CSE 326: Data Structures

Asymptotic Analysis (Continued)

Review Solving Recurrences

1. Determine the recurrence relation. What is the base case(s)?
2. "Expand" the original relation to find an equivalent general expression *in terms of the number of expansions*.
3. Find a closed-form expression by setting *the number of expansions* to a value which reduces the problem to a base case

---

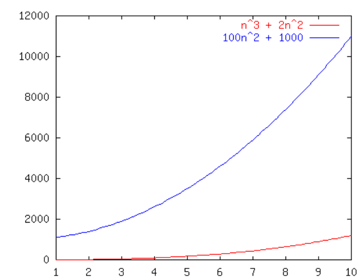
Asymptotic Analysis

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

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 constants c and n' such that
  $T(n) \le c\, f(n)$ for all $n \ge n'$
- Lower bound: $T(n) = \Omega(g(n))$ Omega
  Exist constants c and n' such that
  $T(n) \ge c\, g(n)$ for all $n \ge n'$
- Tight bound: $T(n) = \theta(f(n))$ Theta
  When both hold:
  $T(n) = O(f(n))$
  $T(n) = \Omega(f(n))$

Order Notation: Definition

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

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

$g(n) \le c\, f(n)$ for all $n \ge n_0$

Example: $g(n) = 1000n$ vs. $f(n) = n^2$

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

Pick: n0 = 1000, c = 1

Notation Notes

Note: Sometimes, you'll see the notation:

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

This is equivalent to:

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

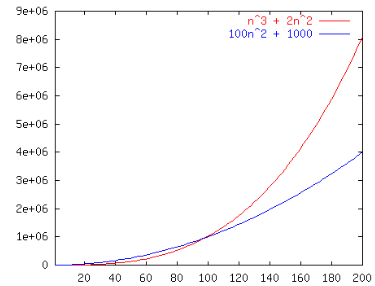
However: The notation

$O(f(n)) = g(n)$ is meaningless!

(in other words big-O is not symmetric)

7

Order Notation: Example



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

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

8

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)

9

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 $f(n)$

10

Meet the Family, Formally

- $g(n) \in O(\,f(n)\,)$ iff
  There exist c and n_0 such that $g(n) \le c\,f(n)$ for all $n \ge n_0$
  - $g(n) \in o(\,f(n)\,)$ iff
    There exists a n_0 such that $g(n) < c\,f(n)$ for all c and $n \ge n_0$
    Equivalent to: $\lim_{n\to\infty} g(n)/f(n) = 0$
- $g(n) \in \Omega(\,f(n)\,)$ iff
  There exist c and n_0 such that $g(n) \ge c\,f(n)$ for all $n \ge n_0$
  - $g(n) \in \omega(\,f(n)\,)$ iff
    There exists a n_0 such that $g(n) > c\,f(n)$ for all c and $n \ge n_0$
    Equivalent to: $\lim_{n\to\infty} g(n)/f(n) = \infty$
- $g(n) \in \theta(\,f(n)\,)$ iff
  $g(n) \in O(\,f(n)\,)$ and $g(n) \in \Omega(\,f(n)\,)$

11

Big-Omega et al. Intuitively

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

12

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

13

Types of Analysis

Two orthogonal axes:

- bound flavor
  - upper bound (O, o)
  - lower bound (Ω, ω)
  - asymptotically tight (θ)
- analysis case
  - worst case (adversary)
  - average case
  - best case
  - “amortized”

14

Algorithm Analysis Examples

- Consider the following program segment:

```
x:= 0;
for i = 1 to N do
  for j = 1 to i do
    x := x + 1;
```

- What is the value of x at the end?

15

Analyzing the Loop

- Total number of times x is incremented is executed =

$$1+2+3+\ldots=\sum_{i=1}^{N} i=\frac{N(N+1)}{2}$$

- Congratulations - You’ve just analyzed your first program!
  - Running time of the program is proportional to N(N+1)/2 for all N
  - Big-O ??

16

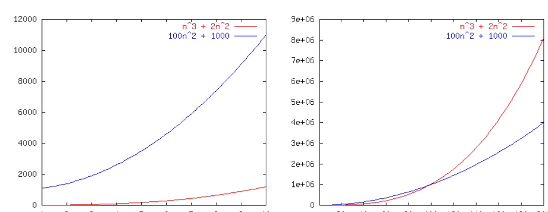
Which Function Grows Faster?

$n^3 + 2n^2$ vs. **$100n^2 + 1000$**

17

Which Function Grows Faster?

$n^3 + 2n^2$ vs. **$100n^2 + 1000$**



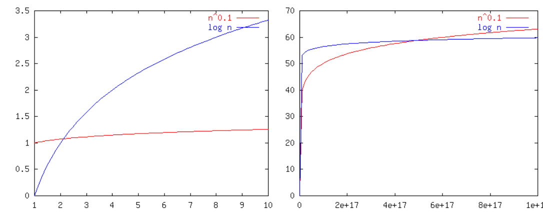
18

Which Function Grows Faster?

$n^{0.1}$ vs. log n

Which Function Grows Faster?

$n^{0.1}$ vs. log n

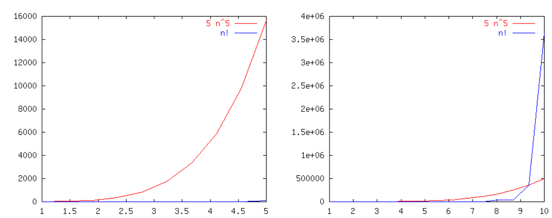


Which Function Grows Faster?

$5n^5$ vs. n!

Which Function Grows Faster?

$5n^5$ vs. n!



Nested Loops

```
for i = 1 to n do
  for j = 1 to n do
    if (cond) {
       do_stuff(sum)
    } else {
       for k  = 1 to n*n
          sum += 1
```


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

- Eliminate low order terms
- Eliminate constant coefficients

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