Asymptotic notation

The running time of an algorithm depends on how long it takes a computer to run the lines of code of the algorithm—and that depends on the speed of the computer, the programming language, and the compiler that translates the program from the programming language into code that runs directly on the computer, among other factors.

To analysis the running time of an algorithm we need two ideas:

- First, we need to determine how long the algorithm takes, in terms of the size of its input; i.e. the running time of the algorithm is defined as a *function of the size of its input*.
- The second idea is that we must focus on how fast a function grows with the input size. We call this the **rate of growth** of the running time.

To find an algorithm's running time as its rate of growth we use **asymptotic notation** by dropping the less significant terms and the constant coefficients.

Asymptotic analysis deals with analyzing the properties of the running time when the input size goes to infinity as orders of growths are more significant for larger input size. Analyzing the running times on small inputs does not allow us to distinguish between efficient and inefficient algorithms. Asymptotic notations are used to describe the asymptotic analysis.

Asymptotic Notations are

- big-Theta $\Theta(g(n))$
- Big-Oh $O(g(n))$
- big-Omega $\Omega(g(n))$
- little-oh $o(g(n))$
- little-omega $\omega(g(n))$

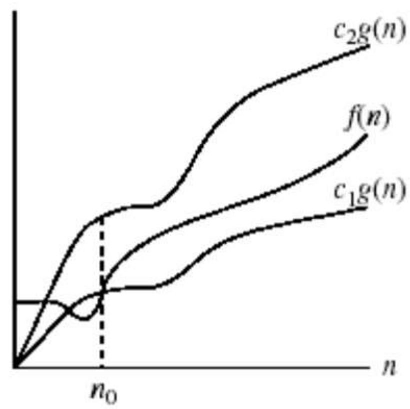
Big-Theta Θ : For a given function g(n),we denote by Θ (g(n)) , a set of functions

$\Theta(g(n))=\{f(n): \exists$ positive constants c1,c2 and n0,such that $0 \le c1\, g(n) \le f(n) \le c2\, g(n), \forall n \ge n0\}$

- We define $\Theta(g(n))$ to be a set of functions that are asymptotically equivalent to g(n)
- A function f(n) belongs to the set $\Theta(g(n))$, if there exist positive constants c1 and c2, such that g(n) can be "sandwiched" between c1g(n) and c2g(n), for sufficiently large n.

This notation is used for average case complexity.

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n)$$



Lower and upper bounds *(Example1)*

- $f(n) = 8n^2 + 2n - 3$
  - To show that $f(n) \in \Theta(n^2)$
  - We need to find the following three values.
  - c1, c2 and n_o

- **To find Lower bound we need c1 and n_o**
- **To find Upper bound we need c2 and n_o**
  - **We will have two n_o, select the maximum n_o**

Finding c_1 and n_o *(Example1)*

Lower bound: $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$

$f(n) = 8n^2 + 2n - 3$, $f(n) \in \Theta(n^2)$

- $c_1 n^2 \le 8n^2 + 2n - 3$??
  - $7n^2 \le 8n^2 + 2n - 3$
  - $c_1 = 7$
  - $N_o = 1$

c_1 can be anything lesser than the constant with n^2 of the expression

$n_o = 1$

$7(1)^2 \le 8(1)^2 + 2(1) - 3$

$7 \le 8+2-3$

$7 \le 7$

Finding c_2 and n_o *(Example1)*

Upper Bound: $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$

$f(n) = 8n^2 + 2n - 3$, $f(n) \in \Theta(n^2)$

$8n^2 + 2n - 3 \le c_2 n^2$

$8n^2 + 2n - 3 \le 9n^2$

$\Rightarrow 8n^2 + 2n - 3 \le 9n^2$

c_2 can be anything greater than the constant with n^2 of the expression

$c_2 = 9$

$N_o = 1$

Exam: Prove $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Here $f(n) = \frac{1}{2}n^2 - 3n$ and $g(n) = n^2$

Proof: To prove that we must determine c_1, c_2 and n_0 such that

$$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2 \text{ for all } n \ge n_0 \ldots\ldots (1)$$

dividing by n^2 in equ(1), we get

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

Now, $c_1 \le \frac{1}{2} - \frac{3}{n}$ and $\frac{1}{2} - \frac{3}{n} \le c_2$

or, $\frac{1}{14} \le \frac{1}{2} - \frac{3}{7}$ $\qquad$ $\frac{1}{2} - \frac{3}{1} \le \frac{1}{2}$

i.e. $c_1 \le \frac{1}{14}$ and $n_0 \ge 7$ $\qquad$ i.e. $c_2 \ge \frac{1}{2}$ and $n_0 \ge 1$

i.e. $f(n) = \Omega(g(n))$ $\qquad$ i.e. $f(n) = O(g(n))$

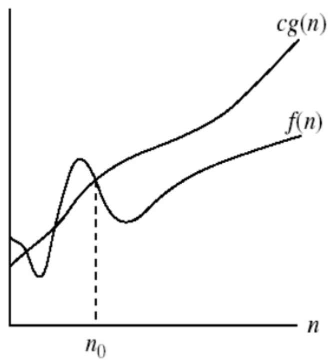
$\therefore f(n) = \Theta(g(n))$

Hence. $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Intuitively, the lower-order terms of an asymptotically positive function can be ignored in determining asymptotically tight bounds because they are insignificant for large *n*. A tiny fraction of the highest-order term is enough 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 larger permits the inequalities in the definition of Θ-notation to be satisfied. The coefficient of the highest-order term can likewise be ignored, since it only changes c_1 and c_2 by a constant factor equal to the coefficient.

Big-O notation : For a given function g(n),we denote by O(g(n)), a set of functions

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



$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

This notation is used for worst case complexity.

For all values n to the right of n0, the value of the function f(n) is on or below of g(n).

We use O-notation, when we have only an asymptotic upper bound .We write as f(n)=O(g(n))

Ex: $f(n) = 2n^2 + n$ [total time to solve a probm]

$2n^2 + n \le c.g(?)$

$\therefore 2n^2 + n \le 3 . g(n^2)$ [$\because$ We choose $g(n^2)$ $\because$ n^2 dominate n & c = 3]

$\therefore 2n^2 + n \le 3n^2$

$\therefore n \le 3n^2 - 2n^2$

$\Rightarrow n \le n^2$

$\Rightarrow 1 \le n$ [divide both side by n]

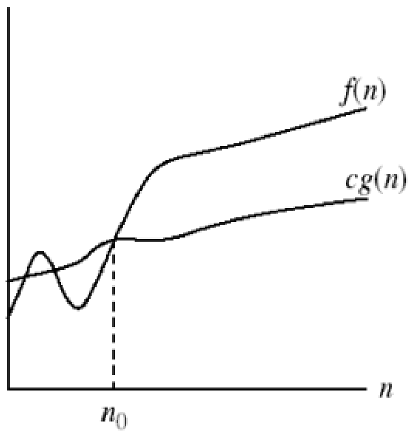
$\Rightarrow n \ge 1$

i.e $2n^2 + n \le 3n^2$ for $n \ge 1$

Big –omega notation: Just as O-notation provides an asymptotic upper bound on a function, omega notation provides an asymptotic lower bound.

Ω(g(n)) = the set of functions with a larger or same order of growth as g(n)

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



$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

This notation is used for best case complexity.

For all values n to the right of n0, the value of the function f(n) is on or above of g(n).

We use O-notation, when we have only an asymptotic upper bound .We write as f(n)= Ω(g(n))

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

$\therefore\ 2n^2 + n \geq c n^2$

$2n^2 + n \geq 2n^2$ [$\because$ 2 is closest to lower boun

$n \geq 0$