Click to add title

- Cli

Transitivity:

$$f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \text{ imply } f(n) = \Theta(h(n)),$$
$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \text{ imply } f(n) = O(h(n)),$$
$$f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) \text{ imply } f(n) = \Omega(h(n)),$$
$$f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) \text{ imply } f(n) = o(h(n)),$$
$$f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) \text{ imply } f(n) = \omega(h(n)).$$

Reflexivity:

$$f(n) = \Theta(f(n)),$$
$$f(n) = O(f(n)),$$
$$f(n) = \Omega(f(n)).$$

ω - notation

$$\omega(g(n)) = \{f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq cg(n) < f(n) \text{ for all } n \geq n_0\}$$

- We use ω-notation to denote a lower bound that is not asymptotically tight.
- For example, $n^2/2 = \omega(n)$, but $n^2/2 \neq \omega(n^2)$.
- Intuitively, in ω-notation, the function f(n) becomes arbitrarily large relative to g(n) as n approaches infinity; that is,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

- The main difference is that in f(n) = O(g(n)), the bound $0 \le f(n) \le cg(n)$ holds for *some* constant $c > 0$, but in f(n) = O(g(n)), the bound $0 \le f(n) < cg(n)$ holds for *all* constants $c > 0$.
- Intuitively, in o-notation, the function f(n) becomes insignificant relative to g(n) as n approaches infinity; that is,

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 .$$

o-notation

$$o(g(n)) = \{f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le f(n) < cg(n) \text{ for all } n \ge n_0\}$$

- The asymptotic upper bound provided by O-notation may or may not be asymptotically
- tight.
- The bound $2n^2 = O(n^2)$ is asymptotically tight, but the bound $2n = O(n^2)$ is not.
- We use o-notation to denote an upper bound that is not asymptotically tight.
- For example, $2n = o(n^2)$, but $2n^2 \neq o(n^2)$.

- The running time of insertion sort therefore belongs to both $\Omega(n)$ and $O(n^2)$.
- The running time of insertion sort is not $\Omega(n^2)$, since there exists an input for which insertion sort runs in $\Theta(n)$ time (e.g., when the input is already sorted).
- It is not contradictory, however, to say that the *worst-case* running time of insertion sort is $\Omega(n^2)$, since there exists an input that causes the algorithm to take $\Omega(n^2)$ time.

- The running time of insertion sort therefore belongs to both $\Omega(n)$ and $O(n^2)$.
- The running time of insertion sort is not $\Omega(n^2)$, since there exists an input for which insertion sort runs in $\Theta(n)$ time (e.g., when the input is already sorted).
- It is not contradictory, however, to say that the worst-case running time of insertion sort is $\Omega(n^2)$, since there exists an input that causes the algorithm to take $\Omega(n^2)$ time.

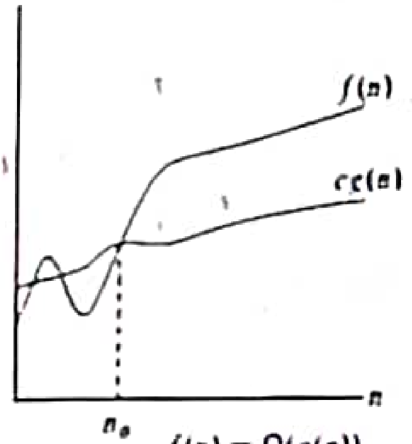
- When we say that the *running time* (no modifier) of an algorithm is $\Omega(g(n))$,we mean that *no matter what particular input of size* n *is chosen for each value of* n, the running time on that input is at least a constant times $g(n)$, for sufficiently large n. Equivalently, we are giving a lower bound on the best-case running time of an algorithm. For example, the best-case running time of insertion sort is $\Omega(n)$, which implies that the running time of insertion sort is $\Omega(n)$.

Theorem

- For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Ω-notation

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



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

- any quadratic function $an^2 + bn + c$, where $a > 0$, is in $\Theta(n^2)$ is also in $O(n^2)$.
- $an + b$ is in $O(n^2)$: $c = a + |b|$; $n_0 = \max(1, -b/a)$
- $n = O(n^2)$
- Using O-notation, we can often describe the running time of an algorithm merely by inspecting the algorithm's overall structure.
- For example, two nested for loop ==> $O(n^2)$

- Since O-notation describes an upper bound, when we use it to bound the worst case running time of an algorithm, we have a bound on the running time of the algorithm on every input—the blanket statement we discussed earlier. Thus, the $O(n^2)$ bound on worst-case running time of insertion sort also applies to its running time on every input.

- The $\Theta(n^2)$ bound on the worst-case running time of insertion sort, however, does not imply a $\Theta(n^2)$ bound on the running time of insertion sort on *every* input. For example, when the input is already sorted, insertion sort runs in $\Theta(n)$ time.

V

$\Theta(1)$: constant time

Because any constant time is a degree 0 polynomial.

$\Theta(n^0) = \Theta(1)$

S

Ex: $6n^3 \neq \Theta(n^2)$

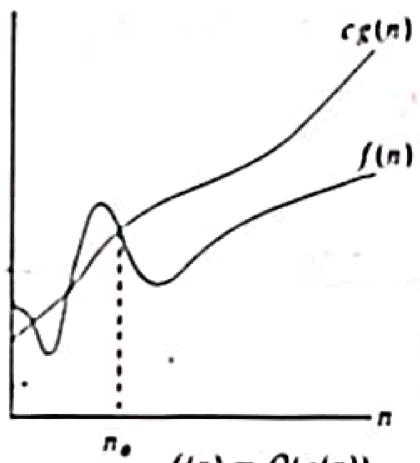
Proof: If $6n^3 = \Theta(n^2)$ then $6n^3 \leq c_2 n^2$

$\div$ by $n^2 \Rightarrow 6n \leq c_2 \Rightarrow n \leq \frac{c_2}{6}$ which is false because n is a variable and RHS is a constant.

O-notation

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

- We use O-notation to give an upper bound on a function.
- $f(n) = \Theta(g(n))$ implies $f(n) = O(g(n))$ (because Θ is a stronger notation than O)
- $O(g(n)) \subseteq \Theta(g(n))$



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

a

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

$\div$ by $n^2 \Rightarrow \quad c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$

$\frac{1}{2} - \frac{3}{n} \leq c_2 \Rightarrow$ when $c_2 \geq \frac{1}{2}$ and $n \geq 1$ this inequality holds

$c_1 \leq \frac{1}{2} - \frac{3}{n} \Rightarrow n = 7 \Rightarrow \frac{1}{2} - \frac{3}{7} = \frac{7-6}{2 \times 7} = \frac{1}{14}$

$\Rightarrow$ when $c_1 \leq \frac{1}{14}$ and $n \geq 7$ this inequality holds

- We say that g(n) is an "*asymptotically tight bound*" for f (n).
- How to find Θ: throw away lower-order terms and ignore the leading coefficient of the highest-order term.
- Ex: $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Proof:

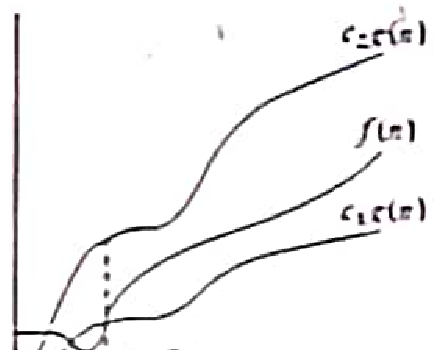
find c_1, c_2, n_0 s.t,

$$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2 \quad \forall n \ge n_0$$

Θ-notation

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \}$$

- $f(n) = \Theta(g(n))$ notation is same as $f(n) \in \Theta(g(n))$



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

- asymptotic notation will usually characterize the running times of algorithms.
- But asymptotic notation can apply to functions that characterize some other aspect of algorithms (Ex: the amount of space they use, for example)
- *which* "running time" we mean:
-             -- worst-case running time sometimes
-             -- running time no matter what the input
-             sometimes