Minimum time required for program execution.
* Average Case − Average time required for program execution.
* Worst Case − Maximum time required for program execution.

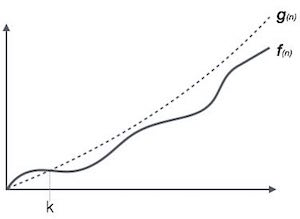
# Asymptotic Notations

Following are the commonly used asymptotic notations to calculate the running time complexity of an algorithm.

* Ο Notation
* Ω Notation
* θ Notation

## Big Oh Notation, Ο

The notation Ο(n) is the formal way to express the upper bound of an algorithm's running time. It measures the worst-case time complexity or the longest amount of time an algorithm can possibly take to complete.



For example, for a function f(n)

Ο(*f*(n)) = { *g*(n) : there exists c > 0 and n0 such that *f*(n) ≤ c.*g*(n) for all n > n0. }

## Omega Notation, Ω

The notation Ω(n) is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

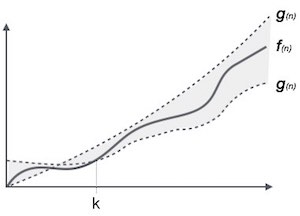


For example, for a function f(n)

Ω(*f*(n)) ≥ { *g*(n) : there exists c > 0 and n0 such that *g*(n) ≤ c.*f*(n) for all n > n0. }

## Theta Notation, θ

The notation θ(n) is the formal way to express both the lower bound and the upper bound of an algorithm's running time. It is represented as follows −



θ(*f*(n)) = { *g*(n) if and only if *g*(n) = Ο(*f*(n)) and *g*(n) = Ω(*f*(n)) for all n > n0. }

# Common Asymptotic Notations

| constant−Ο(1)  logarithmic−Ο(log n)  linear−Ο(n)  n log n−Ο(n log n)  quadratic−Ο(n2)  cubic−Ο(n3)  polynomial−nΟ(1)  exponential−2Ο(n) |
| --- |

## Why we have three different asymptotic analyses?

As we know that big omega is for the best case, big oh is for the worst-case while big theta is for the average case. Now, we will find out the average, worst, and best case of the linear search algorithm.

Suppose we have an array of n numbers, and we want to find the particular element in an array using the linear search. In the linear search, every element is compared with the searched element on each iteration. Suppose, if the match is found in a first iteration only, then the best case would be Ω(1), if the element matches with the last element, i.e., the nth element of the array then the worst case would be O(n). The average case is mid of the best and the worst-case, so it becomes θ(n/1). The constant terms can be ignored in the time complexity so the average case would be θ(n).

So, three different analyses provide the proper bounding between the actual functions. Here, bounding means that we have upper as well as lower limit which assures that the algorithm will behave between these limits only, i.e., it will not go beyond these limits.