Big Oh, Theta, Omega Notation

Notebook:	Data Structures & Algorithms Notebook
Created:	29-05-2020 10:07 AM
Author:	Sachin Pandey

This article attempts to summarize the big oh notation and other notations used to describe the running time of algorithms:

We abuse these notations frequently. Note abuse not misuse

SECTION A: BIG-OH, OMEGA, THETA NOTATIONS

Q. Why is running time of algorithms described in big oh notation?
A. For describing and comparing running times of algorithms, we are not interested in precise running times of algorithms. It gives us too much detail. In running times we can drop the constants and the lower order terms. This makes our analysis much simpler and we don't lose much predictive power.
This is because we are much more interested in order of growth as well as the asymptotic performance of the algorithms, in these conditions the lower order terms and the constants are insignificant as compared to highest order term.

Q. When is an algorithm faster than another algorithm?
A. An algorithm is considered faster then another algorithm when the asymptotic running time grows slower then the other algorithm. What it means is that for sufficiently large inputs the first algorithm beats the second algorithm.

Q. Describe the Theta notation.
A. For any function g(n), we denote by Theta(g(n)) as the set of functions

Theta(g(n)) = { f(n): there exists positive constants c_1 and c_2 and n_0 such that $0 <= c_1g(n) <= f(n) <= c_2g(n)$ for all $n>=n_0$ }

In words this says that a function f(n) belongs to the set Theta(g(n)) if there exists positive constants c_1 and c_2 such that it can be sandwiched between $c_1g(n)$ and $c_2g(n)$ for sufficiently large n.

When this happens we say: g(n) is an asymptotically tight bound for f(n) and that the function f(n) belongs to the set Theta(g(n)).

We abuse this notation to say that these are equivalent statements:

1. f(n) belongs to the set Theta(g(n)).
2. f(n) = Theta(g(n))

Corrolaries:

1. Any polynomial of degree k belongs to the set Theta(n^k).
2. Any constant or a constant function with respect to a variable belongs to the set Theta(n^0) / Theta(1).

Q. Describe the Big Oh notation.
A. For any function g(n), we denote by O(g(n)) as the set of functions

$$O(g(n)) = \{ f(n): \text{there exists positive constants } c \text{ and } n_0 \text{ such that } 0 <= f(n) <= cg(n) \text{ for all } n>=n_0 \}$$

In words this says that a function f(n) belongs to the set O(g(n)) if there exists positive constant c such that it is bounded by g(n) on the above for sufficiently large n.

When this happens we say: g(n) is an asymptotically upper bound for f(n) and that the function f(n) belongs to the set O(g(n)).

Q. Describe the Omega notation.
A. For any function g(n), we denote by Omega(g(n)) as the set of functions

$$Omega(g(n)) = \{ f(n): \text{there exists positive constants } c \text{ and } n_0 \text{ such that } 0 <= cg(n) <= f(n) \text{ for all } n>=n_0 \}$$

In words this says that a function f(n) belongs to the set Omega(g(n)) if there exists positive constant c such that it is bounded by g(n) on the below for sufficiently large n.

When this happens we say: g(n) is an asymptotically lower bound for f(n) and that the function f(n) belongs to the set Omega(g(n)).

Q. Explain the relation between the Theta, Big-Oh and Omega Notation.
A. f(n) = Theta(g(n)) if and only if f(n) = O(g(n)) and f(n) = Omega(g(n)).

Q. Explain the relation of Worst case running time with Big-Oh and Theta notation.
A. Since O notation defines a upper bound thus when we use it to give an upper bound on the worst case running time, it also gives an upper bound on the running time of an algorithm.
However this is not true of theta notation as it also bound from below, so the bound on the worst case may not bound the running times in other inputs (say best case inputs).

Q. Explain the relation of best case running time with Omega and Theta notation.

A. Since Omega notation defines a lower bound thus when we use it to give an lower bound on the best case running time, it also gives an lower bound on the running time of an algorithm.
However this is not true of theta notation.

SECTION B: ASYMPTOTIC NOTATIONS IN EQUATIONS AND INEQUALITIES