CSE 373
Data Structures & Algorithms
Guest Lecturer: Sean Shih-Yen Liu

Lecture 04

Asymptotic Analysis (II)

Announcements

- Homework 1 due tomorrow, by 11:45pm
- Homework 2 is posted on the website, due next Friday at the beginning class. You can turn in in class or submit online.

Some Notes on Notation

Sometimes you'll see (e.g., in Weiss)

- $h(n) = O(\ f(n)\ )$

or

- $h(n)$ is $O(\ f(n)\ )$

These are equivalent to

- $h(n) \in O(\ f(n)\ )$

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)
- hyperexponential: $O(2^{2^{2^{\cdots^{2}}}})$ (a tower of n exponentials

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

Meet the Family, Formally

- $h(n) \in O(f(n))$ iff
  There exist $c>0$ and $n_0>0$ such that $h(n) \leq c\, f(n)$ for all $n \geq n_0$
- $h(n) \in o(f(n))$ iff
  There exists an $n_0>0$ such that $h(n) < c\, f(n)$ for all $c>0$ and $n \geq n_0$
  - This is equivalent to: $\lim_{n\to\infty} h(n)/f(n) = 0$
- $h(n) \in \Omega(g(n))$ iff
  There exist $c>0$ and $n_0>0$ such that $h(n) \geq c\, g(n)$ for all $n \geq n_0$
- $h(n) \in \omega(g(n))$ iff
  There exists an $n_0>0$ such that $h(n) > c\, g(n)$ for all $c>0$ and $n \geq n_0$
  - This is equivalent to: $\lim_{n\to\infty} h(n)/g(n) = \infty$
- $h(n) \in \theta(f(n))$ iff
  $h(n) \in O(f(n))$ and $h(n) \in \Omega(f(n))$
  - This is equivalent to: $\lim_{n\to\infty} h(n)/f(n) = c \neq 0$

Big-Omega et al. Intuitively

Asymptotic Notation	Mathematics Relation
O	$\leq$
Ω	$\geq$
θ	$=$
o	$<$
ω	$>$

Input Size

- Usually: length (in characters) of input

- Sometimes: value of input (if it is a number)

Complexity cases (revisited)

- **Worst-case complexity**: max # steps algorithm takes on “most challenging” input of size **N**
- **Best-case complexity:** min # steps algorithm takes on “easiest” input of size **N**

- **Average-case complexity**: avg # steps algorithm takes on *random* inputs of size **N**
- **Amortized complexity**: max total # steps algorithm takes on M “most challenging” *consecutive* inputs of size N, divided by M (i.e., divide the max total by M).

Example

- Recall the function: find(x, v, n)
- Input size: n (the length of the array)
- T(n) = “running time for size n”
- But T(n) needs clarification:
  – Worst case T(n): it runs in at most T(n) time for any x,v
  – Best case T(n): it takes at least T(n) time for any x,v
  – Average case T(n): average time over all v and x

Bounds vs. Cases

Two orthogonal axes:

- Bound Flavor
  - Upper bound (O, o)
  - Lower bound (Ω, ω)
  - Asymptotically tight (θ)
- Analysis Case
  - Worst Case (Adversary), $T_{worst}(n)$
  - Average Case, $T_{avg}(n)$
  - Best Case, $T_{best}(n)$
  - Amortized, $T_{amort}(n)$

One can estimate the bounds for any given case.

Example: Upper Bound

Claim: $n^2 + 100n = O(n^2)$

Proof: Must find c, n' such that for all $n > n'$,

$$n^2 + 100n \le cn^2$$

Let's try setting $c = 2$. Then

$$n^2 + 100n \le 2n^2$$

$$100n \le n^2$$

$$100 \le n$$

So we can set $n' = 100$ and reverse the steps above.

Using a Different Pair of Constants

Claim: $n^2 + 100n = O(n^2)$

Proof: Must find c, n' such that for all $n > n'$,

$$n^2 + 100n \le cn^2$$

Let's try setting $c = 101$. Then

$$n^2 + 100n \le 100n^2$$

$$n + 100 \le 101n \quad \text{(divide both sides by n)}$$

$$100 \le 100n$$

$$1 \le n$$

So we can set $n' = 1$ and reverse the steps above.

Example: Lower Bound

Claim: $n^2 + 100n = \Omega(n^2)$

Proof: Must find c, n' such that for all $n > n'$,

$$n^2 + 100n \geq cn^2$$

Let's try setting $c = 1$. Then

$$n^2 + 100n \geq n^2$$

$$n \geq 0$$

So we can set $n' = 0$ and reverse the steps above.

Thus we can also conclude $n^2 + 100n = \theta(n^2)$

Conventions of Order Notation

Order notation is not symmetric: write $2n^2 + n = O(n^2)$

but never $O(n^2) = 2n^2 + n$

The expression $O(f(n)) = O(g(n))$ is equivalent to

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

The expression $\Omega(f(n)) = \Omega(g(n))$ is equivalent to

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

The right-hand side is a "cruder" version of the left:

$18n^2 = O(n^2) = O(n^3) = O(2^n)$

$18n^2 = \Omega(n^2) = \Omega(n \log n) = \Omega(n)$

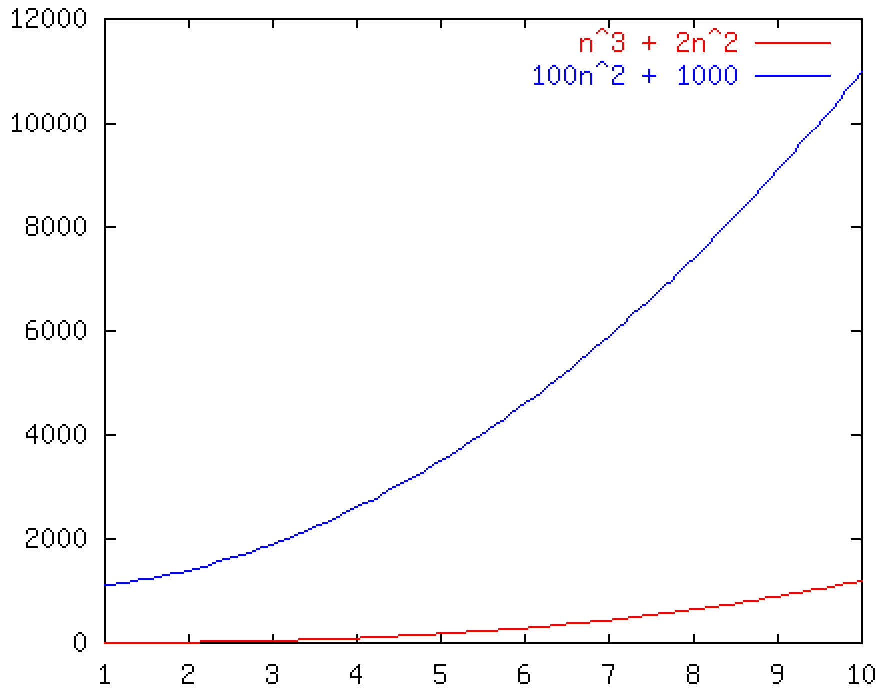
Which Function Dominates?

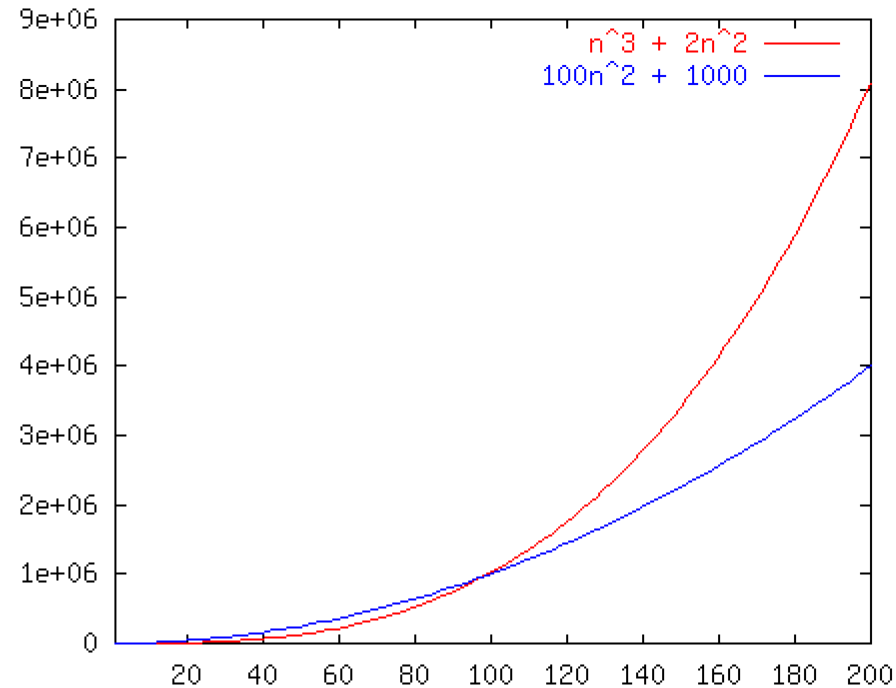
$f(n) =$	$g(n) =$
$n^3 + 2n^2$	$100n^2 + 1000$
$n^{0.1}$	$\log n$
$n + 100n^{0.1}$	$2n + 10 \log n$
$5n^5$	$n!$
$n^{-15}2^n/100$	$1000n^{15}$
$8^{2\log n}$	$3n^7 + 7n$

Question to class: is $f = O(g)$? Is $g = O(f)$?

Race I

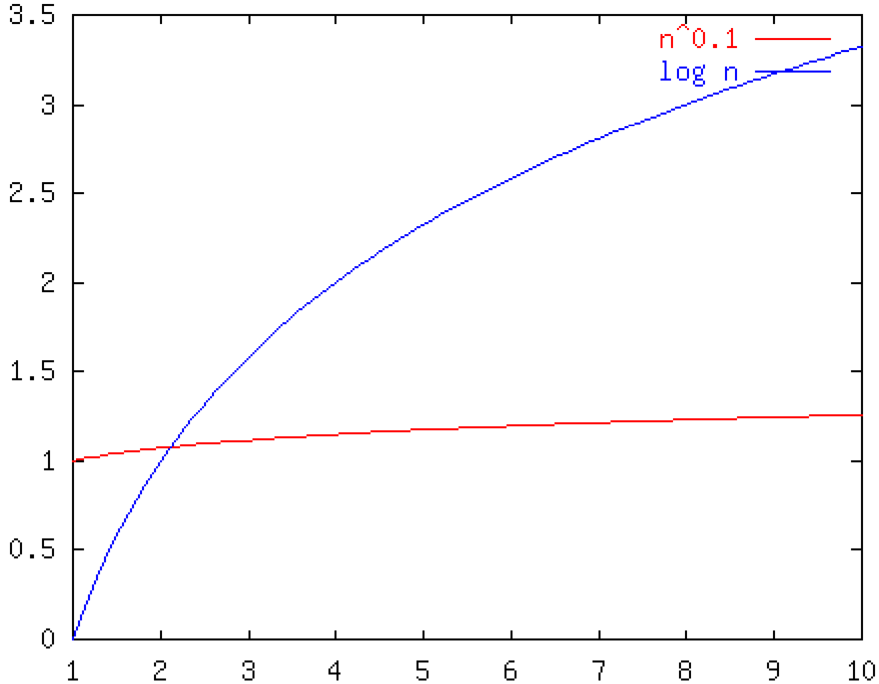
$\mathbf{f(n)= n^3+2n^2}$ vs. $\mathbf{g(n)=100n^2+1000}$

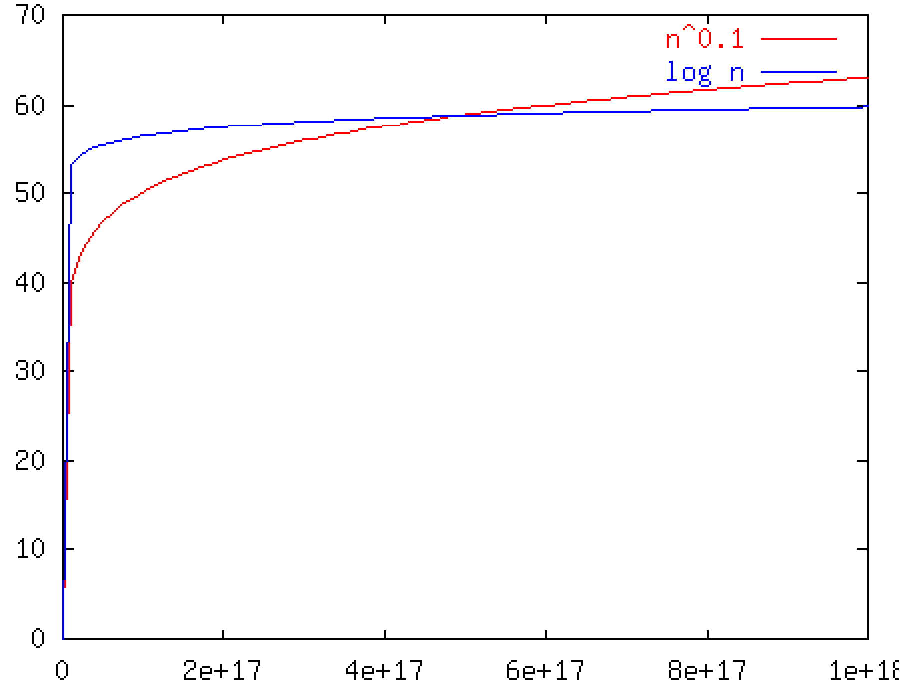




Race II

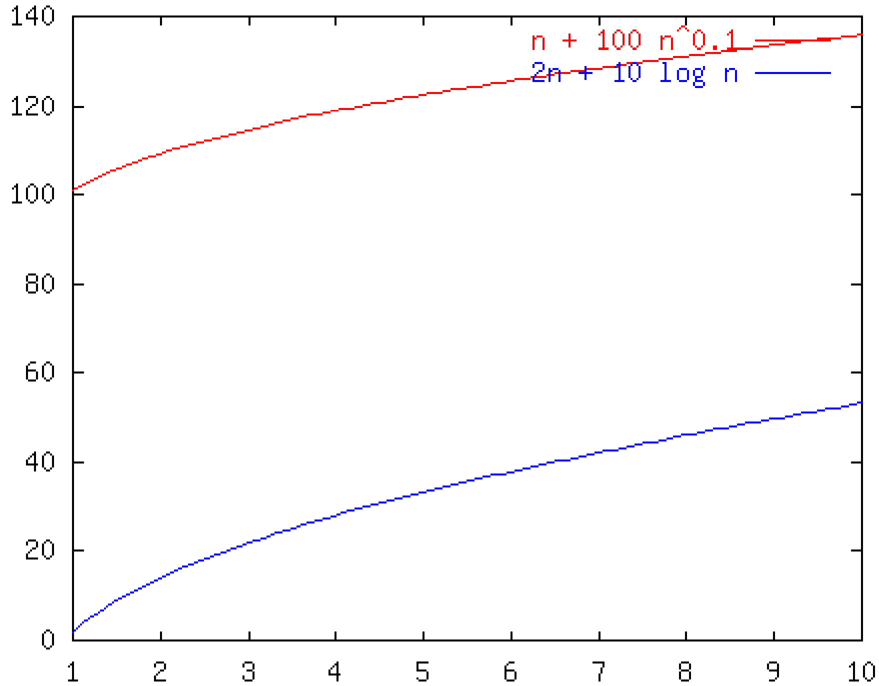
$n^{0.1}$ vs. $\log n$

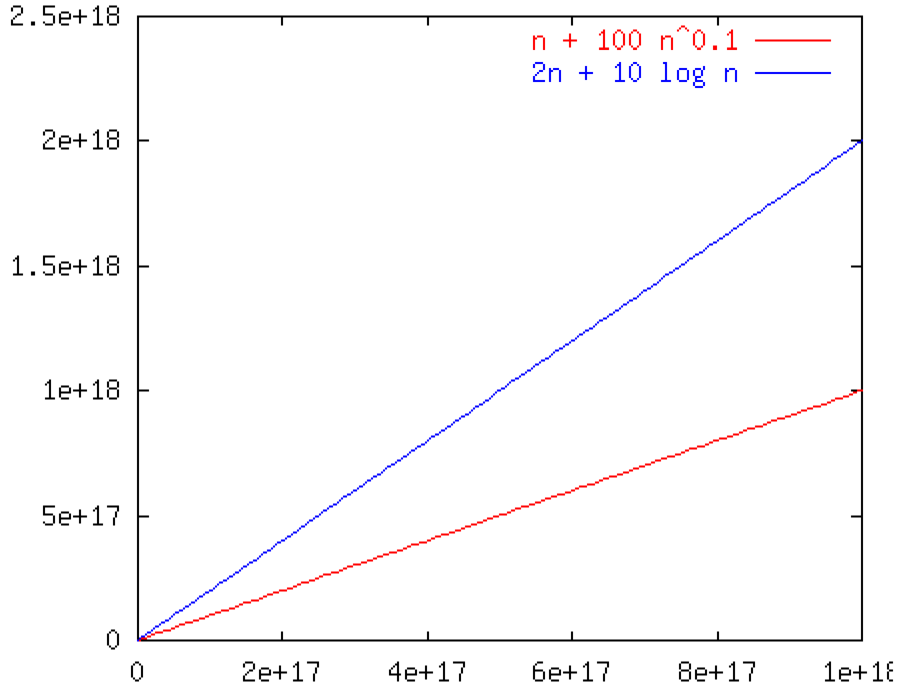




Race III

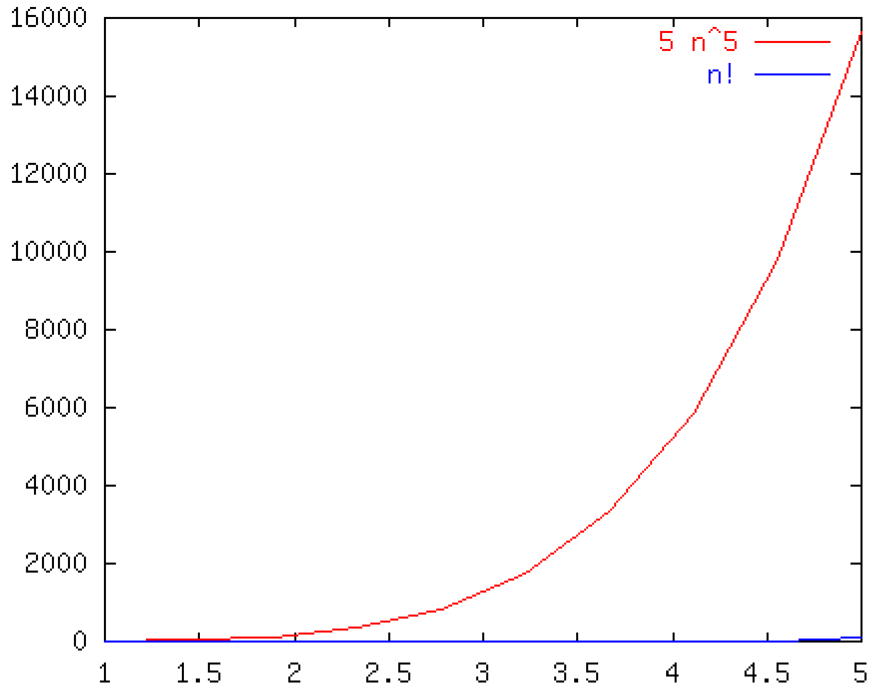
$n + 100n^{0.1}$ vs. **$2n + 10 \log n$**

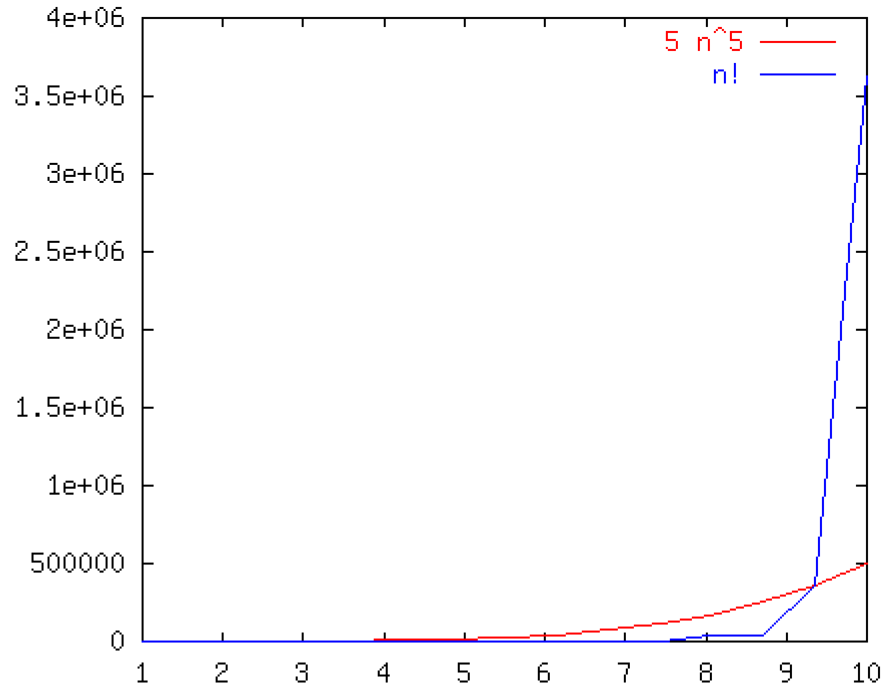




Race IV

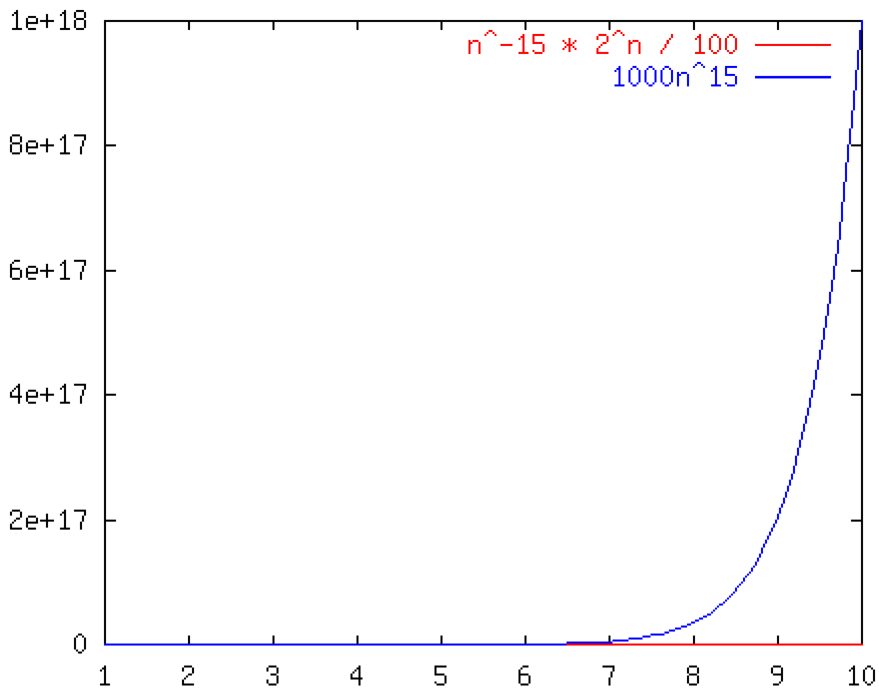
$5n^5$ vs. **n!**

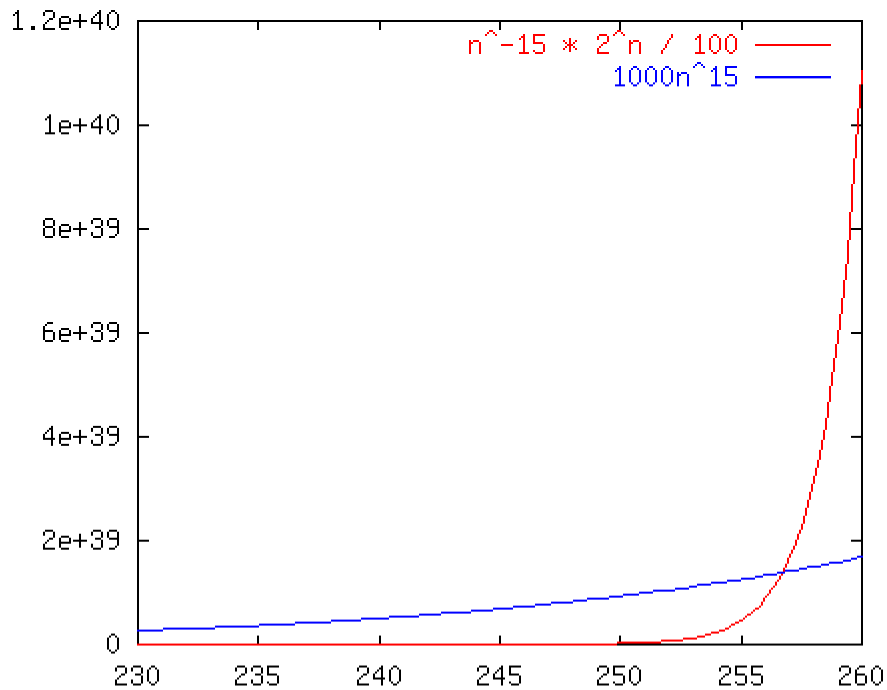




Race V

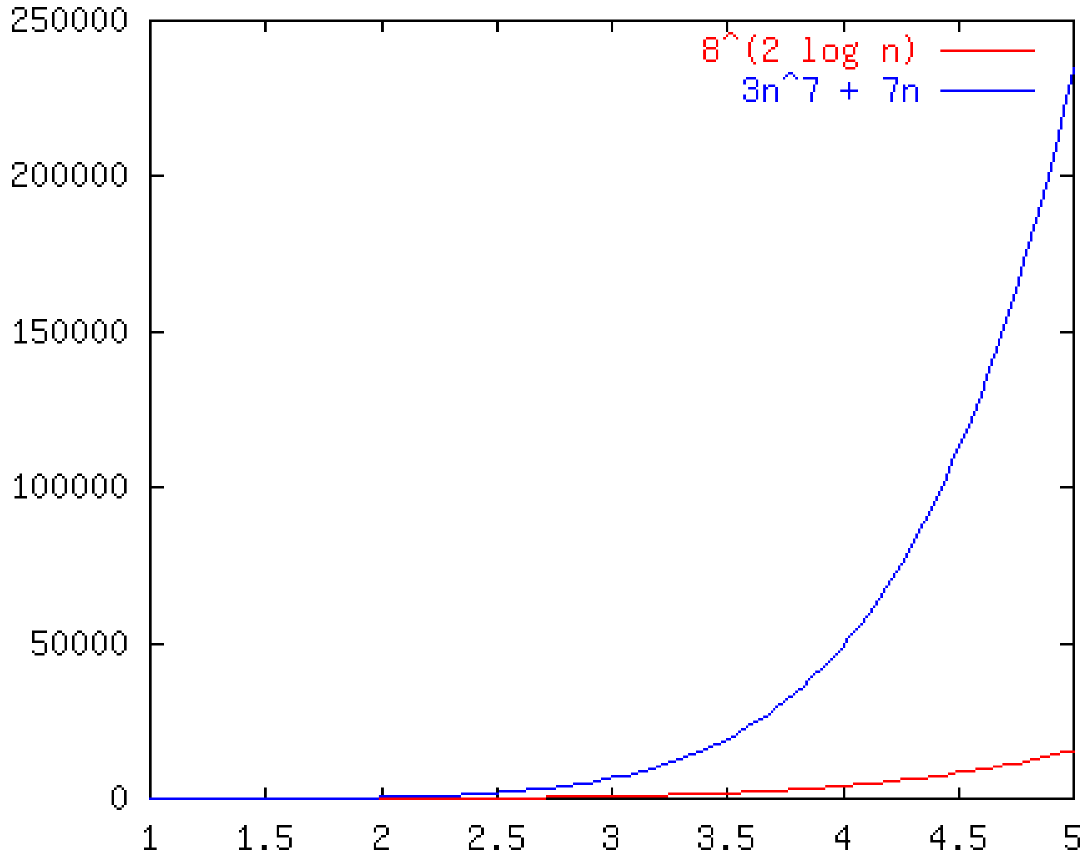
$n^{-15}2^n/100$ vs. $1000n^{15}$





Race VI

$8^{2\log(n)}$ vs. $3n^7 + 7n$



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

- Eliminate low order terms
- Eliminate constant coefficients

$$16n^3 \log_8(10n^2) + 100n^2$$
$$\Rightarrow 16n^3 \log_8(10n^2)$$
$$\Rightarrow n^3 \log_8(10n^2)$$
$$\Rightarrow n^3 \left[\log_8(10) + \log_8(n^2)\right]$$
$$\Rightarrow n^3 \log_8(10) + n^3 \log_8(n^2)$$
$$\Rightarrow n^3 \log_8(n^2)$$
$$\Rightarrow n^3 2 \log_8(n)$$
$$\Rightarrow n^3 \log_8(n)$$
$$\Rightarrow n^3 \log_8(2) \log(n)$$
$$\Rightarrow n^3 \log(n)$$

Sums and Recurrences

Often the function f(n) is not explicit but expressed as:

- A sum, or
- A recurrence

Need to obtain analytical formula first

Sums

$$f(n) = 1 + 2 + \ldots + n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

$$f(n) = 1 + 3 + 5 + \ldots + (2n-1) = \sum_{i=1}^{n} (2i-1) = n^2 = O(n^2)$$

$$f(n) = 1^2 + 2^2 + \ldots + n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$$

$$f(n) = 1^3 + 2^3 + \ldots + n^3 = O(?)$$

$$f(n) = 1^4 + 5^4 + 9^4 + \ldots + (4n-3)^4 = \sum_{i=1}^{n} (4i-3)^4 = O(??)$$

More Sums

$$f(n) = 1 + 3 + 3^2 + ... + 3^n = \sum_{i=1}^{n} 3^i = \frac{3^{n+1} - 1}{3 - 1} = O(3^n)$$

Sometimes sums are easiest computed with integrals:

$$f(n) = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i} \approx 1 + \int_1^n \frac{1}{x} dx = 1 + \ln(n) - \ln(1) = O(\ln(n))$$

$$f(n) = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + ... + \frac{1}{n^2} = \sum_{i=1}^{n} \frac{1}{i^2} \approx 1 + \int_1^n \frac{1}{x^2} dx = 1 + \frac{1}{1} - \frac{1}{n} = O(1)$$

Recurrences

- $f(n) = 2f(n-1) + 1, \; f(0) = T$
- Telescoping

$$
\begin{array}{llr}
\Rightarrow f(n)+1 & = 2(f(n-1)+1) & \\
f(n-1)+1 & = 2(f(n-2)+1) & \times 2 \\
f(n-2)+1 & = 2(f(n-3)+1) & \times 2^2 \\
\ldots\ldots & & \\
f(1) + 1 & = 2(f(0) + 1) & \times 2^{n-1}
\end{array}
$$

$$\Rightarrow f(n)+1 = 2^n(f(0)+1) = 2^n(T+1)$$

$$\Rightarrow f(n) = 2^n(T+1) - 1$$

Recurrences

- Fibonacci: f(n) = f(n-1)+f(n-2), f(0)=f(1)=1

  ➔ try $f(n) = A\,c^n$ What is c ?

  $A\,c^n = A\,c^{n-1} + A\,c^{n-2}$

  $c^2 - c - 1 = 0$

$$c_{1,2} = \frac{1 \pm \sqrt{1+4}}{2} = \frac{1 \pm \sqrt{5}}{2}$$

$$f(n) = A\left(\frac{1+\sqrt{5}}{2}\right)^n + B\left(\frac{1-\sqrt{5}}{2}\right)^n = O\left(\frac{1+\sqrt{5}}{2}\right)^n$$

Constants A, B can be determined from f(0), f(1) – not interesting for us for the Big O notation

Recurrences

- $f(n) = f(n/2) + 1, \quad f(1) = T$
- Telescoping:

  $f(n) = f(n/2) + 1$

  $f(n/2) = f(n/4) + 1$

  . . .

  $f(2) = f(1) + 1 = T + 1$

  ➔ $f(n) = T + \log n = O(\log n)$