Design and Analysis of Algorithms

Lecture 3 : Growth of Function

Instructor: Dr. G P Gupta

Growth of Function

- **What is Rate of Growth?**
  - The rate at which the running time increases as a function of input is called *rate of growth*.

Growth of Function cont..

- ***Why the order of growth of the running time of an algorithm ?***
  - gives a simple characterization of the algorithm's efficiency
  - allows us to compare the ***relative performance*** of alternative algorithms.

- **asymptotic efficiency of algorithms :**
  - we are concerned with ***how the running time of an algorithm increases with the size of the input in the limit***, as the size of the input increases without bound.
  - Usually, an algorithm that is asymptotically more efficient will be the best choice for all but very small inputs.

Rate of Growth

- Consider the example of buying *elephants* and *goldfish:*
  Cost: cost_of_elephants + cost_of_goldfish
  Cost ~ cost_of_elephants (approximation)

- The low order terms in a function are relatively insignificant for **large** *n*

$$n^4 + 100n^2 + 10n + 50 \sim n^4$$

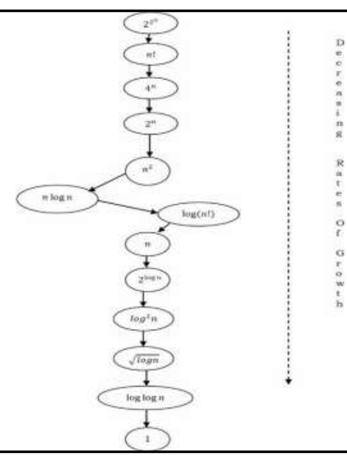
i.e., we say that $n^4 + 100n^2 + 10n + 50$ and n^4 *have the same* ***rate of growth***

Growth of Functions

- Although we can sometimes determine the exact running time of an algorithm, the ***extra precision is not usually worth the effort of computing it.***

- For large inputs, the ***multiplicative constants and lower order terms*** of an exact running time are ***dominated by the effects of the input size itself.***

Commonly Used Rates of Growth



Growth of Function cont..

Time Complexity	Name	Example
1	Constant	Adding an element to the front of a linked list
$logn$	Logarithmic	Finding an element in a sorted array
n	Linear	Finding an element in an unsorted array
$nlogn$	Linear Logarithmic	Sorting n items by 'divide-and-conquer' - Mergesort
n^2	Quadratic	Shortest path between two nodes in a graph
n^3	Cubic	Matrix Multiplication
2^n	Exponential	The Towers of Hanoi problem

L1.7

Asymptotic Analysis

- To compare two algorithms with **running times *f(n)*** and *g(n)*,
- we need a **rough measure** that characterizes **how fast each function grows.**

- *Hint:* use ***rate of growth***

- ***Compare functions in the limit***, that is, **asymptotically!** (i.e., for large values of *n*)

Asymptotic Notation

➢ asymptotic running time of an algorithm are defined in terms of functions whose domains are ***the set of natural numbers***

$$\mathbf{N} = \{0, 1, 2, ...\}$$

O-notation

- For a given function $g(n)$, we denote by $O(g(n))$ the set of functions

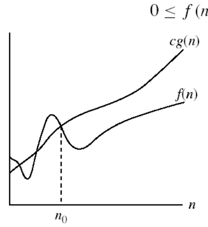
$$O(g(n)) = \left\{ \begin{array}{l} f(n): \text{there exist positive constants } c \text{ and } n_0 \text{ s.t.} \\ 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

- **We use *O*-notation to give an *asymptotic upper bound* of a function, to within a constant factor.**
  - $f(n) = O(g(n))$ means that there existes some constant ***c*** s.t. $f(n)$ is always $\le cg(n)$ for large enough *n*.

Asymptotic notations

- *O-notation*

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



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

Examples

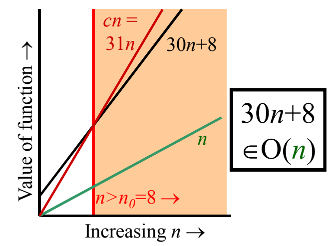
- Show that 30n+8 is O(n).

Examples cont..

- Show that $30n+8$ is $O(n)$.
  - Show $\exists c,n_0$: $30n+8 \le cn$, $\forall n>n_0$.
    - Let $c=31$, $n_0=8$. Assume $n>n_0=8$. Then $cn = 31n = 30n + n > 30n+8$, so $30n+8 < cn$.

Big-O example, graphically

- Note $30n+8$ isn't less than n *anywhere* ($n>0$).
- It isn't even less than $31n$ *everywhere*.
- But it *is* less than $31n$ everywhere to the right of $n=8$.



Example

$$f(n) = 3n^2 - 100n + 6$$

Example

$$3n^2 - 100n + 6 = O(n^2) \quad \text{since for } c = 3,\ 3n^2 > 3n^2 - 100n + 6$$

Ω-Omega notation

- For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions

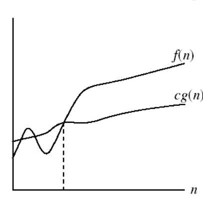
$$\Omega(g(n)) = \left\{ \begin{array}{l} f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ s.t.} \\ 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

- We use *Ω*-notation to give an ***asymptotic lower bound*** on a function, to within a constant factor.
  - $f(n) = \Omega(g(n))$ means that there exists some constant c s.t. $f(n)$ is always $\ge cg(n)$ for large enough n.

Asymptotic notations (cont.)

- 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 $n \ge n_0\}$.



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

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

18

Examples

- $5n^2 = \Omega(n)$

Examples cont..

- $5n^2 = \Omega(n)$

  $\exists\ c,\ n_0$ such that: $0 \le cn \le 5n^2 \Rightarrow cn \le 5n^2 \Rightarrow c = 1$ and $n_0 = 1$

- $100n + 5 \neq \Omega(n^2)$

  $\exists$ c, n_0 such that: $0 \le cn^2 \le 100n + 5$

  $100n + 5 \le 100n + 5n\ (\forall\ n \ge 1) = 105n$

  $cn^2 \le 105n \Rightarrow n(cn - 105) \le 0$

  Since n is positive $\Rightarrow cn - 105 \le 0 \Rightarrow n \le 105/c$

  $\Rightarrow$ contradiction: n cannot be smaller than a constant

- $n = \Omega(2n)$, $n^3 = \Omega(n^2)$, $n = \Omega(\log n)$

Θ-Theta notation

- For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions

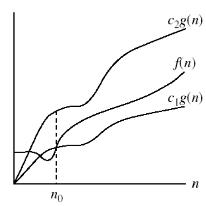
$$\Theta(g(n)) = \left\{ \begin{array}{l} f(n) : \text{there exist positive constants } c_1, c_2, \text{and } n_0 \text{ s.t.} \\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

- A function $f(n)$ belongs to the set $\Theta(g(n))$ if there exist positive constants c_1 and c_2 such that it can be "sand- wiched" between $c_1 g(n)$ and $c_2 g(n)$ or sufficiently large n.

- $f(n) = \Theta(g(n))$ means that there exists some constant c_1 and c_2 s.t. $c_1 g(n) \le f(n) \le c_2 g(n)$ for large enough n.

Asymptotic notations (cont.)

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



Θ(g(n)) is the set of functions with the same order of growth as g(n)

$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Examples

- $n^2/2 - n/2 = \Theta(n^2)$

Examples cont..

- $n^2/2 - n/2 = \Theta(n^2)$

  - $\frac{1}{2} n^2 - \frac{1}{2} n \le \frac{1}{2} n^2\ \forall n \ge 0 \Rightarrow$ $\mathbf{c_2 = \frac{1}{2}}$

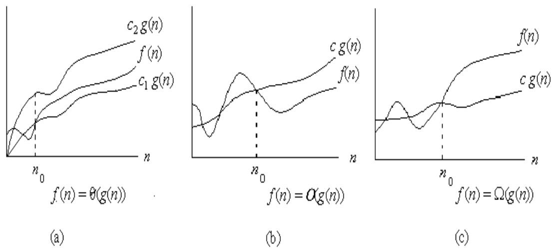
  - $\frac{1}{2} n^2 - \frac{1}{2} n \ge \frac{1}{2} n^2 - \frac{1}{2} n * \frac{1}{2} n\ (\forall n \ge 2) = \frac{1}{4} n^2 \Rightarrow$

    $\mathbf{c_1 = \frac{1}{4}}$

- $n \neq \Theta(n^2)$: $c_1 n^2 \le n \le c_2 n^2$

  $\Rightarrow$ only holds for: $n \le 1/c_1$

Asymptotic notation



Graphic examples of Θ, O, and Ω

Example 1

Show that $f(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$

We must find c_1 and c_2 such that

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

Dividing bothsides by n^2 yields

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

For $n_0 \ge 7$, $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

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

Example 2.

$f(n) = 3n^2 - 2n + 5 = \Theta(n^2)$

Because :

$3n^2 - 2n + 5 = \Omega(n^2)$

$3n^2 - 2n + 5 = O(n^2)$

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$

$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1$, $n^3 > 3n^2 - 100n + 6$ when $n > 3$

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$
$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1$, $n^3 > 3n^2 - 100n + 6$ when $n > 3$
$3n^2 - 100n + 6 \neq O(n)$ since for any c, $cn < 3n^2$ when $n > c$

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$
$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1$, $n^3 > 3n^2 - 100n + 6$ when $n > 3$
$3n^2 - 100n + 6 \neq O(n)$ since for any c, $cn < 3n^2$ when $n > c$

$3n^2 - 100n + 6 = \Omega(n^2)$ since for $c = 2$, $2n^2 < 3n^2 - 100n + 6$ when $n > 100$

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$
$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1$, $n^3 > 3n^2 - 100n + 6$ when $n > 3$
$3n^2 - 100n + 6 \neq O(n)$ since for any c, $cn < 3n^2$ when $n > c$
$3n^2 - 100n + 6 = \Omega(n^2)$ since for $c = 2$, $2n^2 < 3n^2 - 100n + 6$ when $n > 100$

$3n^2 - 100n + 6 \neq \Omega(n^3)$ since for $c = 3$, $3n^2 - 100n + 6 < n^3$ when $n > 3$

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$
$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1$, $n^3 > 3n^2 - 100n + 6$ when $n > 3$
$3n^2 - 100n + 6 \neq O(n)$ since for any c, $cn < 3n^2$ when $n > c$
$3n^2 - 100n + 6 = \Omega(n^2)$ since for $c = 2$, $2n^2 < 3n^2 - 100n + 6$ when $n > 100$
$3n^2 - 100n + 6 \neq \Omega(n^3)$ since for $c = 3$, $3n^2 - 100n + 6 < n^3$ when $n > 3$

$3n^2 - 100n + 6 = \Omega(n)$ since for any c, $cn < 3n^2 - 100n + 6$ when $n > 100$

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$
$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1$, $n^3 > 3n^2 - 100n + 6$ when $n > 3$
$3n^2 - 100n + 6 \neq O(n)$ since for any c, $cn < 3n^2$ when $n > c$
$3n^2 - 100n + 6 = \Omega(n^2)$ since for $c = 2$, $2n^2 < 3n^2 - 100n + 6$ when $n > 100$
$3n^2 - 100n + 6 \neq \Omega(n^3)$ since for $c = 3$, $3n^2 - 100n + 6 < n^3$ when $n > 3$
$3n^2 - 100n + 6 = \Omega(n)$ since for any c, $cn < 3n^2 - 100n + 6$ when $n > 100$

$3n^2 - 100n + 6 = \Theta(n^2)$ since both O and Ω apply.

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3$, $3n^2 > 3n^2 - 100n + 6$
$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1$, $n^3 > 3n^2 - 100n + 6$ when $n > 3$
$3n^2 - 100n + 6 \neq O(n)$ since for any c, $cn < 3n^2$ when $n > c$
$3n^2 - 100n + 6 = \Omega(n^2)$ since for $c = 2$, $2n^2 < 3n^2 - 100n + 6$ when $n > 100$
$3n^2 - 100n + 6 \neq \Omega(n^3)$ since for $c = 3$, $3n^2 - 100n + 6 < n^3$ when $n > 3$
$3n^2 - 100n + 6 = \Omega(n)$ since for any c, $cn < 3n^2 - 100n + 6$ when $n > 100$
$3n^2 - 100n + 6 = \Theta(n^2)$ since both O and Ω apply.

$3n^2 - 100n + 6 \neq \Theta(n^3)$ since only O applies.

Example 3.

$3n^2 - 100n + 6 = O(n^2)$ since for $c = 3,\ 3n^2 > 3n^2 - 100n + 6$

$3n^2 - 100n + 6 = O(n^3)$ since for $c = 1,\ n^3 > 3n^2 - 100n + 6$ when $n > 3$

$3n^2 - 100n + 6 \neq O(n)$ since for any c, $cn < 3n^2$ when $n > c$

$3n^2 - 100n + 6 = \Omega(n^2)$ since for $c = 2,\ 2n^2 < 3n^2 - 100n + 6$ when $n > 100$

$3n^2 - 100n + 6 \neq \Omega(n^3)$ since for $c = 3,\ 3n^2 - 100n + 6 < n^3$ when $n > 3$

$3n^2 - 100n + 6 = \Omega(n)$ since for any c, $cn < 3n^2 - 100n + 6$ when $n > 100$

$3n^2 - 100n + 6 = \Theta(n^2)$ since both O and Ω apply.

$3n^2 - 100n + 6 \neq \Theta(n^3)$ since only O applies.

$3n^2 - 100n + 6 \neq \Theta(n)$ since only Ω applies.

Standard notations and common functions

- Floors and ceilings

$$x - 1 < \lfloor x \rfloor \le x \le \lceil x \rceil < x + 1$$

Standard notations and common functions

- Logarithms:

$$\lg n = \log_2 n$$
$$\ln n = \log_e n$$
$$\log^k n = (\log n)^k$$
$$\lg\lg n = \lg(\lg n)$$

Standard notations and common functions

- Logarithms:

For all real a>0, b>0, c>0, and n

$$a = b^{\log_b a}$$
$$\log_c(ab) = \log_c a + \log_c b$$
$$\log_b a^n = n \log_b a$$
$$\log_b a = \frac{\log_c a}{\log_c b}$$

Standard notations and common functions

- Logarithms:

$$\log_b(1/a) = -\log_b a$$
$$a^{\log_b c} = c^{\log_b a}$$
$$\log_b a = \frac{1}{\log_a b}$$

Standard notations and common functions

- Factorials

For $n \ge 0$ the Stirling approximation:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

$$n! = o(n^n)$$
$$n! = \omega(2^n)$$
$$\lg(n!) = \Theta(n \lg n)$$

More Examples …

- $n^4 + 100n^2 + 10n + 50$ is $O(n^4)$
- $10n^3 + 2n^2$ is $O(n^3)$
- $n^3 - n^2$ is $O(n^3)$
- constants
  - 10 is $O(1)$
  - 1273 is $O(1)$

Back to Our Example

Algorithm 1	**Cost**	*Algorithm 2*	**Cost**
arr[0] = 0;	c_1	for(i=0; i<N; i++)	c_2
arr[1] = 0;	c_1	arr[i] = 0;	c_1
arr[2] = 0;	c_1		
...			
arr[N-1] = 0;	c_1		
	-----------		-------------
$c_1+c_1+...+c_1 = c_1 \times N$		$(N+1) \times c_2 + N \times c_1 = (c_2 + c_1) \times N + c_2$	

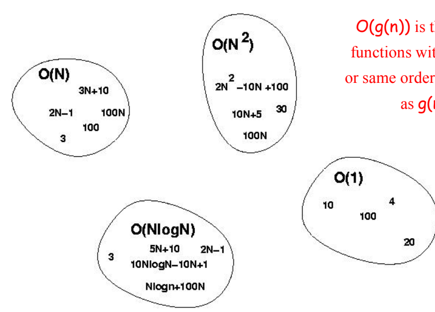
- Both algorithms are of the *same order*: *O(N)*

Example (cont'd)

Algorithm 3	*Cost*
sum = 0;	c_1
for(i=0; i<N; i++)	c_2
for(j=0; j<N; j++)	c_2
sum += arr[i][j];	c_3

$c_1 + c_2 \times (N+1) + c_2 \times N \times (N+1) + c_3 \times N^2 = O(N^2)$

Big-O Visualization



$O(g(n))$ is the set of functions with smaller or same order of growth as $g(n)$

No Uniqueness

- ***There is no unique set of values for n_0 and c in proving the asymptotic bounds***
- Prove that $100n + 5 = O(n^2)$
  - $100n + 5 \le 100n + n = 101n \le 101n^2$ for all $n \ge 5$

    $n_0 = 5$ and $c = 101$ is a solution
  - $100n + 5 \le 100n + 5n = 105n \le 105n^2$ for all $n \ge 1$

    $n_0 = 1$ and $c = 105$ is also a solution
- **Must find SOME constants c and n_0 that satisfy the asymptotic notation relation**

Examples

- $6n^3 \neq \Theta(n^2)$: $c_1 n^2 \le 6n^3 \le c_2 n^2$

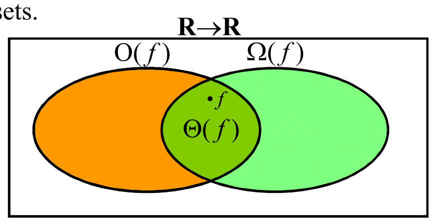
  $\Rightarrow$ only holds for: $n \le c_2 / 6$

- $n \neq \Theta(\log n)$: $c_1 \log n \le n \le c_2 \log n$

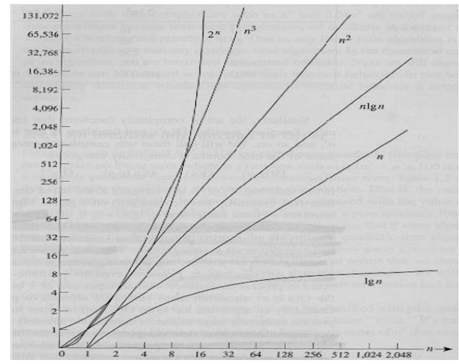
  $\Rightarrow c_2 \ge n/\log n, \forall n \ge n_0$ – impossible

Relations Between Different Sets

- Subset relations between order-of-growth sets.



Common orders of magnitude



Common orders of magnitude

Table 1.4 Execution times for algorithms with the given time complexities

n	$f(n) = \lg n$	$f(n) = n$	$f(n) = n \lg n$	$f(n) = n^2$	$f(n) = n^3$	$f(n) = 2^n$
10	0.003 μs*	0.01 μs	0.033 μs	0.1 μs	1 μs	1 μs
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	8 μs	1 ms†
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	27 μs	1 s
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	64 μs	18.3 min
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	125 μs	13 days
10^2	0.007 μs	0.10 μs	0.664 μs	10 μs	1 ms	4×10^{13} years
10^3	0.010 μs	1.00 μs	9.966 μs	1 ms	1 s	
10^4	0.013 μs	10 μs	130 μs	100 ms	16.7 min	
10^5	0.017 μs	0.10 ms	1.67 ms	10 s	11.6 days	
10^6	0.020 μs	1 ms	19.93 ms	16.7 min	31.7 years	
10^7	0.023 μs	0.01 s	0.23 s	1.16 days	31,709 years	
10^8	0.027 μs	0.10 s	2.66 s	115.7 days	3.17×10^7 years	
10^9	0.030 μs	1 s	29.90 s	31.7 years		

*1 μs = 10^{-6} second.
†1 ms = 10^{-3} second.

Logarithms and properties

- In algorithm analysis we often use the notation "log n" without specifying the base

Binary logarithm $\lg n = \log_2 n$

Natural logarithm $\ln n = \log_e n$

$\lg^k n = (\lg n)^k$

$\lg \lg n = \lg(\lg n)$

$\log x^y = y \log x$

$\log xy = \log x + \log y$

$\log \frac{x}{y} = \log x - \log y$

$a^{\log_b x} = x^{\log_b a}$

$\log_b x = \frac{\log_a x}{\log_a b}$

More Examples

- For each of the following pairs of functions, either f(n) is O(g(n)), f(n) is Ω(g(n)), or f(n) = Θ(g(n)). Determine which relationship is correct.
  - f(n) = log n^2; g(n) = log n + 5 — f(n) = Θ (g(n))
  - f(n) = n; g(n) = log n^2 — f(n) = Ω(g(n))
  - f(n) = log log n; g(n) = log n — f(n) = O(g(n))
  - f(n) = n; g(n) = $\log^2$ n — f(n) = Ω(g(n))
  - f(n) = n log n + n; g(n) = log n — f(n) = Ω(g(n))
  - f(n) = 10; g(n) = log 10 — f(n) = Θ(g(n))
  - f(n) = 2^n; g(n) = 10n^2 — f(n) = Ω(g(n))
  - f(n) = 2^n; g(n) = 3^n — f(n) = O(g(n))

Properties

- *Theorem:*

  f(n) = Θ(g(n)) ⇔ f = O(g(n)) and f = Ω(g(n))
- Transitivity:
  - f(n) = Θ(g(n)) and g(n) = Θ(h(n)) ⇒ f(n) = Θ(h(n))
  - Same for O and Ω
- Reflexivity:
  - f(n) = Θ(f(n))
  - Same for O and Ω
- Symmetry:
  - f(n) = Θ(g(n)) if and only if g(n) = Θ(f(n))
- Transpose symmetry:
  - f(n) = O(g(n)) if and only if g(n) = Ω(f(n))

Asymptotic Notations in Equations

- On the right-hand side
  - $\Theta(n^2)$ stands for some anonymous function in $\Theta(n^2)$

    $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means:

    There exists a function $f(n) \in \Theta(n)$ such that

    $2n^2 + 3n + 1 = 2n^2 + f(n)$
- On the left-hand side

  $2n^2 + \Theta(n) = \Theta(n^2)$

  No matter how the anonymous function is chosen on the left-hand side, there is a way to choose the anonymous function on the right-hand side to make the equation valid.

Common Summations

- Arithmetic series: $\sum_{k=1}^{n} k = 1 + 2 + ... + n = \frac{n(n+1)}{2}$
- Geometric series: $\sum_{k=0}^{n} x^k = 1 + x + x^2 + ... + x^n = \frac{x^{n+1} - 1}{x - 1} (x \neq 1)$
  - Special case: $|x| < 1$: $\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$
- Harmonic series: $\sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + ... + \frac{1}{n} \approx \ln n$
- Other important formulas: $\sum_{k=1}^{n} \lg k \approx n \lg n$

  $\sum_{k=1}^{n} k^p = 1^p + 2^p + ... + n^p \approx \frac{1}{p+1} n^{p+1}$

Mathematical Induction

- A powerful, rigorous technique for proving that a statement $S(n)$ is true for *every* natural number n, no matter how large.
- Proof:
  - **Basis step**: prove that the statement is true for $n = 1$
  - **Inductive step:** assume that $S(n)$ is true and prove that $S(n+1)$ is true for all $n \geq 1$
- Find case n "within" case $n+1$

Example

- Prove that: $2n + 1 \leq 2^n$ for all $n \geq 3$
- **Basis step:**
  - $n = 3$: $2 * 3 + 1 \leq 2^3 \Leftrightarrow 7 \leq 8$ TRUE
- **Inductive step:**
  - Assume inequality is true for n, and prove it for (n+1):

  $2n + 1 \leq 2^n$ must prove: $2(n + 1) + 1 \leq 2^{n+1}$

  $2(n + 1) + 1 = (2n + 1) + 2 \leq 2^n + 2 \leq$

  $\leq 2^n + 2^n = 2^{n+1}$, since $2 \leq 2^n$ for $n \geq 1$