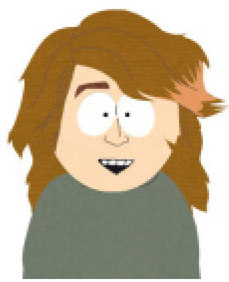
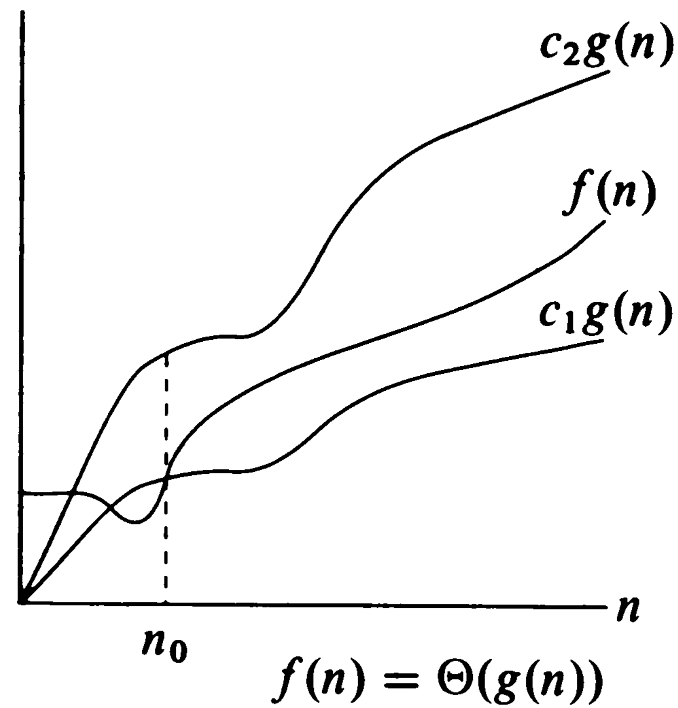
Asymptotic Analysis





Carlos Moreno
cmoreno@uwaterloo.ca
EIT-4103

https://ece.uwaterloo.ca/~cmoreno/ece250

These slides, the course material, and course web site are based on work by Douglas W. Harder

Asymptotic Analysis

- Today's class:
  - Introduce and justify the notion of Asymptotic Analysis.
  - Introduce Asymptotic notation (Landau symbols).
  - Look at some of the common functions in the analysis of algorithms.
  - Next class, we'll look at an alternative criterion; namely, using limits as $n \longrightarrow \infty$

Asymptotic Analysis

- Main goal:
  - Mathematically describe the behaviour of algorithms.
  - By “behaviour” we often refer to:
    - How long does it take to execute (the “run time”)
    - How much memory does it require

Asymptotic Analysis

- Main goal:
  - Mathematically describe the behaviour of algorithms.
  - By "behaviour" we often refer to:
    - How long does it take to execute (the "run time")
    - How much memory does it require
  - Typically, this description is given as a function of the algorithm's input *size*.

Asymptotic Analysis

- An important detail to consider:
  - We need to specify what we mean by the size of the input.
  - Typically, this will be related (directly or indirectly) to the number of elements that the algorithm operates on.
    - The number of elements in an array or linked list (e.g., for an algorithm that finds the highest value in a list)
    - The size (dimensions) of a square ($n \times n$) matrix.
    - The length of a given fragment of text.
    - etc.

Asymptotic Analysis

- An important detail to consider:
  - We need to specify what we mean by the size of the input.
  - In some cases (probably not too often in this course), it might be the size in bits of the input value to the algorithm (in this case, if the input value is x, the size of the input is $\lg x$ bits)

Asymptotic Analysis

- Problem:
  - How can we hope to determine the time it takes for an algorithm to run, if the algorithm (or rather, an implementation of it) can be run on different computers, with different clock speeds, different instruction sets, different memory access speeds, etc.? (not to mention *different implementations*)

Asymptotic Analysis

- Problem:
  - How can we hope to determine the time it takes for an algorithm to run, if the algorithm (or rather, an implementation of it) can be run on different computers, with different clock speeds, different instruction sets, different memory access speeds, etc.? (not to mention *different implementations*)
  - This seems to suggest that instead of measuring actual time, we should measure something like *number of operations* that it takes to execute.

Asymptotic Analysis

- Problem:
  - But then, if we measure number of operations, we still have a problem — different implementations of an algorithm (perhaps implementations in different programming languages?) still translate into different actual number of CPU instructions.

Asymptotic Analysis

- Problem:
  - But then, if we measure number of operations, we still have a problem — different implementations of an algorithm (perhaps implementations in different programming languages?) still translate into different actual number of CPU instructions.
  - Maybe not too bad — we're talking differences given by a scaling constant (C++ may be on average 2.5 times faster than Java, and maybe twice as slow as assembler code)

Asymptotic Analysis

- Bottom line:
  - This suggests that maybe we shouldn't care about proportionality constants when analyzing algorithms.

Asymptotic Analysis

- Bottom line:
  - This suggests that maybe we shouldn't care about proportionality constants when analyzing algorithms.
  - In fact, a difference by a factor of 2 may easily be compensated for by purchasing a machine that is twice as fast!

Asymptotic Analysis

- Bottom line:
  - This suggests that maybe we shouldn't care about proportionality constants when analyzing algorithms.
  - In fact, a difference by a factor of 2 may easily be compensated for by purchasing a machine that is twice as fast!
  - This may sound absurd, since, well, if we have the machine twice as fast, we run the faster algorithm on that machine anyway! But consider the following:

Asymptotic Analysis

- Bottom line:
  - We have an algorithm that has been already implemented and tested, and it's ready to be used.
  - We have an alternative — an algorithm that runs twice as fast. However, we have to develop it, incurring costs for:
    - Implementation / Integration
    - Testing
    - Documentation

Asymptotic Analysis

- Bottom line:
  - We have an algorithm that has been already implemented and tested, and it's ready to be used.
  - We have an alternative — an algorithm that runs twice as fast. However, we have to develop it, incurring costs for:
    - Implementation / Integration
    - Testing
    - Documentation
  - We may be talking $10k or $20k in salaries!!

Asymptotic Analysis

- Bottom line:
  - With that amount of money, we can definitely purchase a machine that is twice as fast!!

Asymptotic Analysis

- BTW … Is this “we buy a machine twice as fast” argument valid if we're comparing “linear” search vs. binary search?

Asymptotic Analysis

- BTW … Is this "we buy a machine twice as fast" argument valid if we're comparing "linear" search vs. binary search?

  Linear search takes, in the worst-case, $c_1 \cdot n$ (for some constant $c_1 > 0$) operations to find a value in a list of n elements. Binary search takes $c_2 \log n$ (for some constant $c_2 > 0$).

Asymptotic Analysis

- BTW … Is this "we buy a machine twice as fast" argument valid if we're comparing "linear" search vs. binary search?

  Linear search takes, in the worst-case, $c_1 \cdot n$ (for some constant $c_1 > 0$) operations to find a value in a list of n elements. Binary search takes $c_2 \log n$ (for some constant $c_2 > 0$).

  Is it enough to purchase a faster computer?

Asymptotic Analysis

- Ok, so we established that we should not care for proportionality constants ("scale factors")
- But what about in functions such as polynomials where we have several terms, some more "important" than others? (e.g., $f(n) = n^3 + 2n^2 + 10n + 7$)
- In the above example, should we care about the low-order terms, $2n^2$, $10n$, and 7?

Asymptotic Analysis

- Notice that those terms make a noticeable difference (relative difference, that is) only for low(-ish) values of n. For large values of n, the polynomial's behaviour is essentially defined by the leading (dominating) term — in this example, n^3.

Asymptotic Analysis

- Notice that those terms make a noticeable difference (relative difference, that is) only for low(-ish) values of *n*. For large values of *n*, the polynomial's behaviour is essentially defined by the leading (dominating) term — in this example, n^3.
- Ok, but if *n* is low, then the execution time will be fast anyway (in any case, the "faster computer" argument would apply), so why would we care about the low-order terms?

Asymptotic Analysis

- Bottom line:
  - We try to define behaviour of algorithms focusing on the important part — pattern or rate of growth of the function; disregarding (1) low-order terms and (2) scale factors / proportionality constants.

Asymptotic Analysis

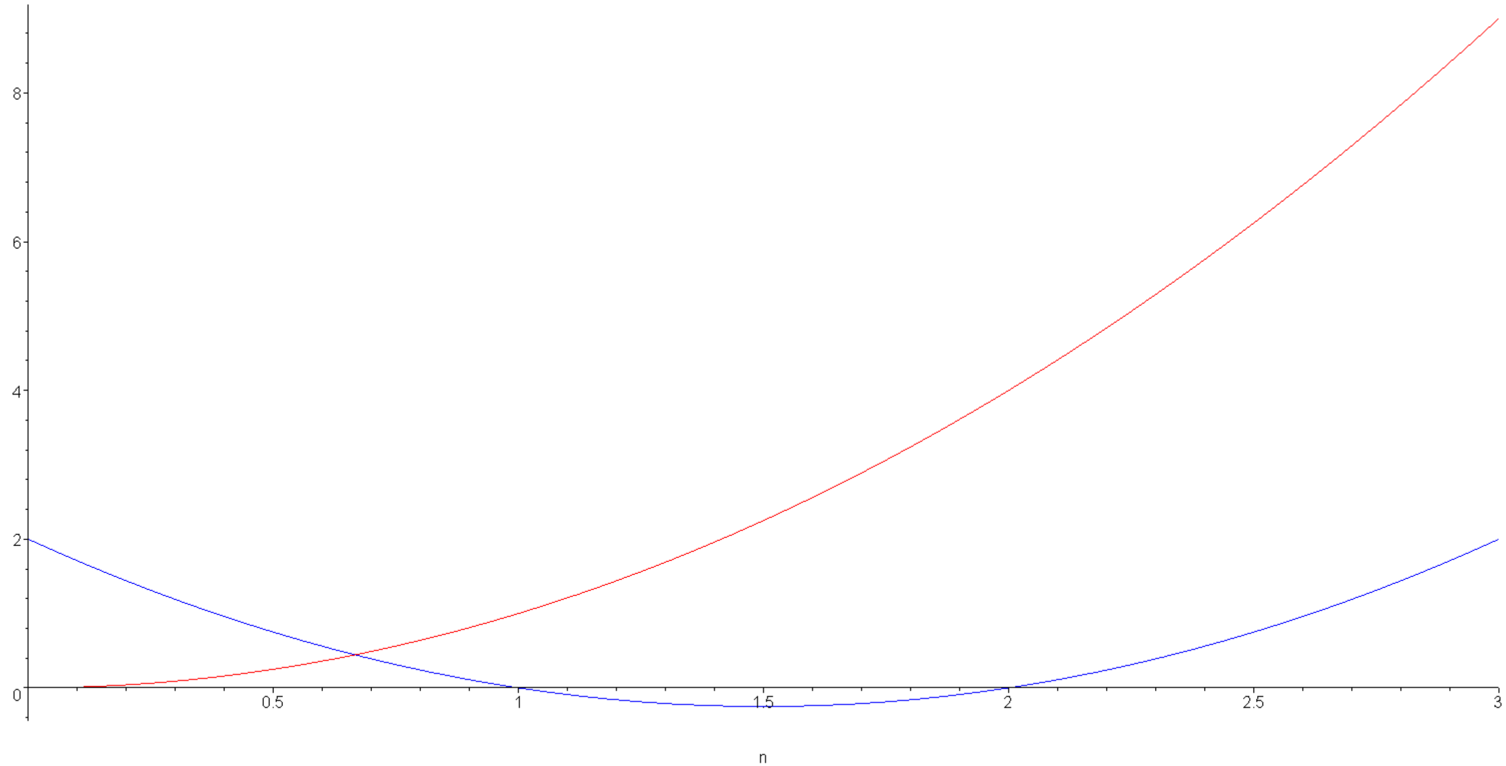
- Bottom line:
  - We try to define behaviour of algorithms focusing on the important part — pattern or rate of growth of the function; disregarding (1) low-order terms and (2) scale factors / proportionality constants.
  - Thus, we define the asymptotic behaviour of the function, which is directly relevant to the issue of *scalability* of the algorithm? How does execution time grow as the size of the input grows — say, when it doubles?

Asymptotic Analysis

- Couple of examples:

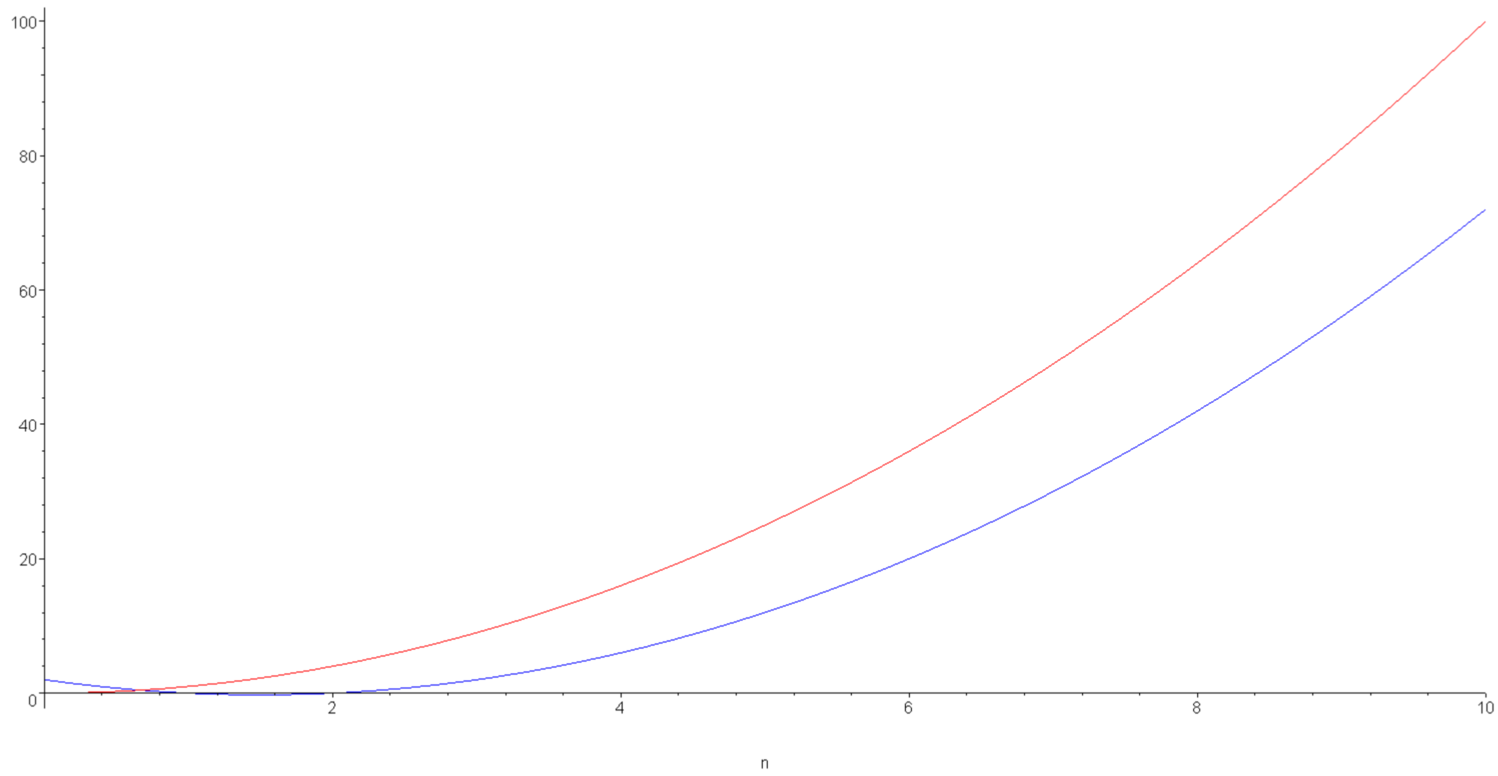
Asymptotic Analysis

- Quadratic growth:
  - Consider the two functions $f(n) = n^2$ and $g(n) = n^2 - 3n + 2$ around $n = 0$.



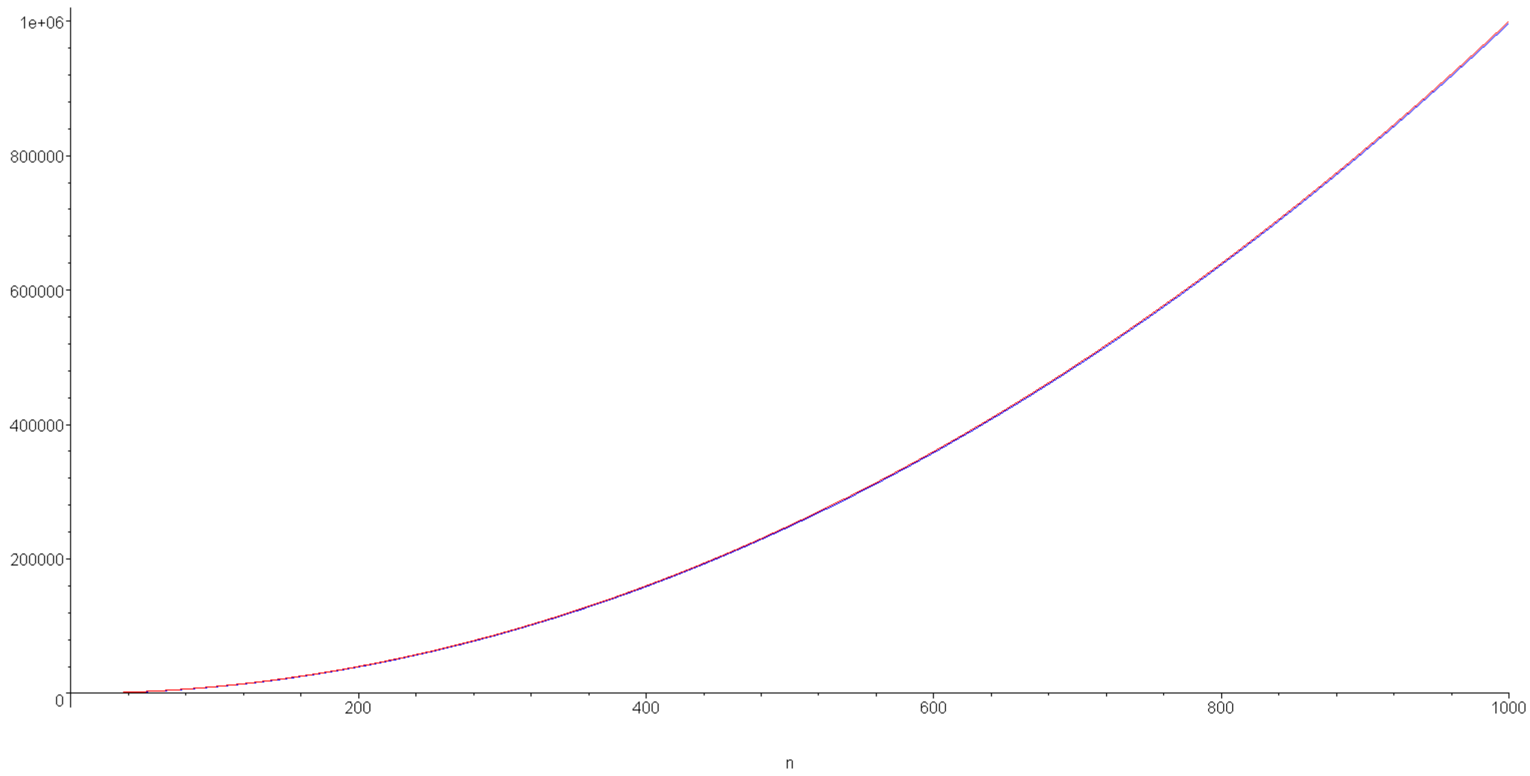
Asymptotic Analysis

- Quadratic growth:
  - If we look ahead, say, to n = 10, they start to look more similar:



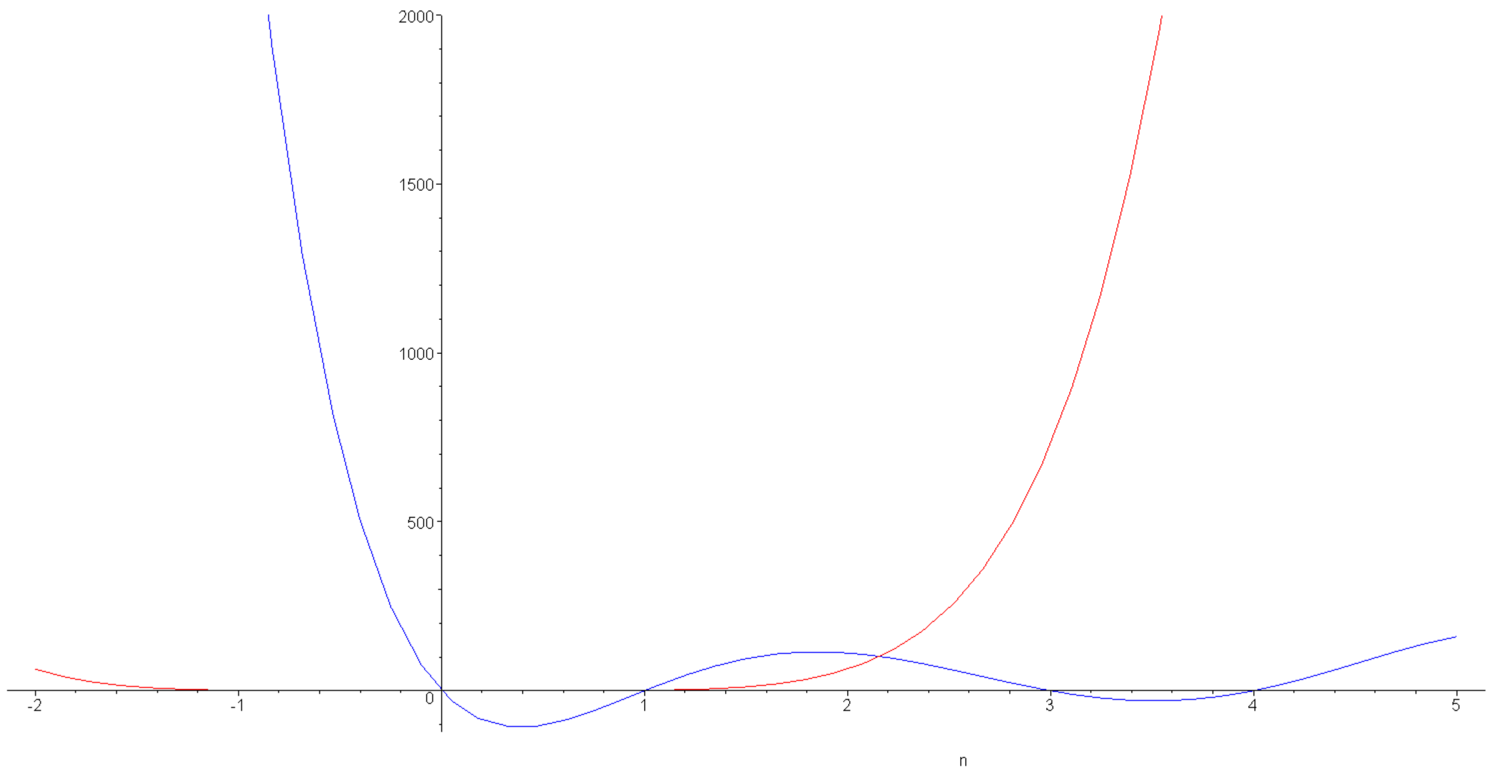
Asymptotic Analysis

- Quadratic growth:
  - By $n = 1000$, they are almost indistinguishable:



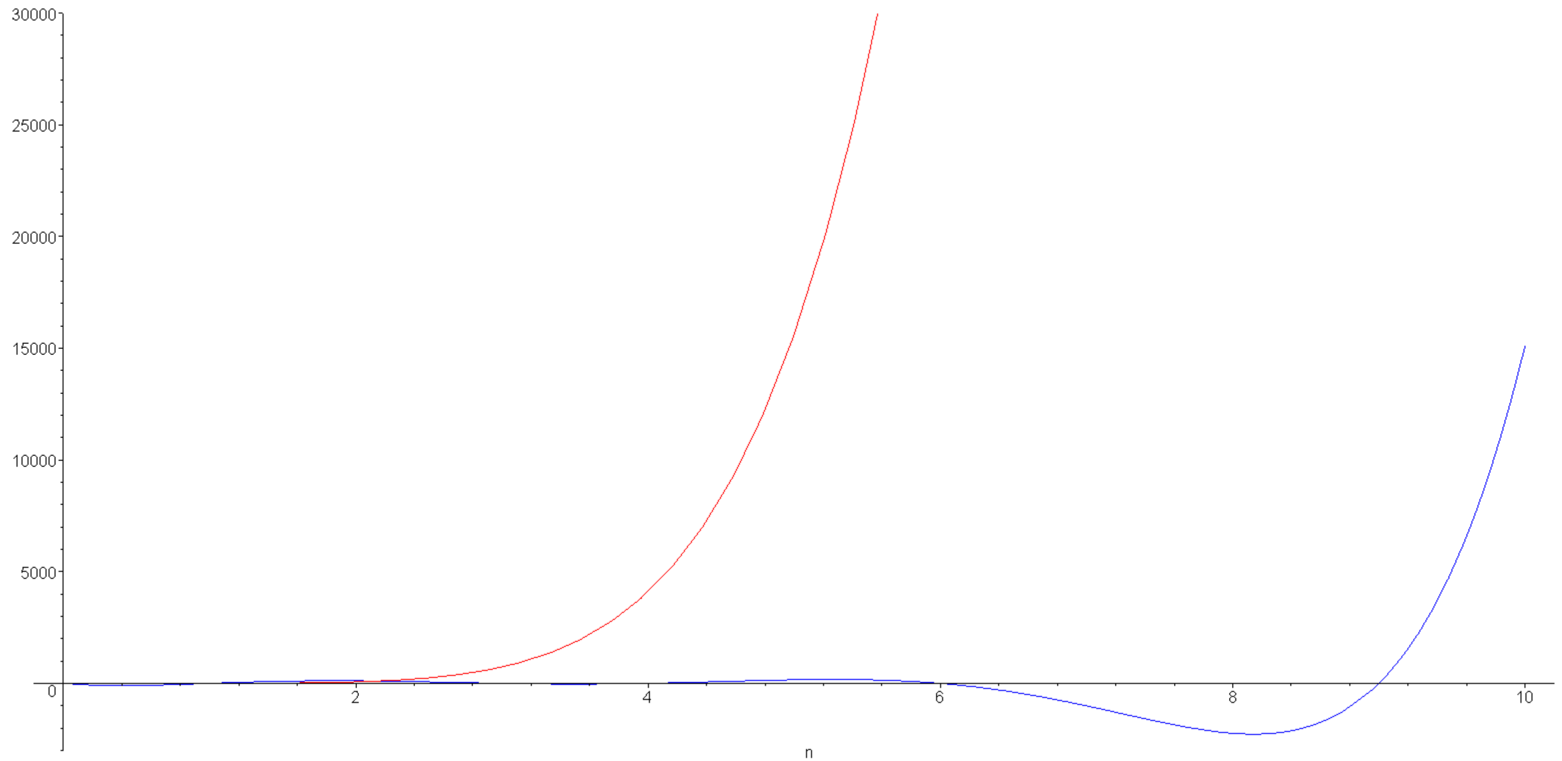
Asymptotic Analysis

- Polynomial growth:
  - Consider, first around $n = 0$, $f(n) = n^6$ and $g(n) = n^6 - 23n^5 + 193n^4 - 729n^3 + 1206n^2 - 648n$:



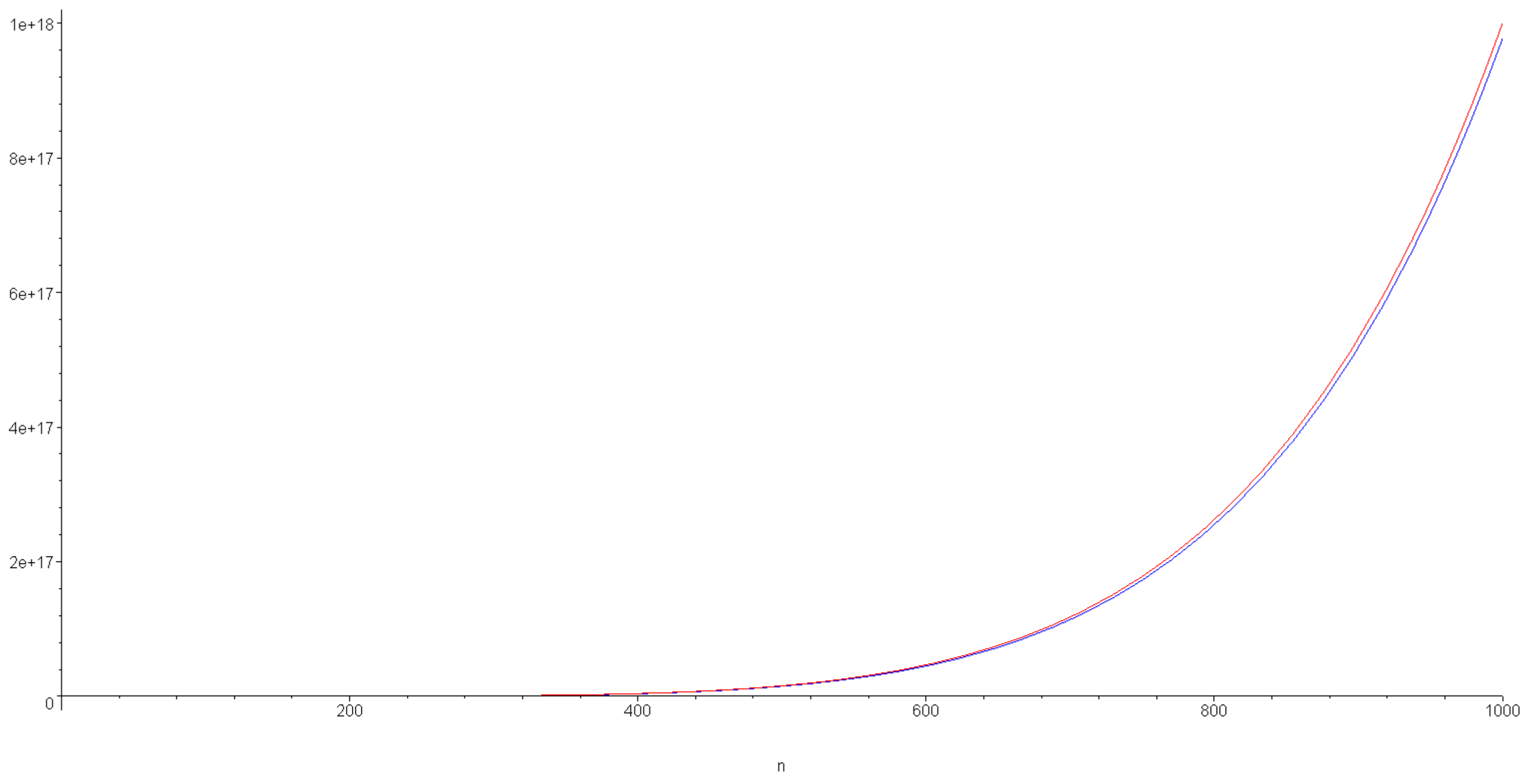
Asymptotic Analysis

- Polynomial growth:
  - Even around *n* = 10 they don't look too similar:



Asymptotic Analysis

- Polynomial growth:
  - But sooner or later (showing to n = 1000), the term n^6 definitely wins/dominates:



Asymptotic Analysis

- Asymptotic notation is based on Landau symbols.
- We'll start with the three arguably most important: O (big-Oh), Θ (big-Theta), and Ω (big-Omega)
- Formally, each of these symbols, applied to a given function f(n), defines a set that includes all the functions related to f(n) in a particular way (each symbol defines such particular way):

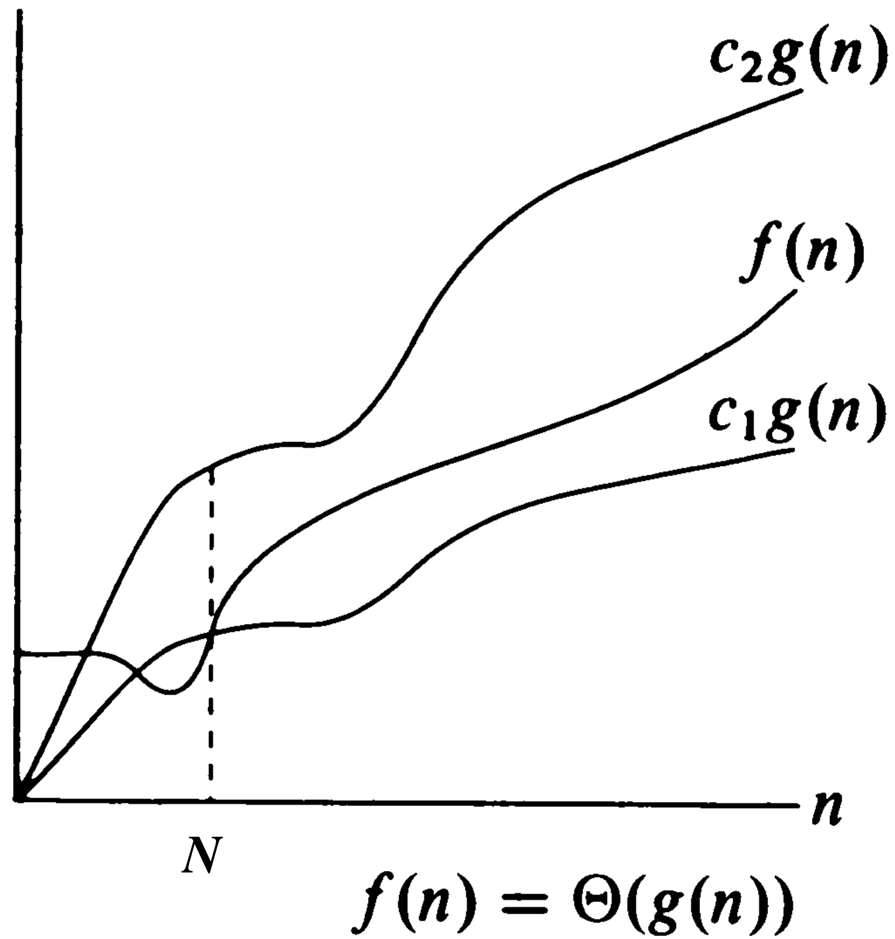
Asymptotic Analysis

- Θ (big-Theta) is defined as follows:

$$\Theta\left(g\left(n\right)\right) = \left\{ f\left(n\right) \;\middle|\; \begin{array}{l} \exists \;\; c_1 > 0,\;\; c_2 > 0,\;\; N > 0 \;\; \textit{such that} \\ 0 \leqslant c_1 g\left(n\right) \leqslant f\left(n\right) \leqslant c_2 g\left(n\right) \;\; \forall \; n \geqslant N \end{array} \right\}$$

Asymptotic Analysis

- It may be visualized as follows:



Asymptotic Analysis

- This corresponds with the notion of a function being "asymptotically proportional" to another, in that two scaled versions of g(*n*) provide a tight bound (upper- and lower-bound) for f(*n*) for sufficiently large values of *n*
- That is, it goes with our notion of focusing on what's important, disregarding proportionality constants and lower-order terms (why? One answer for each of the two aspects being disregarded)

Asymptotic Analysis

- An example:
  - Let's show that $\mathrm{f}(n) = 5\mathrm{n}^2 + 2\mathrm{n} \in \Theta(n^2)$

Asymptotic Analysis

- An example:
  - Let's show that $f(n) = 5n^2 + 2n \in \Theta(n^2)$
  - In this case, g(n) is n^2, so we have to show that there exist (i.e. by finding) constants $c_1 > 0$, $c_2 > 0$, and $N > 0$ such that:

$$0 \leqslant c_1 n^2 \leqslant 5n^2 + 2n \leqslant c_2 n^2$$

Asymptotic Analysis

- An example:
  - The first (left-most) inequality is trivial (it usually is, since we'll be mostly interested in functions that represent running-time of algorithms — so they have to be strictly positive functions)
  - The challenge is, then, finding $c_1 > 0$, $c_2 > 0$, and $N > 0$ such that the other two inequalities are met (one at a time; then, pick the higher N, so that both are met):

Asymptotic Analysis

- An example:
  - First inequality challenges us to find $c_1 > 0$ and $N > 0$ such that:

$$c_1 n^2 \leqslant 5\text{n}^2 + 2\text{n}$$

  - And this one is rather trivial — any value of c_1 with $0 < c_1 < 5$ satisfies the above for all $n > 0$ (i.e., $c_1 = 5$ and $N = 0$ satisfy the condition)

Asymptotic Analysis

- An example:
  - Second inequality challenges us to find $c_2 > 0$ and $N > 0$ such that:

$$5n^2 + 2n \leqslant c_2 n^2$$

  - This one is a little less obvious (still easy — it just requires a bit of manipulation/algebra). Maybe start by eliminating the common (positive!) factor n:

$$5n + 2 \leqslant c_2 n \;\Rightarrow\; (c_2 - 5)n \geqslant 2$$

Asymptotic Analysis

- An example:
  - The smallest value of n for which that inequality can be satisfied is clearly 1 (for $n = 0$, no matter how large we choose c_2, the expression on the left is 0, so the inequality can not be satisfied).
  - And for $n \geq 1$, we have:

$$(c_2 - 5)n \geqslant 2 \;\Rightarrow\; c_2 \geqslant 7$$

  - So, $c_1 = 5$, $c_2 = 7$, and $N = 1$ satisfy the definition, showing that $5n^2 + 2n \in \Theta(n^2)$

Asymptotic Analysis

- A comment on notation:
  - Though $\Theta(g(n))$ is defined as a set and thus we should say that a function $f(n)$ is or is not in that set (like the example $5n^2 + 2n \in \Theta(n^2)$, we normally use a different notation.

Asymptotic Analysis

- A comment on notation:
  - Though $\Theta(g(n))$ is defined as a set and thus we should say that a function $f(n)$ is or is not in that set (like the example $5n^2 + 2n \in \Theta(n^2)$, we normally use a different notation.
  - Since what we're doing is really *describing* the function $f(n)$, we want to read that as $f(n)$ *is* $\Theta(n^2)$, and thus, the standard notation uses the equal sign, instead of the set inclusion sign. In the example above, we would say that $5n^2 + 2n = \Theta(n^2)$

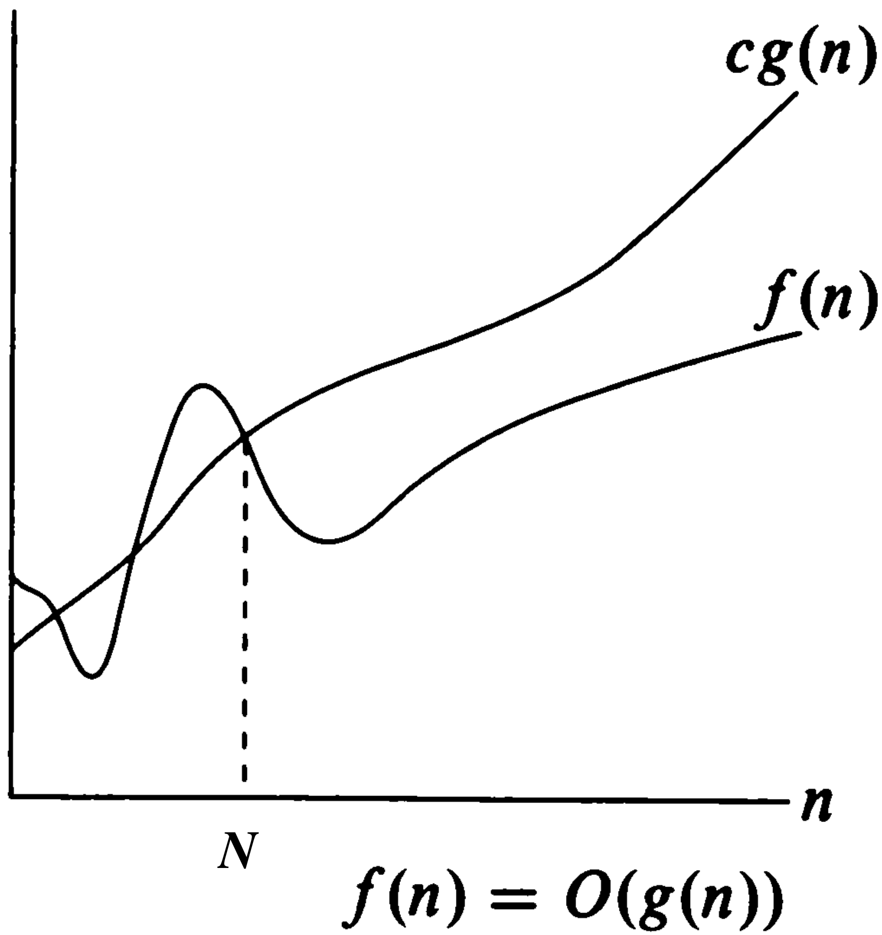
Asymptotic Analysis

- O (big-Oh) is defined as follows:

$$O(g(n)) = \left\{ f(n) \;\middle|\; \begin{array}{l} \exists \;\; c > 0, \;\; N > 0 \;\; \textit{such that} \\ 0 \leqslant f(n) \leqslant c\, g(n) \;\; \forall \; n \geqslant N \end{array} \right\}$$

Asymptotic Analysis

- It may be visualized as follows:



Asymptotic Analysis

- This one corresponds with the notion of a function being “asymptotically bounded” (upper-bounded) by another.

Asymptotic Analysis

- This one corresponds with the notion of a function being “asymptotically bounded” (upper-bounded) by another.
- An important detail is that it covers cases of functions where f(n) and g(n) are asymptotically proportional as well as cases where f(n) is asymptotically negligible with respect to g(n).

Asymptotic Analysis

- This one corresponds with the notion of a function being “asymptotically bounded” (upper-bounded) by another.
- An important detail is that it covers cases of functions where f(n) and g(n) are asymptotically proportional as well as cases where f(n) is asymptotically negligible with respect to g(n).
- So, in particular, this tells us that:

$$\mathrm{f}(n) = \Theta(\mathrm{g}(n)) \;\Rightarrow\; \mathrm{f}(n) = O(\mathrm{g}(n))$$

Asymptotic Analysis

- Right? Can we see the following from the definitions?

$$\mathrm{f}(n) = \Theta(\mathrm{g}(n)) \;\Rightarrow\; \mathrm{f}(n) = O(\mathrm{g}(n))$$

Asymptotic Analysis

- Right? Can we see the following from the definitions?

$$\mathrm{f}(n) = \Theta(\mathrm{g}(n)) \;\Rightarrow\; \mathrm{f}(n) = O(\mathrm{g}(n))$$

- Sure !! The inequality in the big-Oh definition is one of the inequalities in the big-Theta definition — so, we choose $c = c_2$ from the big-Theta definition and with that we satisfy the condition for big-Oh.

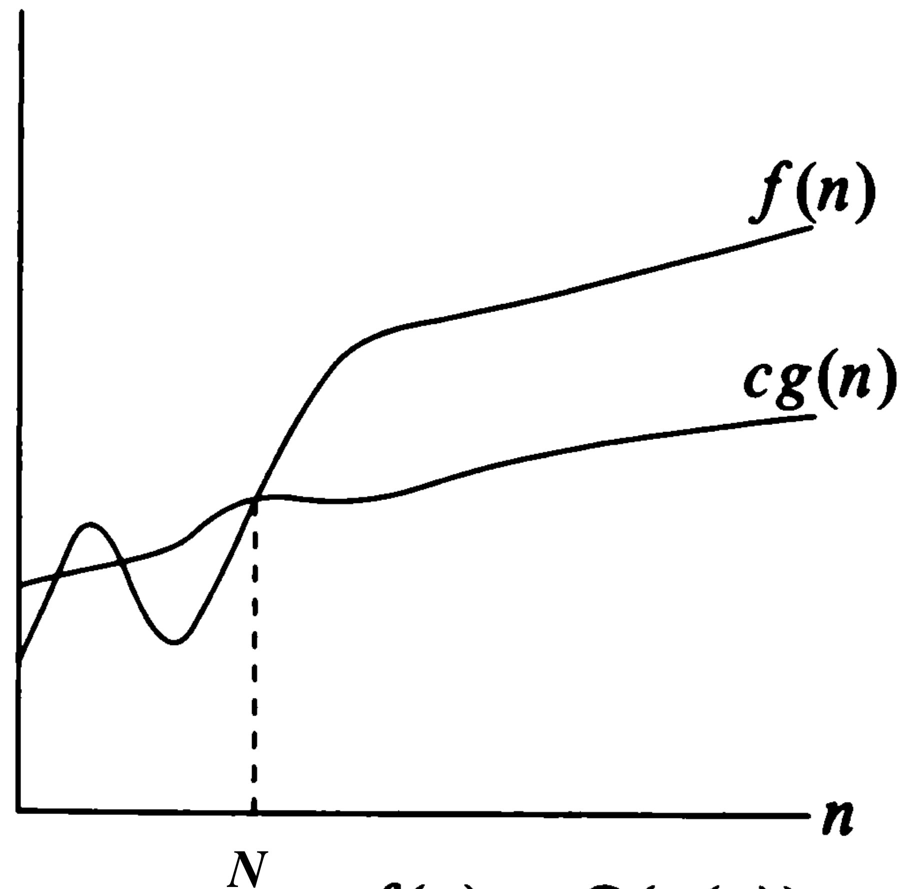
Asymptotic Analysis

- Ω (big-Omega) is defined as follows:

$$\Omega(g(n)) = \left\{ f(n) \;\middle|\; \begin{array}{l} \exists \;\; c > 0,\;\; N > 0 \;\; \textit{such that} \\ 0 \leqslant c\,g(n) \leqslant f(n) \;\; \forall \; n \geqslant N \end{array} \right\}$$

Asymptotic Analysis

- It may be visualized as follows:



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

Asymptotic Analysis

- And this one has the interpretation of an asymptotic lower-bound.
- Again, this includes cases of asymptotically proportional functions as well as cases where any scaled version of g(*n*) is asymptotically negligible with respect to f(*n*).
- So, like in the big-Oh case, we have that:

$$\mathrm{f}(n) = \Theta(\mathrm{g}(n)) \Rightarrow \mathrm{f}(n) = \Omega(\mathrm{g}(n))$$

Asymptotic Analysis

- And speaking of big-Omega as an asymptotic lower-bound — someone remembers when did we encounter this notion during the previous lectures?

Asymptotic Analysis

- And speaking of big-Omega as an asymptotic lower-bound — someone remembers when did we encounter this notion during the previous lectures?

- Recall that we mentioned the known fact that no sort algorithm can sort a group of n values in less than an amount proportional to $n \log n$ (in the worst-case)

Asymptotic Analysis

- Some common functions encountered in the analysis of algorithms — we give them *names*:

Asymptotic Analysis

- Some common functions encountered in the analysis of algorithms — we give them *names*:

$\Theta(1)$	Constant (e.g., constant time)
$\Theta(\log n)$	Logarithmic
$\Theta(n)$	Linear
$\Theta(n \log n)$	«en log en»
$\Theta(n^a)$, $1<a<2$	Sub-quadratic
$\Theta(n^2)$	Quadratic
$\Theta(n^a)$ in general	Polynomial
$\Theta(a^n)$ $(a > 1)$	Exponential

Asymptotic Analysis

- Perhaps a subtle/tricky detail — notice that there is no single class for exponentials:

$$a^n \neq \Theta\left(b^n\right) \quad \text{if} \;\; a \neq b$$

(why? why is it not the same for logarithms?)

Summary

- During today's class, we discussed:
  - Rationale and justification for asymptotic analysis
  - Landau symbols, in particular, big-Theta, big-Oh, and big-Omega
  - Some of the common functions we encounter as part of the analysis of actual algorithms.