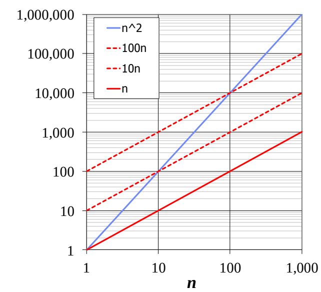
Big O Fact

- A polynomial of degree k is $O(n^k)$
- Proof:
  - Suppose $f(n) = b_k n^k + b_{k-1} n^{k-1} + \ldots + b_1 n + b_0$
    - Let $a_i = | b_i |$
  - $f(n) \leq a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$

$$\leq \quad n^k \sum a_i \frac{n^i}{n^k} \quad \leq \quad n^k \sum a_i \quad \leq \quad cn^k$$

Big-Oh Example

- Example: the function n^2 is not $O(n)$
  - $n^2 \leq cn$
  - $n \leq c$
  - The above inequality cannot be satisfied since c must be a constant



More Big-Oh Examples

- *7n-2*

  7n-2 is O(n)

  need c > 0 and $n_0 \geq 1$ such that 7n-2 $\leq$ c•n for n $\geq n_0$

  this is true for c = 7 and $n_0 = 1$

- *$3n^3 + 20n^2 + 5$*

  $3n^3 + 20n^2 + 5$ is $O(n^3)$

  need c > 0 and $n_0 \geq 1$ such that $3n^3 + 20n^2 + 5 \leq c \cdot n^3$ for n $\geq n_0$

  this is true for c = 4 and $n_0 = 21$

- *3 log n + 5*

  3 log n + 5 is O(log n)

  need c > 0 and $n_0 \geq 1$ such that 3 log n + 5 $\leq$ c•log n for n $\geq n_0$

  this is true for c = 8 and $n_0 = 2$

Big-Oh Rules

- If is $f(n)$ a polynomial of degree d, then $f(n)$ is $O(n^d)$, i.e.,
  1. Drop lower-order terms
  2. Drop constant factors
- Use the smallest possible class of functions
  - Say "$2n$ is $O(n)$" instead of "$2n$ is $O(n^2)$"
- Use the simplest expression of the class
  - Say "$3n + 5$ is $O(n)$" instead of "$3n + 5$ is $O(3n)$"

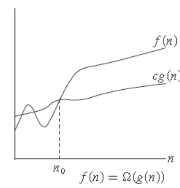
Big-Oh and Growth Rate

- The big-Oh notation gives an upper bound on the growth rate of a function
- The statement "***f***(***n***) is ***O***(***g***(***n***))" means that the growth rate of ***f***(***n***) is no more than the growth rate of ***g***(***n***)
- We can use the big-Oh notation to rank functions according to their growth rate

	f(***n***) is ***O***(***g***(***n***))	***g***(***n***) is ***O***(***f***(***n***))
g(***n***) grows more	Yes	No
f(***n***) grows more	No	Yes
Same growth	Yes	Yes

Lower Bound Notation

- We say Insertion Sort's run time is $\Omega(n)$
- In general a function
  - f(n) is $\Omega(g(n))$ if $\exists$ positive constants c and n_0 such that $0 \leq c \cdot g(n) \leq f(n) \;\; \forall\, n \geq n_0$
- Proof:
  - Suppose run time is an + b
    - Assume a and b are positive (what if b is negative?)
  - an $\leq$ an + b

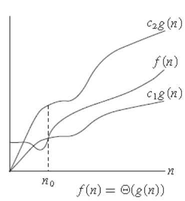


Asymptotic Tight Bound

- A function f(n) is $\Theta(g(n))$ if $\exists$ positive constants c_1, c_2, and n_0 such that

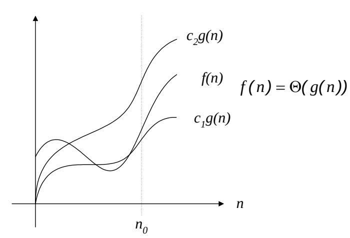
  $c_1\, g(n) \leq f(n) \leq c_2\, g(n) \;\; \forall\, n \geq n_0$

- Theorem
  - $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ AND $f(n) = \Omega(g(n))$



Asymptotic Tight Bound

- The definition of $\Theta(g(n))$ requires that every member be asymptotically nonnegative.



Asymptotic Tight Bound

EXAMPLE:

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

$$c_1 n^2 \le \frac{n^2}{2} - 3n \le c_2 n^2 \quad for \;\; all \;\; n \ge n_0$$

$$\frac{n^2}{14} \le \frac{n^2}{2} - 3n \le \frac{n^2}{2} \;\; if \;\; n>7$$

Asymptotic Tight Bound

$6n^3 \ne \Theta(n^2)$ ***Why ?***

$f(n) = an^2 + bn + c,\ a, b, c$ constants, $a>0.$

$\Rightarrow f(n)=\Theta(n^2).$

- In general,

  $p(n) = \sum_{i=0}^{d} a_i n^i$ where a_i are constant with $a_d > 0.$

  Then $P(n) = \Theta(n^d).$

o-notation (little-oh)

- An upper bound that is not asymptotically tight .
- $f(n) = o(g(n)) \Leftrightarrow lim_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$
- $o(g(n)) = \{ f(n) | \forall c, \exists n_0 \;\; \forall n > n_0, 0 \le f(n) \le cg(n)\}$

  All
- $2n^2 = O(n^2)$ $\qquad$ $2n^2 \ne o(n^2)$

  $2n = O(n^2)$ $\qquad$ $2n = o(n^2)$

ω-notation (little-omega)

- An lower bound that is not asymptotically tight .

  All
- $\omega(g(n)) = \{ f(n) | \forall c, \exists n_0 \;\; \forall n > n_0, 0 \le cg(n) \le f(n)\}$
- $f(n) = \omega(g(n)) \Leftrightarrow lim_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty$
- $\dfrac{n^2}{2} = \omega(n)$

  $\dfrac{n^2}{2} \ne \omega(n^2)$

Example Uses of the Relatives of Big-Oh

- ***$5n^2$ is $\Omega(n^2)$***
  f(n) is $\Omega(g(n))$ if there is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$
  let $c = 5$ and $n_0 = 1$
- ***$5n^2$ is $\Omega(n)$***
  f(n) is $\Omega(g(n))$ if there is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$
  let $c = 1$ and $n_0 = 1$
- ***$5n^2$ is $\Theta(n^2)$***
  f(n) is $\Theta(g(n))$ if it is $\Omega(n^2)$ and $O(n^2)$. We have already seen the former, for the latter recall that f(n) is $O(g(n))$ if there is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \leq c \cdot g(n)$ for $n \geq n_0$
  Let $c = 5$ and $n_0 = 1$

L3.13

Analysis of Algorithms

- Worst case $O(n)$
  - Provides an upper bound on running time
  - An absolute guarantee
- Average case $\Theta(n)$
  - Provides the expected running time
  - Very useful, but treat with care: what is "average"?
    - Random (equally likely) inputs
    - Real-life inputs
- Best case $\Omega(n)$
  - Not useful
  - Cannot too optimistic

L3.14

Relational properties

- Transitivity

$$f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$$
$$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$
$$f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$$
$$f(n) = o(g(n)) \wedge g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$$
$$f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$$

- Reflexivity

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

- Symmetry

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

L3.15

- Transpose symmetry

$$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$$
$$f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$$

$$f(n) = O(g(n)) \approx a \leq b$$
$$f(n) = \Omega(g(n)) \approx a \geq b$$
$$f(n) = \Theta(g(n)) \approx a = b$$
$$f(n) = o(g(n)) \approx a < b$$
$$f(n) = \omega(g(n)) \approx a > b$$

L3.16

3.1.4 from Textbook (page 53)

- Is $2^{n+1} = O(2^n)$?

$2^{n+1} = O(2^n)$ because $2^{n+1} = 2 * 2^n = O(2^n)$.

$2^{n+1} =< c2^n$ When c = 2, $n_0 = 1$, it satisify the inequality.

- Is $2^{2n} = O(2^n)$?

Suppose $2^{2n} = O(2^n)$ Then there exists a constant c such that for n beyond some n_0,

$2^{2n} <= c\ 2^n$. Dividing both sides by 2^n, we get $2^n < c$. There's no values for c and n_0 that can make this true, so the hypothesis is false and 2^{2n} != $O(2^n)$.

L3.17