CS146: Data Structures and Algorithms
Lecture 3

ORDER OF GROWTH

O, Ω, Θ

INSTRUCTOR: KATERINA POTIKA

CS SJSU

Measures of Algorithm Complexity

- **Worst-Case Running Time**: the longest time for any input size of **n**
  - an upper bound on running time for any input

- **Best-Case Running Time**: the shortest time for any input size of **n**
  - an lower bound on running time for any input

- **Average-Case Behavior**: the expected performance averaged over all possible inputs
  - it is generally better than worst case behavior, but sometimes it's roughly as bad as worst case

Order of Growth

- For very large input size n, it is the *rate of grow*, or *order of growth* that matters asymptotically
  - ignore the *lower-order terms*, since they are relatively insignificant for very large *n*
  - ignore *leading term's constant coefficients*, since they are not as important for the rate of growth in computational efficiency for very large *n*

- **Higher order** functions of *n* are normally considered **less efficient**

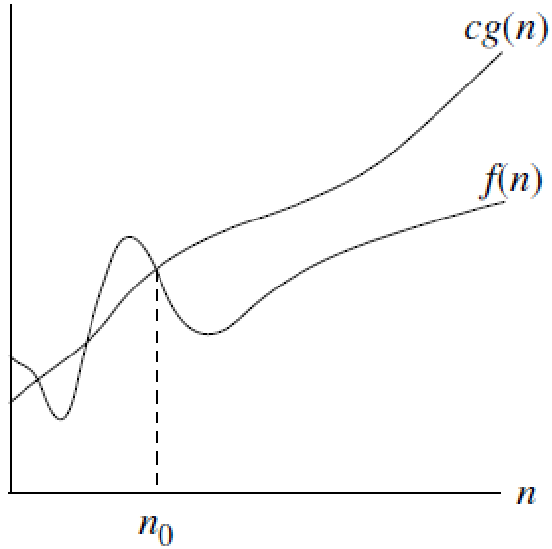
O - Notation

FORMAL DEFINITIONS

Big-O notation
(Upper Bound – Worst Case)

O-notation

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



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

Big-O notation
(Upper Bound – Worst Case)

- A mathematically formal way of ignoring constant factors, and looking only at the "shape" of the function

- $f(n)=O(g(n))$ should be considered as saying that "$f(n)$ is at most $g(n)$, up to constant factors".

- We usually will have $f(n)$ be the running time of an algorithm and $g(n)$ a nicely written function
- *Example*: The running time of insertion sort algorithm is $O(n^2)$

Big-O notation examples

Example: $2n^2 = O(n^3)$, with $c = 1$ and $n_0 = 2$.

Examples of functions in $O(n^2)$:

n^2
$n^2 + n$
$n^2 + 1000n$
$1000n^2 + 1000n$
Also,
n
$n/1000$
$n^{1.99999}$
$n^2/\lg\lg\lg n$

Big-O notation
(Upper Bound – Worst Case)

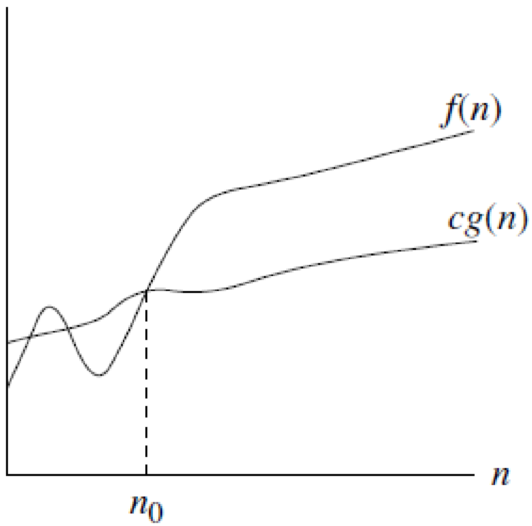
- ignore the multiplicative constants and the lower order terms, e.g.,
  - $n, n+1, n+80, 40n, n+\log n$ is $O(n)$
  - $n^{1.1} + \mathbf{10000000000n}$ is $O(n^{1.1})$
  - n^2 is $O(n^2)$
  - $\mathbf{3n^2 + 6n + \log n + 24.5}$ is $O(n^2)$

- $O(1) < O(\log n) < O((\log n)^3) < O(n) < O(n^2) < O(n^3) < O(n^{\log n}) < O(2^{\mathrm{sqrt}(n)}) < O(2^n) < O(n!) < O(n^n)$

- Constant < Logarithmic < Linear < Quadratic< Cubic < Polynomial < Factorial < Exponential

Ω-notation (Omega)
(Lower Bound – Best Case)

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



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

Ω-notation (Omega)

- We say Insertion Sort's run time $T(n)$ is $\Omega(n)$
  - Why?
- For example
  - the worst-case running time of insertion sort is $O(n^2)$, and
  - the best-case running time of insertion sort is $\Omega(n)$

Ω-notation (Omega)

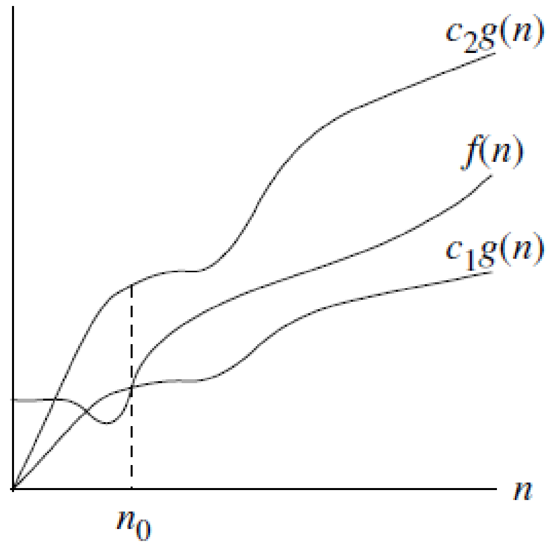
Example: $\sqrt{n} = \Omega(\lg n)$, with $c = 1$ and $n_0 = 16$.

Examples of functions in $\Omega(n^2)$:

n^2
$n^2 + n$
$n^2 - n$
$1000n^2 + 1000n$
$1000n^2 - 1000n$
Also,
n^3
$n^{2.00001}$
$n^2 \lg\lg\lg n$
2^{2^n}

Θ notation (Theta) (Tight Bound)

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



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

Θ notation (Theta)

- We say *g(n)* is an *asymptotic tight bound* for *f(n)*:

- **Theta notation**
  - **Θ(g(n))** means that as **n** → ∞, the execution time **f(n)** is at **most** $\mathbf{c_2.g(n)}$ and at **least** $\mathbf{c_1.g(n)}$ for some constants $\mathbf{c_1}$ and $\mathbf{c_2}$.

- f(n) = Θ(g(n)) if and only if
  - f(n) = O(g(n)) & f(n) = Ω(g(n))

Θ notation (Theta)

- **<u>Example1:</u>**
- Show that $6n^3 \neq \Theta(n^2)$
- Suppose for the purpose of contradiction that $\mathbf{c_2}$ and $\mathbf{n_0}$ exist such that $6n^3 \leq c_2 n^2$ for all $n \geq n_0$
  - Dividing by n^2 yields
    - $n \leq c_2/6$
  - which cannot possibly hold for arbitrary large **n**, since $\mathbf{c_2}$ is constant
  - Also, $\lim_{n\to\infty}[6n^3 / n^2] = \lim_{n\to\infty}[6n] = \infty$, *which is not a non-zero constant*

o-notation

We say $g(n)$ is an *upper bound* for $f(n)$ that is *not* asymptotically tight (strictly).

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

Another view, probably easier to use: $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} = 0.$

$n^{1.9999} = o(n^2)$
$n^2/\lg n = o(n^2)$
$n^2 \neq o(n^2)$ (just like $2 \not< 2$)
$n^2/1000 \neq o(n^2)$

O() versus o()

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

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

Thus o*(f(n))* is a weakened O*(f(n))*.

For example: $n^2 = O(n^2)$

$$n^2 \neq o(n^2)$$

$$n^2 = O(n^3)$$

$$n^2 = o(n^3)$$

ω-notation

We say $g(n)$ is a *lower bound* for $f(n)$ that is not asymptotically tight.

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

Another view, again, probably easier to use: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

$n^{2.0001} = \omega(n^2)$
$n^2 \lg n = \omega(n^2)$
$n^2 \neq \omega(n^2)$

Properties

- Transitivity

$$f(n) = \Theta(g(n)) \ \& \ g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$$
$$f(n) = O(g(n)) \ \& \ g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$
$$f(n) = \Omega(g(n)) \ \& \ g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$$

- Symmetry

$f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$

- Transpose Symmetry

$f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$

Example

f(n)	***g(n)***	**Is**	**Solution**
$5n^2 + 100n$	$3n^2 + 2$	$f = \Theta(g)$?	$f = \Theta(n^2)$, $n^2 = \Theta(g)$ $\Rightarrow f = \Theta(g)$

Some Common Name for Complexity

O(1)	Constant time
O(log n)	Logarithmic time
$O(\log^2 n)$	Log-squared time
O(n)	Linear time
$O(n^2)$	Quadratic time
$O(n^3)$	Cubic time
$O(n^i)$ for some i	Polynomial time
$O(2^n)$	Exponential time

Growth Rates of some Functions

$$O(\log n) < O(\log^2 n) < O(\sqrt{n}) < O(n)$$
$$< O(n \log n) < O(n \log^2 n) < O(n^{1.5}) < O(n^2)$$
$$< O(n^3) < O(n^4)$$

Polynomial Functions

$$O(n^c) = O(2^{c \log n}) \text{ for any constant } c$$
$$< O(n^{\log n}) = O(2^{\log^2 n})$$
$$< O(2^n) < O(3^n) < O(4^n)$$
$$< O(n!) < O(n^n)$$

Exponential Functions

A Survey of Common Running Times

THIS SECTION

Why it matters

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	4 sec
$n = 30$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	18 min	10^{25} years
$n = 50$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	11 min	36 years	very long
$n = 100$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	$<$ 1 sec	$<$ 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	$<$ 1 sec	$<$ 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	$<$ 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Linear Time: O(n)

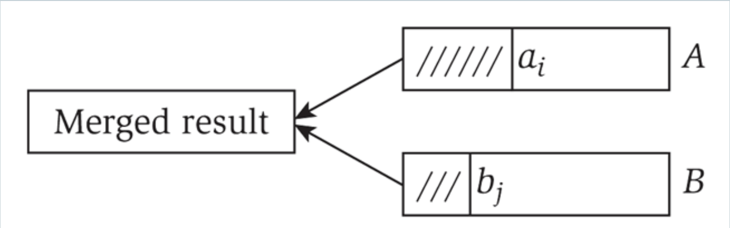
- Linear time. Running time is at most a constant factor times the size of the input.

- Computing the maximum. Compute maximum of n numbers $a_1, \ldots, a_n$.

```
max = a1
for i = 2 to n {
   if (ai > max)
      max ← ai
}
```

Linear Time: O(n)

- Merge. Combine two sorted lists $A = a_1, a_2, \ldots, a_n$ with $B = b_1, b_2, \ldots, b_n$ into sorted whole.




```
i = 1, j = 1
while (both lists are nonempty) {
   if (a_i ≤ b_j) append a_i to output list and increment i
   else(a_i ≤ b_j)append b_j to output list and increment j
}
append remainder of nonempty list to output list
```

- Claim. Merging two lists of size n takes O(n) time.
- Pf. After each comparison, the length of output list increases by 1.

O(n log n) Time

- O(n log n) time. Arises in divide-and-conquer algorithms.

  also referred to as linearithmic time

- Sorting. Mergesort and heapsort are sorting algorithms that perform O(n log n) comparisons.

- Largest empty interval. Given n time-stamps $x_1, \ldots, x_n$ on which copies of a file arrive at a server, what is largest interval of time when no copies of the file arrive?

- O(n log n) solution. Sort the time-stamps. Scan the sorted list in order, identifying the maximum gap between successive time-stamps.

Quadratic Time: $O(n^2)$

- Quadratic time. Enumerate all pairs of elements.
- Closest pair of points. Given a list of n points in the plane $(x_1, y_1), \ldots, (x_n, y_n)$, find the pair that is closest.
- $O(n^2)$ solution. Try all pairs of points.

```
min ← (x1 - x2)² + (y1 - y2)²
for i = 1 to n {
   for j = i+1 to n {
      d ← (xi - xj)² + (yi - yj)²
      if (d < min)
         min ← d
   }
}
```

← don't need to take square roots

- Remark. $\Omega(n^2)$ seems inevitable, but this is just an illusion.

Cubic Time: $O(n^3)$

- Cubic time. Enumerate all triples of elements.

- Set disjointness. Given n sets $S_1, \ldots, S_n$ each of which is a subset of 1, 2, ..., n, is there some pair of these which are disjoint?

- $O(n^3)$ solution. For each pairs of sets, determine if they are disjoint.

```
foreach set Si {
   foreach other set Sj {
      foreach element p of Si {
         determine whether p also belongs to Sj
      }
      if (no element of Si belongs to Sj)
         report that Si and Sj are disjoint
   }
}
```

Polynomial Time: $O(n^k)$ Time

- Independent set of size k. Given a graph, are there k nodes such that no two are joined by an edge?

k is a constant

- $O(n^k)$ solution. Enumerate all subsets of k nodes.

```
foreach subset S of k nodes {
   check whether S in an independent set
   if (S is an independent set)
      report S is an independent set
   }
}
```

  - Check whether S is an independent set = $O(k^2)$.
  - Number of k element subsets = $\binom{n}{k} = \frac{n\,(n-1)\,(n-2)\cdots(n-k+1)}{k\,(k-1)\,(k-2)\cdots(2)\,(1)} \leq \frac{n^k}{k!}$
  - $O(k^2\, n^k / k!) = O(n^k)$.

poly-time for k=17,
but not practical

Exponential Time

- Independent set. Given a graph, what is maximum size of an independent set?

- $O(n^2 2^n)$ solution. Enumerate all subsets.

```
S* ← ϕ
foreach subset S of nodes {
   check whether S in an independent set
   if (S is largest independent set seen so far)
      update S* ← S
   }
}
```

Useful facts

Proofs by Counterexample & Contradiction

- There are several ways to prove a theorem:
  - **Counterexample:**
    - By providing an example of in which the theorem **does not hold**, you prove the theory to be **false**.
    - For example: All multiples of 5 are even. However 3x5 is 15, which is odd. The theorem is false.
  - **Contradiction:**
    - Assume the theorem to be **true**. If the assumption **implies** that some known property is **false**, then the theorem **CANNOT** be **true**.

Floors & Ceilings

- For any **real** number **x**, we denote the **greatest integer** *less than or equal* to **x** by $\lfloor x \rfloor$
  - read “the floor of x”
- For any **real** number **x**, we denote the **least integer** *greater than or equal* to **x** by $\lceil x \rceil$
  - read “the ceiling of x”
- For all real **x**, (example for x=4.2)
  - $x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$
- For any integer **n** ,
  - $\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$

Polynomials

- Given a positive integer d, a **polynomial** in n of degree d is a function $P(n)$ of the form
  - $P(n) =$ $$\sum_{i=0}^{d} a_i n^i$$
  - where $a_0, a_1, \ldots, a_d$ are coefficient of the polynomial
  - $a_d \neq 0$
- A polynomial is **asymptotically positive** iff $a_d > 0$
  - Also $P(n) = \Theta(n^d)$

Exponents

- $x^0 = 1$ $x^1 = x$ $x^{-1} = 1/x$
- $x^a \cdot x^b = x^{a+b}$
- $x^a / x^b = x^{a-b}$
- $(x^a)^b = (x^b)^a = x^{ab}$
- $x^n + x^n = 2x^n \neq x^{2n}$
- $2^n + 2^n = 2.2^n = 2^{n+1}$

Logarithms (1)

- In computer science, all logarithms are to **base 2** unless specified otherwise
- $x^a = b$ iff $\log_x(b) = a$
- $\lg(n) = \log_2(n)$
- $\ln(n) = \log_e(n)$
- $\lg^k(n) = (\lg(n))^k$
- $\log_a(b) = \log_c(b) / \log_c(a)$; $c > 0$
- $\lg(ab) = \lg(a) + \lg(b)$
- $\lg(a/b) = \lg(a) - \lg(b)$
- $\lg(a^b) = b \cdot \lg(a)$

Logarithms (2)

- $a = b^{\log_b(a)}$
- $a^{\log_b(n)} = n^{\log_b(a)}$
- $\lg(1/a) = -\lg(a)$
- $\log_b(a) = 1/\log_a(b)$
- $\lg(n) < n$ for all $n > 0$
- $\log_a(a) = 1$
- $\lg(1) = 0$, $\lg(2) = 1$, $\lg(1024=2^{10}) = 10$
- $\lg(1048576=2^{20}) = 20$

log*n

- called ***log star of n***
- is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1
- Examples:

lg*2=1

lg*4=2

lg*16=3

lg*65536=4

Harmonic number

- $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k} = lnn + O(1)$

Summations

- $\sum_{k=1}^{n} k = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} = \Theta(n^2)$

- $\sum_{k=1}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1}-1}{x-1}$

Summations (cont.)

$$\sum_{k=1}^{n}(a_k - a_{k-1}) = a_n - a_0$$

$$\sum_{k=1}^{n}(a_k - a_{k+1}) = a_0 - a_n$$

Series

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$$

$$\sum_{i=0}^{N} A^i = \frac{A^{N+1} - 1}{A - 1}$$

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2} \approx \frac{N^2}{2}$$

if $0 < A < 1$:

$$\sum_{i=0}^{N} A^i \leq \frac{1}{1 - A}$$

$$\sum_{i=0}^{\infty} A^i = \frac{1}{1 - A}$$

Factorials

- **n!** ("n factorial") is defined for integers n ≥ 0 as

n! = $\begin{cases} 1, & if\ n = 0 \\ n * (n-1)!, & if\ n > 0 \end{cases}$

- n! = 1 . 2 .3 … n
- n! < n^n for n ≥ 2
- Stirling's approximation

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

Useful Facts

- For $a > 0$, $b > 0$, $\lim_{n\to\infty} (\lg^a n / n^b) = 0$, so $\lg^a n = o(n^b)$, and $n^b = \omega(\lg^a n)$
  - Prove using L'Hospital's rule and induction

- $\lg(n!) = \Theta(n \lg n)$

Examples

A	B
$5n^2 + 100n$	$3n^2 + 2$
$\log_3(n^2)$	$\log_2(n^3)$
$n^{\lg 4}$	$3^{\lg n}$
$\lg^2 n$	$n^{1/2}$

Examples

	A	B	
■	$5n^2 + 100n$	$3n^2 + 2$	$A \in \Theta(B)$
	$A \in \Theta(n^2), n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$		
■	$\log_3(n^2)$	$\log_2(n^3)$	
■	$n^{\lg 4}$	$3^{\lg n}$	
■	$\lg^2 n$	$n^{1/2}$	

Examples

A B

- $5n^2 + 100n$ $3n^2 + 2$ $A \in \Theta(B)$

  $A \in \Theta(n^2), n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$

- $\log_3(n^2)$ $\log_2(n^3)$ $A \in \Theta(B)$

  $\log_b a = \log_c a / \log_c b$; $A = 2\lg n / \lg 3$, $B = 3\lg n$, $A/B = 2/(3\lg 3)$

- $n^{\lg 4}$ $3^{\lg n}$

- $\lg^2 n$ $n^{1/2}$

Examples

A | B |
---|---|---

- $5n^2 + 100n$ $\quad$ $3n^2 + 2$ $\quad$ $A \in \Theta(B)$

  $A \in \Theta(n^2),\ n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$

- $\log_3(n^2)$ $\quad$ $\log_2(n^3)$ $\quad$ $A \in \Theta(B)$

  $\log_b a = \log_c a / \log_c b$; $A = 2\lg n / \lg 3$, $B = 3\lg n$, $A/B = 2/(3\lg 3)$

- $n^{\lg 4}$ $\quad$ $3^{\lg n}$ $\quad$ $A \in \omega(B)$

  $a^{\log b} = b^{\log a}$; $B = 3^{\lg n} = n^{\lg 3}$; $A/B = n^{\lg(4/3)} \to \infty$ as $n \to \infty$

- $\lg^2 n$ $\quad$ $n^{1/2}$

Examples

A B

- $5n^2 + 100n$ $\quad 3n^2 + 2$ $\quad A \in \Theta(B)$
  $A \in \Theta(n^2),\ n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$
- $\log_3(n^2)$ $\quad \log_2(n^3)$ $\quad A \in \Theta(B)$
  $\log_b a = \log_c a \,/\, \log_c b$; $A = 2\lg n \,/\, \lg 3$, $B = 3\lg n$, $A/B = 2/(3\lg 3)$
- $n^{\lg 4}$ $\quad 3^{\lg n}$ $\quad A \in \omega(B)$
  $a^{\log b} = b^{\log a}$; $B = 3^{\lg n} = n^{\lg 3}$; $A/B = n^{\lg(4/3)} \rightarrow \infty$ as $n \rightarrow \infty$
- $\lg^2 n$ $\quad n^{1/2}$ $\quad A \in o(B)$
  $\lim_{n\rightarrow\infty} (\lg^a n \,/\, n^b) = 0$ (here $a = 2$ and $b = 1/2$) $\Rightarrow A \in o(B)$

True or false

- $10\,f(n) + 10100 = O(f(n))$ True
- $f(n) + g(n) = \Theta(\min\{f(n), g(n)\})$ False
- $f(n) + g(n) = \Omega(\min\{f(n), g(n)\})$ True
- $f(n) + g(n) = O(\max\{f(n), g(n)\})$ True

exercises

f(n)	g(n)	Θ(g(n)	O(g(n))	o(g(n))	Ω(g(n))	ω(g(n))
2^n	$2^{n+2}+5$					
n^4	16^{lgn}					
5^{4n}	10^{2n}					
$n^{1/lgn}$	$n^{0.001}$					
$n!$	n^n					
n^{lgn}	2^n					

Self test

- Choose the correct order of the following functions by asymptotic growth rate (smaller to bigger)
- lgn , 2^{log3} , n^2 , $(\frac{3}{2})^n$, $n^{3/2}$, 2^{logn}

A. lgn, 2^{log3} , 2^{lgn}, n^2 , $n^{3/2}$, $(\frac{3}{2})^n$

B. 2^{log3}, lgn, 2^{lgn} , $n^{3/2}$, n^{2}, $(\frac{3}{2})^n$

C. 2^{log3}, lgn, 2^{lgn} , $n^{3/2}$, $(\frac{3}{2})^n$, n^2

D. 2^{log3}, lgn, $n^{\frac{3}{2}}$, n^2, 2^{lgn}, $\left(\frac{3}{2}\right)^n$