CS 2009
Design and Analysis of Algorithms

Google Classroom Code: su6b3ea

Lecture 4, 5 and 6:
Asymptotic Analysis

13th, 15th and 16th September, 2021

Farrukh Salim Shaikh

Efficiency of Algorithm

Revision

INTRODUCING...

ASYMPTOTIC ANALYSIS

Some guiding principles:

- we want some measure of runtime that's independent of hardware, programming language, memory layout, etc.
  - We want to reason about high-level algorithmic approaches rather than lower-level details
- we care about how the running time/number of operations *scales* with the size of the input (i.e. the runtime's *rate of growth*),
- Not concerned with small values of n, Concerned with VERY LARGE values of n.
- Asymptotic –refers to study of function f as n approaches infinity

Revision

Asymptotic Analysis (High Level Idea)

We'll express the asymptotic runtime of an algorithm using

BIG-O NOTATION

- We would say Multiplication **"runs in time $O(n^2)$"**
  - Informally, this means that the runtime "scales like" n^2
- What will be the complexity of a simple loop

```
for (i = 0, i < n, i++)
        A[i];
```

$$c_1 + c_2\,(n+1) + c_{3}.n + c_4.n$$

Asymptotic Analysis (High Level Idea)

Revision

BIG-O NOTATION

THE POINT OF ASYMPTOTIC NOTATION

suppress $\underbrace{\textbf{constant factors}}_{\textit{too system dependent}}$ **and** $\underbrace{\textbf{lower-order terms}}_{\textit{irrelevant for large inputs}}$

Example $f(n) = 2n^2 + 4n + 1$

$f(n) = O(n^2)$: 2 is constant, n^2 is the dominant term, and the term $4n + 1$ becomes insignificant as n grows larger.

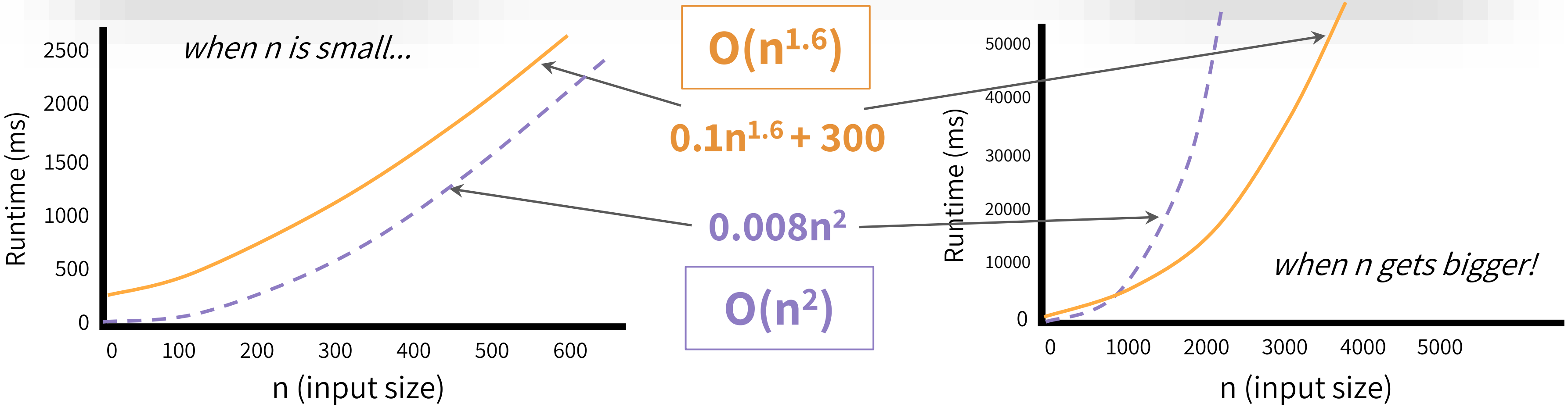
Asymptotic Analysis (High Level Idea)

Revision

THE POINT OF ASYMPTOTIC NOTATION

suppress constant factors and lower-order terms

too system dependent *irrelevant for large inputs*



Number of Function Executions for Various Times

Revision

		1 second	1 hour	1 month	1 century
Logarithmic	**$\log_2 n$**	10^{300000}			
Linear	**n**	10^6	$3.6 * 10^9$	$2.59 * 10^{12}$	$3.11 * 10^{15}$
n log n	**$n \log_2 n$**	62746	$1.3 * 10^8$	$7.1 * 10^{10}$	$6.7 * 10^{13}$
Ploynomial {e.g., $O(n^2)$, $O(n^3)$ etc.} {i.e., quadratic, cubic etc.}	**n^2**	10^3	$6 * 10^4$	$1.6 * 10^6$	$5.57 * 10^7$
	n^3	10^2	$1.53 * 10^3$	$1.37 * 10^4$	$1.46 * 10^5$
Expoential {e.g., $O(1.6^n)$, $O(2^n)$ etc.}	**2^n**	19	31	41	51
Factorial	**n!**	9	12	15	17

Number of Compuatations required for Various Input Sizes

Revision

	Input size: n	5	10	100	1000	10000
Constant	(1)	1	1	1	1	1
Logarithmic	log n	3	4	5	7	13
Linear	n	5	10	100	1000	10000
n log n	n log n	15	33	664	10^4	10^5
Ploynomial {e.g., $O(n^2)$, $O(n^3)$ etc.} {i.e., quadratic, cubic etc.}	n^2	25	100	10^4	10^6	10^8
	n^3	125	10^3	10^6	10^9	10^{12}
Expoential {e.g., $O(1.6^n)$, $O(2^n)$ etc.}	2^n	32	10^3	10^{30}	10^{300}	10^{3000}

To Do: For some insights, a graphic calculator can be used, e.g., desmos.com/calculator

Growth Rate Ranking of Function

Revision

		Number of Function Executions for Various Times				Number of computations required for given input sizes			
		$f(n) = 1\ \mu s$	1 Second	1 Hour	1 Month	Input size: n	5	10	10000
Logarithmic	**$\log_2 n$**		10^{300000}				3	4	13
Linear	**n**		10^6	$3.6 * 10^9$	$2.59 * 10^{12}$		5	10	10000
n log n	**$n \log_2 n$**		62746	$1.3 * 10^8$	$7.1 * 10^{10}$		15	33	105
Ploynomial {e.g., $O(n^2)$, $O(n^3)$ etc.} {i.e., quadratic, cubic etc.}	**n^2**		10^3	$6 * 10^4$	$1.6 * 10^6$		25	100	$1 * 10^8$
	n^3		10^2	$1.53 * 10^3$	$1.37 * 10^4$		125	$1 * 10^3$	$1 * 10^{12}$
Expoential {e.g., $O(1.6^n)$, $O(2^n)$ etc.}	**2^n**		19	31	41		32	$1 * 10^3$	$1 * 10^{3000}$

Runtime Analysis

There are a few different ways to analyze the runtime of an algorithm:

We'll mainly focus on worst case analysis since it tells us how fast the algorithm is on *any* kind of input

Worst-case analysis:
What is the runtime of the algorithm on the *worst* possible input?

Best-case analysis:
What is the runtime of the algorithm on the *best* possible input?

Average-case analysis:
What is the runtime of the algorithm on the *average* input?

Big-O Notation

Let f(n) & g(n) be functions defined on the positive integers.

What do we mean when we say "f(n) is O(g(n))"?

In Math

f(n) grows no faster than g(n)

or

g(n) is upper bound on f(n)

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

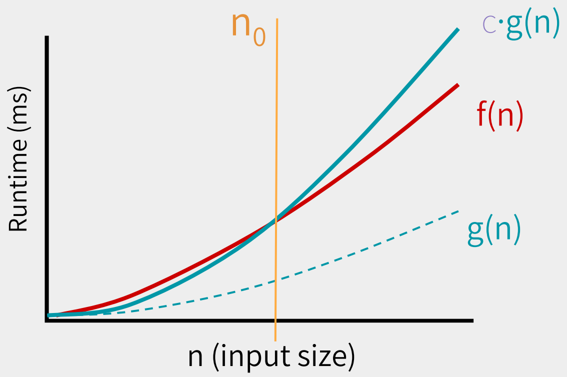
if and only if

there exists positive **constants**

c and **n_0** such that *for all $n \geq n_0$*

$f(n) \leq c \cdot g(n)$

In Picture



Big-O Notation

Let f(n) & g(n) be functions defined on the positive integers.

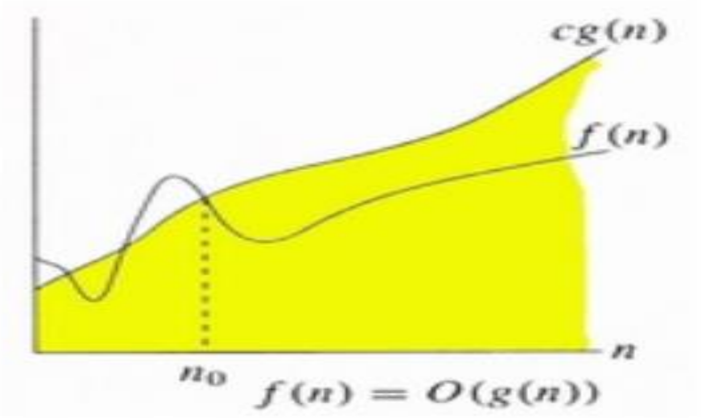
What do we mean when we say "f(n) is O(g(n))"?

In Math

f(n) = O(g(n))
if and only if
there exists positive **constants**
c and **n_0** such that *for all $n \geq n_0$*

$f(n) \leq c \cdot g(n)$

In Picture



Big-O Notation

Let f(n) & g(n) be functions defined on the positive integers.

What do we mean when we say "f(n) is O(g(n))"?

In Math

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

if and only if

there exists positive **constants** **c** and $\mathbf{n_0}$ such that *for all* $n \geq n_0$

$f(n) \leq c \cdot g(n)$

In Math

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

$\Leftrightarrow$

$\exists\ c, n_0 > 0$ s.t. $\forall\ n \geq n_0$,

$f(n) \leq c \cdot g(n)$

Proving Big-O Bounds

If you're ever asked to formally prove that f(n) is O(g(n)), use the *MATH* definition:

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

"if and only if" $\Leftrightarrow$ "for all"

$$\exists\ c,\ n_0 > 0 \ \text{ s.t. } \forall\ n \geq n_0,$$

must be constants! i.e. c & n_0 cannot depend on n!

$$f(n) \leq c \cdot g(n)$$

"such that"

"there exists"

- To **prove** f(n) = O(g(n)), you need to announce your c & n_0 up front!
  - Play around with the expressions to find appropriate choices of c & n_0 (positive constants)
  - Then you can write the proof! Here how to structure the start of the proof:

Proving Big-O Bounds: Example # 1 (Method # 1)

$$f(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists\ c, n_0 > 0 \text{ s.t. } \forall\ n \geq n_0,$$
$$f(n) \leq c \cdot g(n)$$

Prove that $3n^2 + 5n = O(n^2)$.

find a c & n_0 such that for all $n \geq n_0$:

$$3n^2 + 5n \leq c \bullet n^2$$

rearrange this inequality just to see things a bit more clearly:

$$5n \leq (c - 3) \bullet n^2$$

Now let's cancel out the n:

$$5 \leq (c - 3)\, n$$

Let's choose:

$c = 4$

$n_0 = 5$

(other choices work too!
e.g. $c = 5$, $n_0 = 4$
$c = 10$, $n_0 = 10$)

Proving Big-O Bounds: Example # 2 (Method # 2)

Prove that $f(n) = 3n^2 + 5n + 7 = O(n^2)$.

find a c & n_0 such that for all $n \geq n_0$:

$$3n^2 + 5n + 7 \leq c \bullet n^2$$

$3n^2 \leq 3n^2$ for $n \geq 0$

$5n \leq 5n^2$ for $n \geq 0$

$7 \leq 7n^2$ for $n \geq 1$

$3n^2 + 5n + 7 \leq 3n^2 + 5n^2 + 7n^2$ for $n \geq 1$

$3n^2 + 5n + 7 \leq 15n^2$ for $n \geq 1$

Proved that $f(n) = 3n^2 + 5n + 7 = O(n^2)$ [for $c = 15$, $n_0 = 1$]

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

$\Leftrightarrow$

$\exists\ c, n_0 > 0$ s.t. $\forall\ n \geq n_0$,

$f(n) \leq c \cdot g(n)$

Proving Big-O Bounds: Example # 2 (Method # 1)

Prove that $f(n) = 3n^2 + 5n + 7 = O(n^2)$.

find a c & n_0 such that for all $n \geq n_0$:

$$3n^2 + 5n + 7 \leq c \bullet n^2$$

Divide both sides by n^2, we get:

$3 n^2 / n^2 + \; 5n/ n^2 + \; 7/n^2 \leq \; c \bullet n^2 /n^2$

$3 + 5/n + 7/n^2 \leq c$

If we choose n_0 equal to 1 then we have value of c

$3 + 5 + 7 \leq c$

$c \leq 15$

$3n^2 + 5n + 7 \leq 15n^2$ for $n \geq 1$

Proved that $f(n) = 3n^2 + 5n + 7 = O(n^2)$ [for $c = 15$, $n_0 = 1$]

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

$\Leftrightarrow$

$\exists\ c, n_0 > 0$ s.t. $\forall\ n \geq n_0$,

$f(n) \leq c \cdot g(n)$

Proving Big-O Bounds: Example # 1 (Method # 2)

Prove that $f(n) = 3n^2 + 5n = O(n^2)$.

find a c & n_0 such that for all $n \geq n_0$:

$$3n^2 + 5n \leq c \bullet n^2$$

$3n^2 \leq 3n^2$ for $n \geq 0$

$5n \leq 5n^2$ for $n \geq 0$

$3n^2 + 5n \leq 3n^2 + 5n^2$ for $n \geq 0$

$3n^2 + 5n \leq 8n^2$ for $n \geq 0$

So $f(n) = 3n^2 + 5n = O(n^2)$ [for $c = 8$, $n_0 = 0$]

The c & n_0 are selected as positive constants, so:

Proved that $f(n) = 3n^2 + 5n = O(n^2)$ [for $c = 8$, $n_0 = 1$]

$$f(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists\ c, n_0 > 0 \text{ s.t. } \forall\ n \geq n_0,$$
$$f(n) \leq c \cdot g(n)$$

Proving Big-O Bounds: Example # 3 (Method # 2)

Show that $f(n) = 5n \log_2 n + 8n + 200 = O(n \log_2 n)$

$$f(n) = O(g(n)) \Leftrightarrow \exists\, c, n_0 > 0 \text{ s.t. } \forall\, n \geq n_0,\ f(n) \leq c \cdot g(n)$$

find a c & n_0 such that for all $n \geq n_0$:

$$5n \log_2 n + 8n + 200 \leq c \bullet n \log_2 n$$

$$5n \log_2 n + 8n + 200 \leq 5n \log_2 n + 8n \log_2 n + 200n \log_2 n \quad \text{for } n \geq 2$$

$$5n \log_2 n + 8n + 200 \leq 213n \log_2 n \qquad \text{for } n \geq 2$$

Thus

$f(n) = 5n \log_2 n + 8n + 200 = O(n \log_2 n)$ [for $c = 213$, $n_0 = 2$]

Disproving Big-O Bounds

If you're ever asked to formally disprove that T(n) is O(f(n)), use **proof by contradiction!**

This means you
need to show that
NO POSSIBLE
CHOICE of c & n_0
exists
such that the Big-O
definition holds

Disproving Big-O Bounds

Skip in Class

$f(n) = O(g(n))$
$\Leftrightarrow$
$\exists\, c, n_0 > 0$ s.t. $\forall\, n \geq n_0$,
$f(n) \leq c \cdot g(n)$

Prove that $3n^2 + 5n$ is *not* $O(n)$.

For sake of contradiction, assume that $3n^2 + 5n$ is $O(n)$. This means that there exists positive constants c & n_0 such that $3n^2 + 5n \leq c \cdot n$ for all $n \geq n_0$. Then, we would have the following:

$$3n^2 + 5n \leq c \cdot n$$
$$3n + 5 \leq c$$
$$n \leq (c - 5)/3$$

However, since $(c - 5)/3$ is a constant, we've arrived at a contradiction since n cannot be bounded above by a constant for all $n \geq n_0$. For instance, consider $n = n_0 + c$: we see that $n \geq n_0$, but $n > (c - 5)/3$. Thus, our original assumption was incorrect, which means that $3n^2 + 5n$ is not $O(n)$. ■

Big-Ω Notation

Let f(n) & g(n) be functions defined on the positive integers.

What do we mean when we say "f(n) is Ω(g(n))"?

In *Math*

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

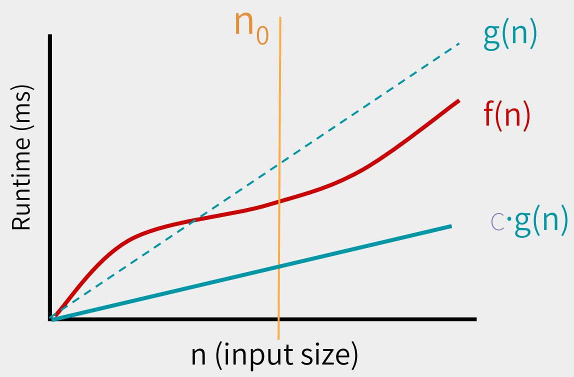
$$\Leftrightarrow$$

$$\exists\ c, n_0 > 0 \text{ s.t. } \forall\ n \geq n_0,$$

$$f(n) \geq c \cdot g(n)$$

inequality switched directions!

In Pictures



Big-Ω Notation

Let f(n) & g(n) be functions defined on the positive integers.

What do we mean when we say "f(n) is Ω(g(n))"?

In *Math*

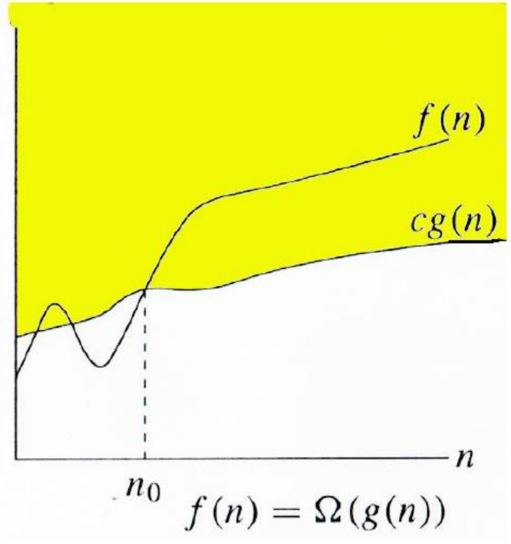
$$f(n) = \Omega(g(n))$$

$$\Leftrightarrow$$

$$\exists\ c, n_0 > 0 \text{ s.t. } \forall\ n \geq n_0,$$

$$f(n) \geq c \cdot g(n)$$

inequality switched directions!



Big-Θ Notation

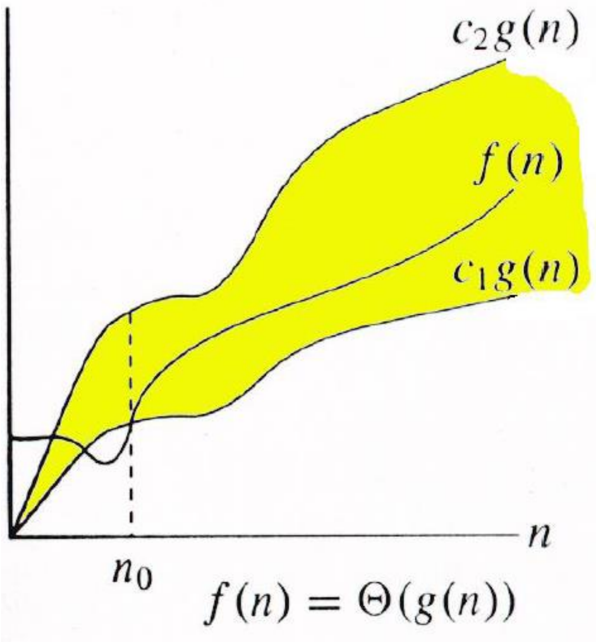
We say **“f(n) is Θ(g(n))”**
if and only if both

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

and

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

$$f(n) = \Theta(g(n))$$
$$\Leftrightarrow$$
$$\exists\ c_1, c_2, n_0 > 0 \text{ s.t. } \forall\ n \ge n_0,$$
$$c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$$



PROVING BIG-Θ NOTATION

Prove that $n^2 + 4n^2 = \Theta(n^2)$.

$$f(n) = \Theta(g(n))$$
$$\Leftrightarrow$$
$$\exists\, c_1, c_2, n_0 > 0 \text{ s.t. } \forall\, n \geq n_0,$$
$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

$$\mathbf{n^2 + 4n^2 = \Theta(n^2)} \quad c_1 = ?,\ c_2 = ?\ n_0 = ?$$

$$c_1 \times \mathbf{n^2} \leq \mathbf{n^2 + 4n^2} \leq c_2\ \mathbf{n^2}$$

$$c_1 \times \mathbf{n^2} \leq \quad \mathbf{5n^2} \quad \leq c_2\ \mathbf{n^2}$$

$$1 \times \mathbf{n^2} \leq \quad \mathbf{5n^2} \quad \leq 5\ \mathbf{n^2}$$

$$1 \times \mathbf{n^2} \leq \mathbf{n^2 + 4n^2} \leq 5\ \mathbf{n^2}$$

$$c_1 = 1,\ c_2 = 5\ n_0 = 1$$

Proving Big- ϴ Bounds: Example

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

find c_1, c_2 & n_0 such that for all $n \geq n_0$:

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

$$0 \leq c_1 . n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 . n^2$$

Divide by n^2: $0 \leq c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$

$c_1 \leq \frac{1}{2} - \frac{3}{n}$ holds for $n \geq 10$ and $c_1 = 1/5$

$\frac{1}{2} - \frac{3}{n} \leq c_2$ holds for $n \geq 10$ and $c_2 = 1$

$$0 \leq \frac{1}{5} . n^2 \leq \frac{1}{2}n^2 - 3n \leq 1 . n^2$$

Thus it is shown that $\frac{1}{2}n^2 - 3n = \theta(n^2)$ [for $c_1 = 1/5$, $c_2 = 1$, $n_0 = 10$]

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

$\Leftrightarrow$

$\exists\ c_1, c_2, n_0 > 0$ s.t. $\forall\ n \geq n_0$,

$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

Asymptotic Notations (continued)

O(1) - Constant Time

- Algorithm requires same fixed number of steps regardless of the size of the task.
- For example: Push/Pop in Stack or Insert or Remove for a Queue.
- Constant Time Algorithms are best algorithms unless that time is very long.
- 25 = O(1), i.e. [any constant] = O(1)

O(n) – Linear Time

- Algorithm requires number of steps proportional to the size of the task.
- For example: Traversal of linked-list or array, finding max./min. element in a list etc.

O (lg n)

- Algorithm having running time growing more slowly than the size of the input.
- Double the input, and the running time only gets a little longer, not doubled.
- For example: Binary Search.

Asymptotic Notations (continued)

$O(n^2)$ - Quadratic Time

- The number of operations is proportional to the size of task squared.
- Example 1: Selection sort of n elements.
- Example 2: Comparing two-dimensional array of size n by n

Big-O notation

- Big-O only gives sensible comparison of algorithms in different complexity classes when n is large.
- Big-O notation cannot compare algorithms in the same complexity class.
- For example: $O(n^2)$ is a set, or family, of fucntion with the same of smaller order of growth like $n^2 + n$, $100n + 5$, $4n^2 – n \lg n + 12$, $n^2/5 – 100n$, $n \log n$, $50n$, and so forth. Moreover, note! $n^3 \notin O(n^2)$

Arithmetic of of Big-O, Ω and ϴ Notations

Transitivity

- $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$ $\Rightarrow$ $f(n) \in O(h(n))$
- $f(n) \in \Omega(g(n))$ and $g(n) \in \Omega(h(n))$ $\Rightarrow$ $f(n) \in \Omega(h(n))$
- $f(n) \in \Theta(g(n))$ and $g(n) \in \Theta(h(n))$ $\Rightarrow$ $f(n) \in \Theta(h(n))$

Scaling

- If $f(n)) \in O(g(n))$ then for any $k > 0$, $f(n)) \in O(k.g(n))$

Reflexivity

- $f(n) \in \Theta(g(n))$ then $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$

Arithmetic of of Big-O, Ω and ϴ Notations

Sums

- If $f_1(n)) \in O\ (g_1(n))$ and $f_2(n)) \in O\ (g_2(n))$ then $(f_1 + f_2)(n) \in O\ (\max(g_1(n),g_2(n))$

Symmetry

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

Transpose Symmetry

- $f(n) \in O\ (g(n))$ if and only if $g(n) \in \Omega\ (f(n))$
- $f(n) \in o\ (g(n))$ if and only if $g(n) \in \omega\ (f(n))$

Arithmetic of of Big-O, Ω and Θ Notations

- $f_1(n) * f_2(n) = O(g_1(n) * g_2(n))$

- O $(n^{c1}) \subset$ O (n^{c2}) for any c1 < c2

- For any costants a, b, c > 0

  $$O(a) \subset O(\log n) \subset O(n^b) \subset O(c^n)$$

- Multipying with n, will result in:

  $$O(an) \subset O(n.\log n) \subset O(n^{b+1}) \subset O(nc^n)$$

Little-o Notation

Let f(n) & g(n) be functions defined on the positive integers.

What do we mean when we say "f(n) is o(g(n))"?

In Math

$$f(n) = o(g(n))$$

$$\Leftrightarrow$$

$$\forall\ c > 0,\ \exists\ n_0 > 0 \text{ s.t. } \forall\ n \geq n_0,$$

$$f(n) < c \cdot g(n)$$

f(n) becomes insignificant relative to g(n) as n approaches infinity:

$$\lim_{n \to \infty} [f(n) / g(n)] = 0$$

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

o notation

f(n) = o(g(n)) for any **constant**
$c > 0$ there is a constant **n_0** > 0 such that

$$0 \le f(n) < c \cdot g(n)$$

$3n^2 + 5n = O(n^2)$ *asymptotically tight.*
But
$3n^2 + 5n = O(n^3)$ is not *asymptotically tight.*
$3n + 5 = O(n^2)$ is not *asymptotically tight.*

$3n + 5 = o(n^2)$
$3n^2 + 5 \neq o(n^2)$

Little- ω Notation

Let f(n) & g(n) be functions defined on the positive integers.

What do we mean when we say "f(n) is ω(g(n))"?

In Math

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

$$\Leftrightarrow$$

$$\forall\ c > 0,\ \exists\ n_0 > 0 \text{ s.t. } \forall\ n \geq n_0,$$

$$f(n) > c \cdot g(n)$$

f(n) becomes very large relative to g(n) as n approaches infinity:

$$\lim_{n \to \infty} [f(n) / g(n)] = \infty$$

g(n) is an **lower bound** *for f(n) that is* ***not asymptotically tight.***

ω notation

$f(n) = \omega(g(n))$ for any **constant** $c > 0$ there is a constant $n_0 > 0$ such that

$$0 \leq f(n) > c \cdot g(n)$$

$3n^2 + 5 = \omega(n)$

$3n + 5 \neq \omega(n)$

g(n) is **lower bound** *for f(n) that is not asymptotically tight.*

Asymptotic Notation Summary

Bound	Definition (How To Prove)	It Represents
$f(n) = O(g(n))$	$\exists\ c > 0,\ \exists\ n_0 > 0$ s.t. $\forall\ n \geq n_0,\ f(n) \leq c \cdot g(n)$	upper bound
$f(n) = o(g(n))$	$\forall\ c > 0,\ \exists\ n0 > 0$ s.t. $\forall\ n \geq n0,\ f(n) < c \cdot g(n)$	upper bound Not asymptotically tight
$f(n) = \Omega(g(n))$	$\exists\ c > 0,\ \exists\ n_0 > 0$ s.t. $\forall\ n \geq n_0,\ f(n) \geq c \cdot g(n)$	lower bound
$f(n) = \omega(g(n))$	$\forall\ c > 0,\ \exists\ n0 > 0$ s.t. $\forall\ n \geq n0,\ f(n) > c \cdot g(n)$	lower bound Not asymptotically tight
$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$	tight bound

Comparison Of functions

$f(n) = O(g(n)) \approx a \leq b$

$f(n) = \Omega(g(n)) \approx a \geq b$

$f(n) = \Theta(g(n)) \approx a = b$

$f(n) = o(g(n)) \approx a < b$

$f(n) = \omega(g(n)) \approx a > b$

Proving Big-O Bounds: Example # 3(ii) (Method # 2)

Show that $f(n) = 5n \log_2 n + 8n - 200 = O(n \log_2 n)$

$$f(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists\ c, n_0 > 0 \text{ s.t. } \forall\ n \geq n_0,$$
$$f(n) \leq c \cdot g(n)$$

find a c & n_0 such that for all $n \geq n_0$:

$$5n \log_2 n + 8n - 200 \leq c \bullet n \log_2 n$$

While finding c and n0, in

$$5n \log_2 n + 8n - 200 \leq 5n \log_2 n + 8n \log_2 n$$

$$5n \log_2 n + 8n - 200 \leq 13n \log_2 n \qquad \text{for} \quad n \geq 2$$

Thus

$$f(n) = 5n \log_2 n + 8n - 200 = O(n \log_2 n) \text{ [for } c = 13,\ n_0 = 2]$$

Proving Big- ϴ Bounds: Example

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

find c_1, c_2 & n_0 such that for all $n \geq n_0$:

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

$$0 \leq c_1 . n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 . n^2$$

Divide by n^2: $0 \leq c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$

$c_1 \leq \frac{1}{2} - \frac{3}{n}$ holds for $n \geq 7$ and $c_1 \leq \frac{1}{14}$

$\frac{1}{2} - \frac{3}{n} \leq c_2$ holds for $n \geq 7$ and $c_2 \geq \frac{1}{14}$

$$0 \leq \frac{1}{14}n^2 \leq \frac{1}{2}n^2 - 3n \leq \frac{1}{14} . n^2$$

Thus it is shown that $\frac{1}{2}n^2 - 3n = \theta(n^2)$ [for $c_1 = \frac{1}{14}$, $c_2 = \frac{1}{14}$, $n_0 = 7$]

Proving Big- ϴ Bounds: Example (Method # 1)

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

find c_1, c_2 & n_0 such that for all $n \geq n_0$:

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

$$0\ \leq c_1 . n^2 \leq \frac{1}{2}\,n^2\ - 3n\ \leq c_2\ .\ n^2$$

Divide by n^2: $0\ \leq c_1\ \leq \frac{1}{2}\ -\ \frac{3}{n}\ \leq c_2$

$c_1\ \leq \frac{1}{2}\ -\ \frac{3}{n}$ holds for $n \geq 7$ and $c1 \leq \frac{1}{14}$, $n \geq 7$

$\frac{1}{2}\ \leq c_2$ holds for $n \geq 1$ and $c_2 \geq \frac{1}{2}$

$$0\ \leq \frac{1}{14} . n^2 \leq \frac{1}{2}\,n^2\ - 3n\ \leq \frac{1}{2} .\ n^2$$

Thus it is shown that $\frac{1}{2}\,n^2\ - 3n =\ \theta(n^2)$ [for $c_1 \leq \frac{1}{14}$, $c_2 \geq \frac{1}{2}$, $n_0 = 7$]

f(n) = ϴ(g(n))

$\Leftrightarrow$

$\exists\ c_1, c_2, n_0 > 0$ s.t. $\forall\ n \geq n_0$,

$c_1 \cdot g(n)\ \leq\ f(n)\ \leq\ c_2 \cdot g(n)$

Proving Big-O Bounds: Example # 3 (Method # 2)

Prove that $f(n) = 3n^2 + 5n - 7 = O(n^2)$.

$f(n) = O(g(n))$
$\Leftrightarrow$
$\exists\, c, n_0 > 0$ s.t. $\forall\, n \geq n_0$,
$f(n) \leq c \cdot g(n)$

find a c & n_0 such that for all $n \geq n_0$:

$$3n^2 + 5n - 7 \leq c \bullet n^2$$

$3n^2 \leq 3n^2$ for $n \geq 0$

$5n \leq 5n^2$ for $n \geq 0$

$\cancel{-7 \leq -7n^2 \quad \text{for } n \geq 1}$

$3n^2 + 5n - 7 \leq 3n^2 + 5n^2$ for $n \geq 0$

$3n^2 + 5n - 7 \leq 8n^2$ for $n \geq 1$

Proved that $f(n) = 3n^2 + 5n - 7 = O(n^2)$ [for $c = 8$, $n_0 = 1$]