CSE 326: Data Structures

Asymptotic Analysis

James Fogarty

Autumn 2007

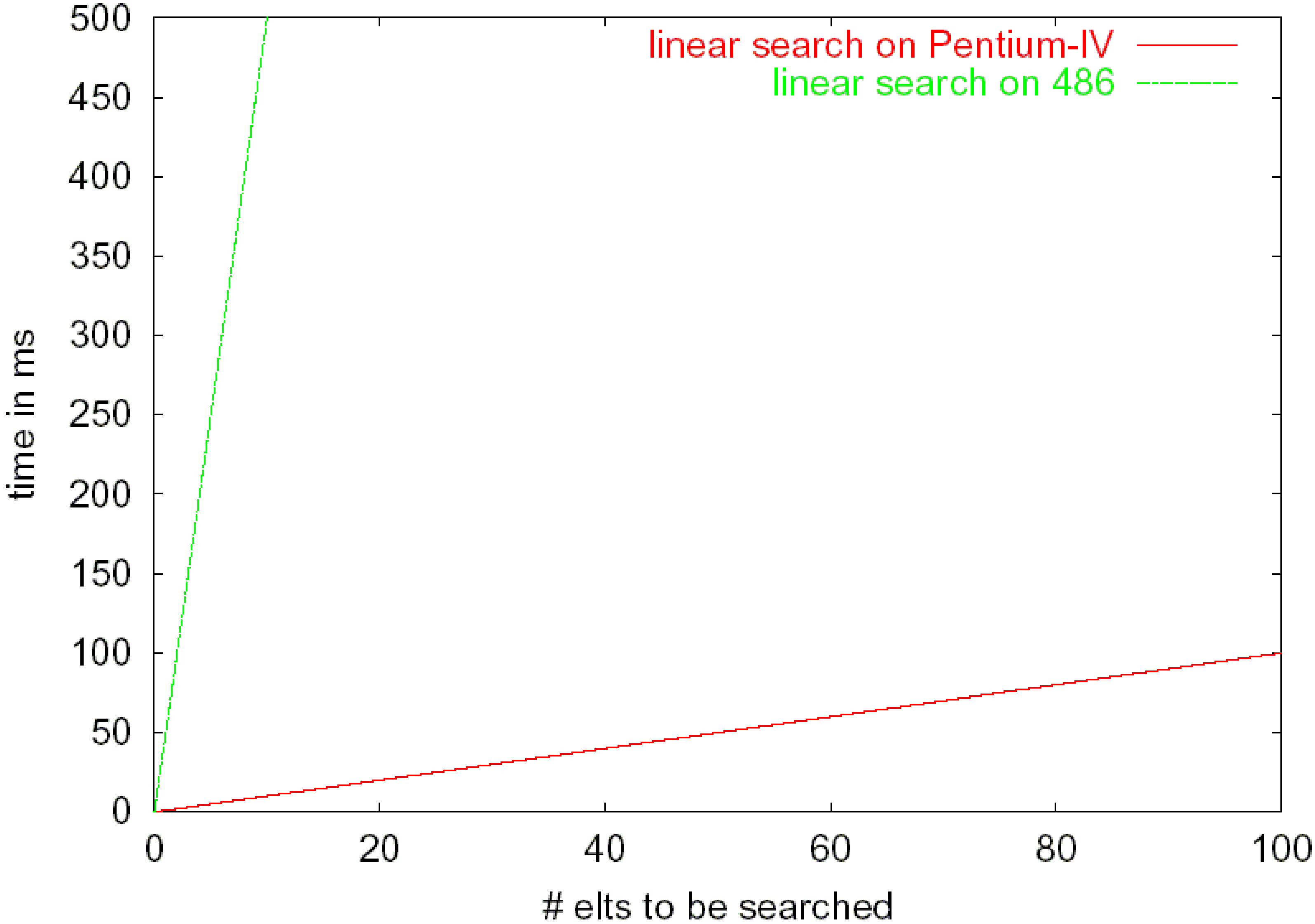
Lecture 3

Linear Search vs Binary Search

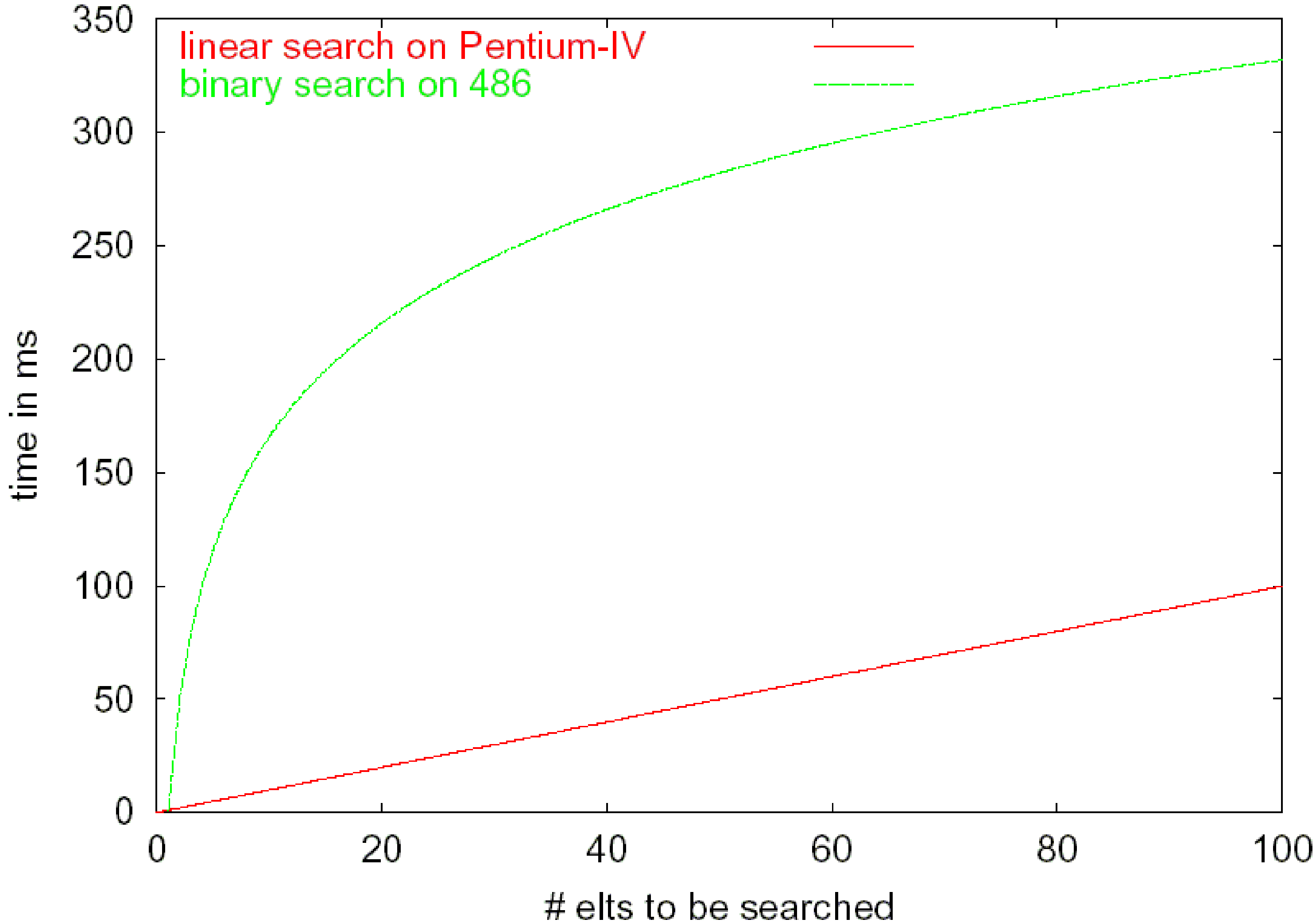
	Linear Search	Binary Search
Best Case	4 at [0]	4 at [middle]
Worst Case	3n+2	4 log n + 4

So … which algorithm is better?
What tradeoffs can you make?

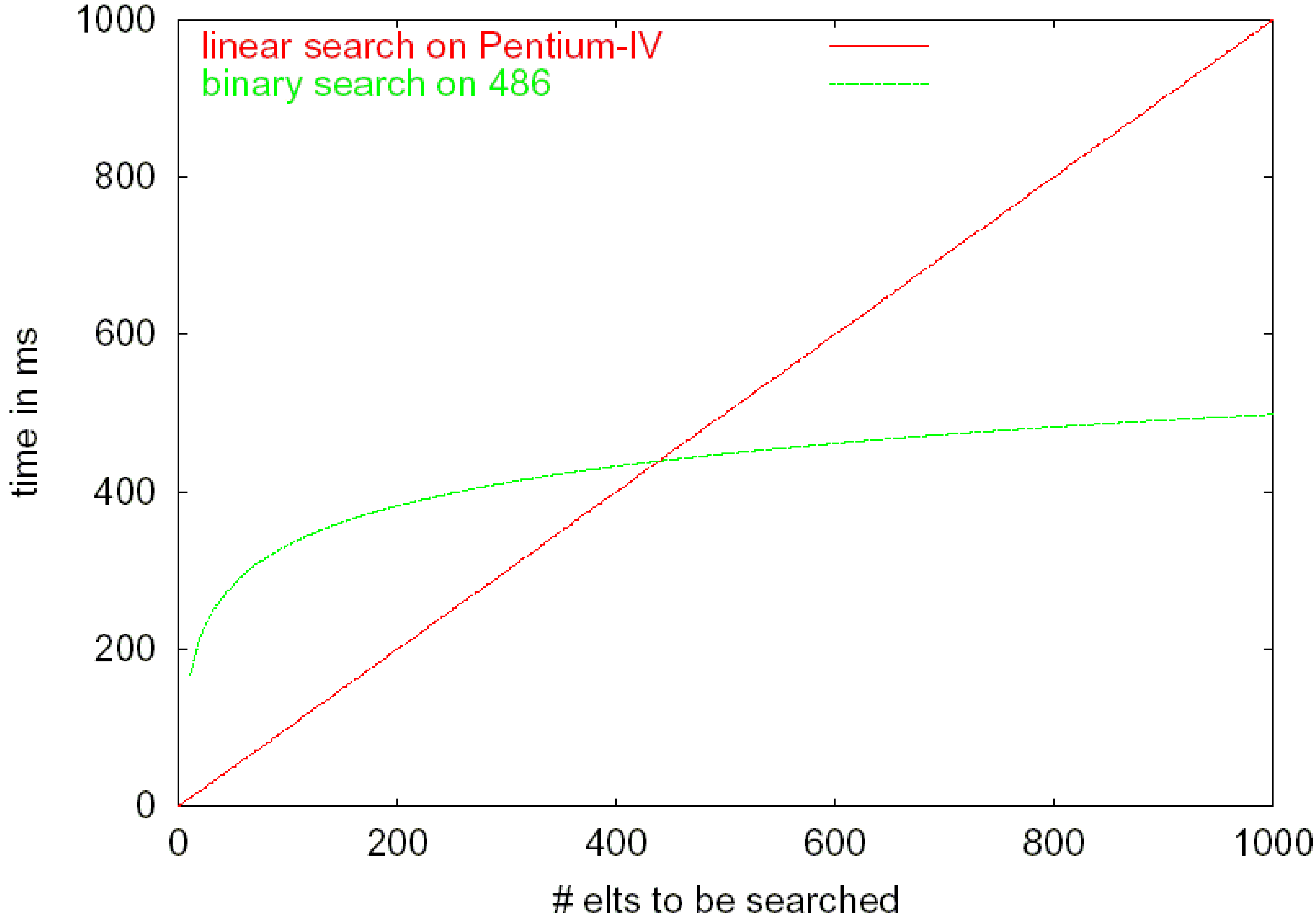
Fast Computer vs. Slow Computer



Fast Computer vs. Smart Programmer (round 1)



Fast Computer vs. Smart Programmer (round 2)



Asymptotic Analysis

- Asymptotic analysis looks at the *order* of the running time of the algorithm
  - A valuable tool when the input gets “large”
  - Ignores the *effects of different machines* or *different implementations* of an algorithm

- Intuitively, to find the asymptotic runtime, throw away the constants and low-order terms
  - Linear search is $T(n) = 3n + 2 \in \mathbf{O}(n)$
  - Binary search is $T(n) = 4 \log_2 n + 4 \in \mathbf{O}(\log n)$

Remember: the fastest algorithm has the slowest growing function for its runtime

Asymptotic Analysis

- Eliminate low order terms
  - $4n + 5 \Rightarrow$
  - $0.5\ n \log n + 2n + 7 \Rightarrow$
  - $n^3 + 2^n + 3n \Rightarrow$
- Eliminate coefficients
  - $4n \Rightarrow$
  - $0.5\ n \log n \Rightarrow$
  - $n \log n^2$ =>

Properties of Logs

- log AB = log A + log B
- Proof: $A = 2^{\log_2 A}, B = 2^{\log_2 B}$

$$AB = 2^{\log_2 A} \cdot 2^{\log_2 B} = 2^{(\log_2 A + \log_2 B)}$$

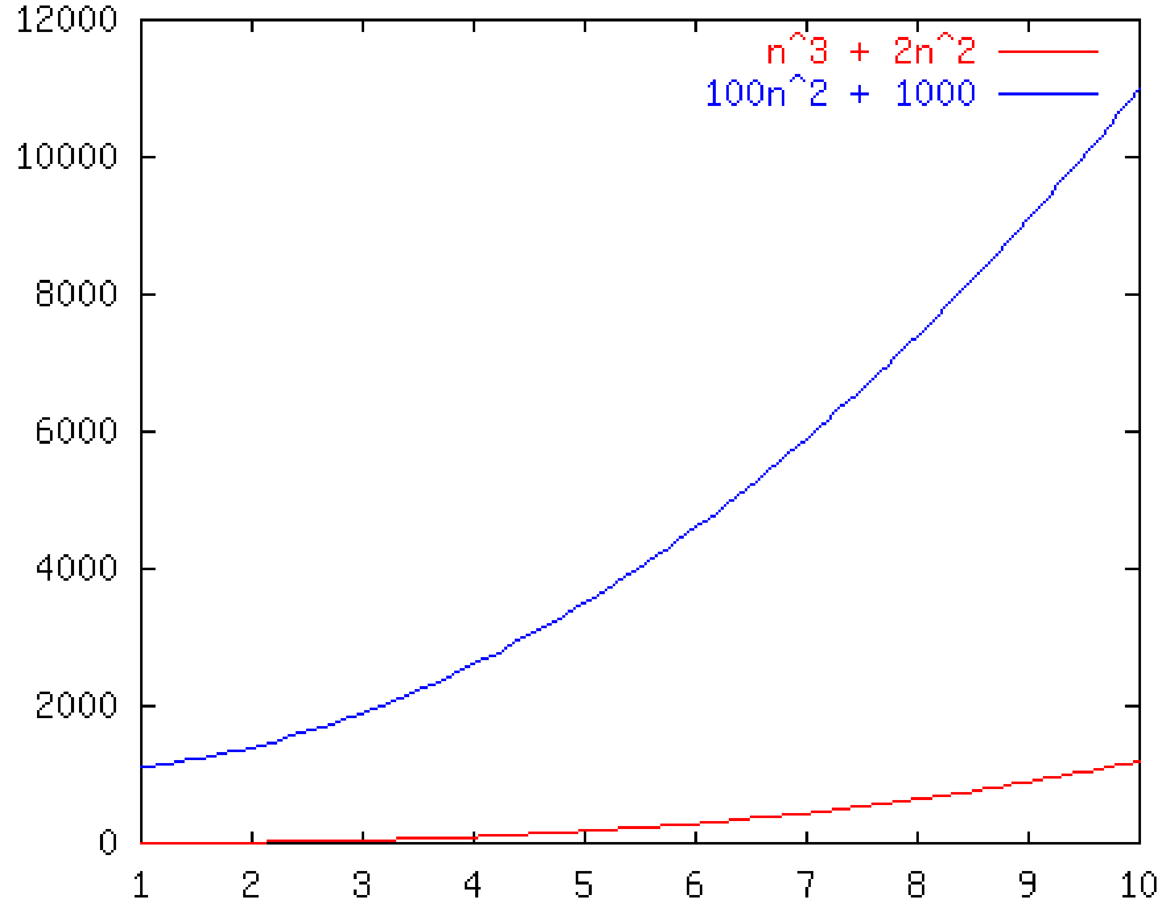
$$\therefore \log AB = \log A + \log B$$

- Similarly:
  - log(A/B) = log A – log B
  - log(A^B) = B log A
- Any log is equivalent to log-base-2

Order Notation: Intuition

$f(n) = n^3 + 2n^2$

$g(n) = 100n^2 + 1000$



Although not yet apparent, as n gets "sufficiently large", $f(n)$ will be "greater than or equal to" $g(n)$

Definition of Order Notation

- Upper bound: $T(n) = O(f(n))$ Big-O

  Exist positive constants c and n' such that

  $T(n) \leq c\ f(n)$ for all $n \geq n'$

- Lower bound: $T(n) = \Omega(g(n))$ Omega

  Exist positive constants c and n' such that

  $T(n) \geq c\ g(n)$ for all $n \geq n'$

- Tight bound: $T(n) = \theta(f(n))$ Theta

  When both hold:

  $T(n) = O(f(n))$

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

Definition of Order Notation

O(f(*n*)) : <u>a set or class of functions</u>

g(n) $\in$ O(f(n)) iff there exist positive consts c and n_0 such that:

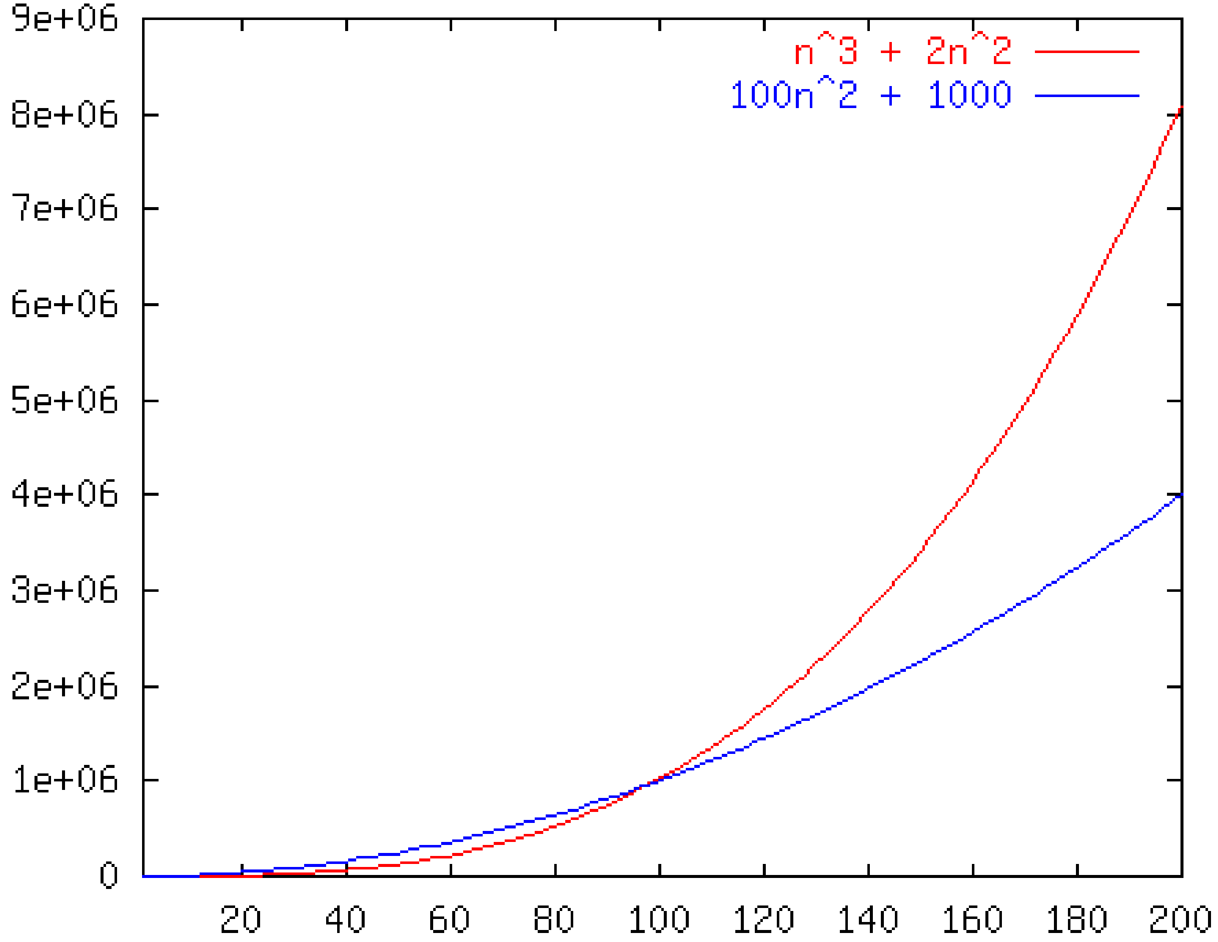
g(n) $\leq$ c f(n) for all $n \geq n_0$

Example:

$100n^2 + 1000 \leq 5\,(n^3 + 2n^2)$ for all $n \geq 19$

So g(n) $\in$ O(f(n))

Order Notation: Example



$100n^2 + 1000 \le 5\ (n^3 + 2n^2)$ for all $n \ge 19$

So $f(n) \in O(\ g(n)\ )$

Some Notes on Notation

- Sometimes you'll see

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

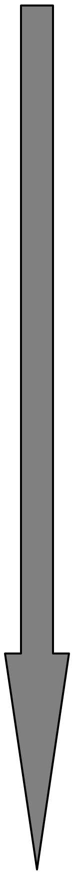
- This is equivalent to

$$g(n) \in O(\ f(n)\ )$$

- What about the reverse?

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

Big-O: Common Names



- constant: $O(1)$
- logarithmic: $O(\log n)$ ($\log_k n$, $\log n^2 \in O(\log n)$)
- linear: $O(n)$
- log-linear: $O(n \log n)$
- quadratic: $O(n^2)$
- cubic: $O(n^3)$
- polynomial: $O(n^k)$ (k is a constant)
- exponential: $O(c^n)$ (c is a constant > 1)

Meet the Family

- $O(f(n))$ is the set of all functions asymptotically less than or equal to $f(n)$
  - $o(f(n))$ is the set of all functions asymptotically strictly less than $f(n)$
- $\Omega(f(n))$ is the set of all functions asymptotically greater than or equal to $f(n)$
  - $\omega(f(n))$ is the set of all functions asymptotically strictly greater than $f(n)$
- $\theta(f(n))$ is the set of all functions asymptotically equal to $f(n)$

Meet the Family, Formally

- g(n) $\in$ O(f(n)) iff
  There exist c and n_0 such that g(n) $\leq$ c f(n) for all $n \geq n_0$
  - g(n) $\in$ o(f(n)) iff
    There exists a n_0 such that g(n) **<** c f(n) for all c and $n \geq n_0$

    Equivalent to: $\lim_{n \to \infty} g(n)/f(n) = 0$

- g(n) $\in$ Ω(f(n)) iff
  There exist c and n_0 such that g(n) $\geq$ c f(n) for all $n \geq n_0$
  - g(n) $\in$ ω(f(n)) iff
    There exists a n_0 such that g(n) **>** c f(n) for all c and $n \geq n_0$

    Equivalent to: $\lim_{n \to \infty} g(n)/f(n) = \infty$

- g(n) $\in$ θ(f(n)) iff
  g(n) $\in$ O(f(n)) and g(n) $\in$ Ω(f(n))

Big-Omega et al. Intuitively

Asymptotic Notation	Mathematics Relation
O	$\le$
Ω	$\ge$
θ	=
o	<
ω	>

Pros and Cons of Asymptotic Analysis

Perspective: Kinds of Analysis

- Running time may depend on actual data input, not just length of input
- Distinguish
  - Worst Case
    - Your worst enemy is choosing input
  - Best Case
  - Average Case
    - Assumes some probabilistic distribution of inputs
  - Amortized
    - Average time over many operations

Types of Analysis

Two <u>orthogonal</u> axes:

- Bound Flavor
  - Upper bound (O, o)
  - Lower bound (Ω, ω)
  - Asymptotically tight (θ)
- Analysis Case
  - Worst Case (Adversary)
  - Average Case
  - Best Case
  - Amortized

$16n^3\log_8(10n^2) + 100n^2 = O(n^3\log n)$

- Eliminate low-order terms
- Eliminate constant coefficients

$16n^3\log_8(10n^2) + 100n^2$

$\Rightarrow 16n^3\log_8(10n^2)$

$\Rightarrow n^3\log_8(10n^2)$

$\Rightarrow n^3(\log_8(10) + \log_8(n^2))$

$\Rightarrow n^3\log_8(10) + n^3\log_8(n^2)$

$\Rightarrow n^3\log_8(n^2)$

$\Rightarrow 2n^3\log_8(n)$

$\Rightarrow n^3\log_8(n)$

$\Rightarrow n^3\log_8(2)\log(n)$

$\Rightarrow n^3\log(n)/3$

$\Rightarrow n^3\log(n)$

- Should be started on Homework 1
- Priority Queues and Heaps up Next (relevant to Project 2)