CS216: Program and Data Representation
University of Virginia Computer Science
Spring 2006 David Evans

Lecture 2: Orders of Growth

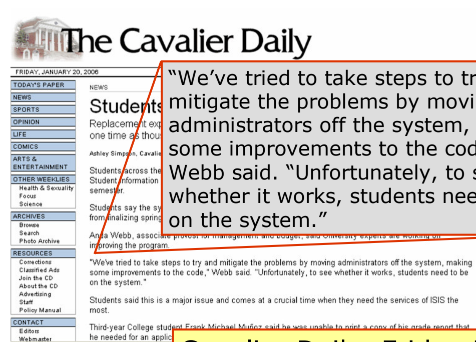
http://www.cs.virginia.edu/cs216

Menu

- Predicting program properties
- Orders of Growth: O, Ω
- Course Survey Results

Everyone should have received an email:
1. Informing you of your PS1 partner
2. Giving the section room locations
3. Explaining that PS1 is now due Monday, Jan 30

Predicting Program Properties



"We've tried to take steps to try and mitigate the problems by moving administrators off the system, making some improvements to the code," Webb said. "Unfortunately, to see whether it works, students need to be on the system."

Cavalier Daily, Friday Jan 20
(http://www.cavalierdaily.com/CVArticle.asp?ID=25481)

Predicting Running Time

- Experimentally: measure how long the program takes on particular inputs
  - Generalize, extrapolate to other input sizes
- Analytically: figure out how the amount of work scales with the input size
  - Understand what the program does
- Need to combine with experimental results on some inputs and analytical understanding to make good predictions

---

Order Notation

- Four notations: $O(f)$, $\Omega(f)$, $o(f)$, $\Theta(f)$
- These notations define **sets of functions**
  - Functions from positive integer to real
- When we say, "Algorithm A is $O(n)$" we mean,

  running time of A $\in O(n)$

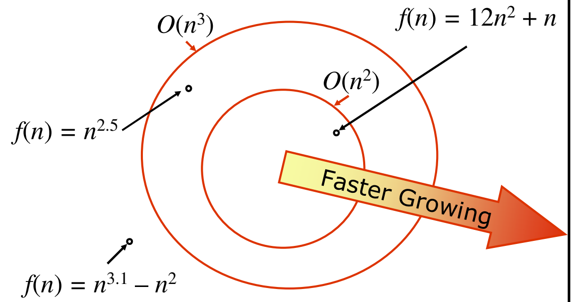
  where n measures the input size to A.

Big O

- Intuition: the set $O(f)$ is the set of functions that *grow* **no faster than** f
  - More formal definition coming soon
- Asymptotic growth rate
  - As input to f approaches infinity, how fast does value of f increase
  - Hence, only the fastest-growing term in f matters:

$$O(n^3) \subset O(12n^2 + n)$$
$$O(n) \equiv O(63n + \log n - 423)$$

Examples



Formal Definition

$f \in O(g)$ means:

There are *positive* constants c and n_0 such that

$$f(n) \le cg(n)$$

for all values $n \ge n_0$.

O Examples

$f(n) \in O(g(n))$ means: there are positive constants c and n_0 such that $f(n) \le cg(n)$ for all values $n \ge n_0$.

$x \in O(x^2)$?	Yes, $c = 1$, n_0=2 works fine.
$10x \in O(x)$?	Yes, $c = 11$, n_0=2 works fine.
$\cancel{x^2 \in O(x)?}$	No, no matter what c and n_0 we pick, $cx^2 > x$ for big enough x

Question

Given $f \in O(h)$ and $g \notin O(h)$ which of these are true:

a. For **all** positive integers m,
 $f(m) < g(m)$.

b. For **some** positive integer m,
 $f(m) < g(m)$.

c. For some positive integer m_0, and all positive integers $m > m_0$,
 $f(m) < g(m)$. (left as problem for Exam 1)

a is false: Prove by Counter-Example

$f(n) \in O(h(n))$ and $g(n) \notin O(h(n))$

a. For all positive integers m, $f(m) < g(m)$.

Pick $h(n) = n^2$, $f(n) = 5n^2$, $g(n) = n^3$.
For $m = 2$, $f(m) = 20 > 8 = g(m)$.
Therefore, a is false.

> $f(n) \in O(g(n))$ means there are *positive* constants c and n_0 such that $f(n) \le cg(n)$ for all values $n \ge n_0$.

b is true: Intuition

If $f \in O(h)$ and $g \notin O(h)$ then, for **some** positive integer m, $f(m) < g(m)$.

g must grow faster than h, otherwise g would be in $O(h)$.

f must grow no faster than h, since $f \in O(h)$

So, if g grows faster than h, but f grows as slow or slower than h, eventually, $g(n) > f(n)$ so for some m, $f(m) < g(m)$.

b: Proof by Contradiction

If $f \in O(h)$ and $g \notin O(h)$ then, for **some** positive integer m, $f(m) < g(m)$.

1. $f \in O(h) \Rightarrow$ there are *positive* constants c and n_0 such that $f(n) \leq ch(n)$ for all values $n \geq n_0$
2. $g \notin O(h) \Rightarrow$ there are **no** *positive* constants c_1 and n_1 such that $g(n) \leq c_1 h(n)$ for all values $n \geq n_1$. So, for **all** *positive* constants c_2, $g(q) \leq c_2 h(q)$ for some value q.

b: Proof by Contradiction

If $f \in O(h)$ and $g \notin O(h)$ then, for **some** positive integer m, $f(m) < g(m)$.

Suppose statement is false.

Then, for all positive k, $f(k) \geq g(k)$

From (1), $\exists c\ \exists n_0$ such that $\forall\ n > n_0$, $f(n) \leq ch(n)$

From (2), $\forall c_1\ \exists n_1$ such that $\forall\ n > n_1$, $g(n) > c_1 h(n)$

Combining, $\exists c\ \forall c_1\ \exists n_2$ such that $\forall\ n > n_2$

$\boxed{ch(n) > c_1 h(n)}$ Note: $n_2 \leq \max(n_0, n_1)$

This is a contradiction! Only works if $c = \text{infinity}$, but c must be a positive integer

Lower Bound: Ω (Omega)

$f(n)$ is $\Omega(g(n))$ means:

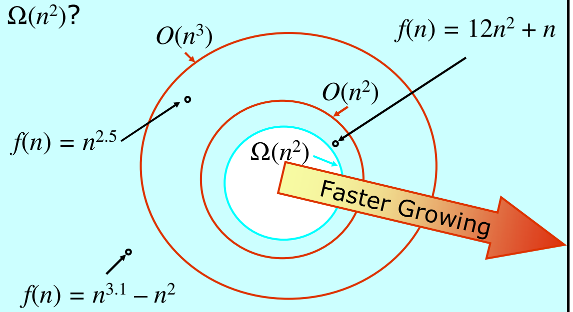
There are positive constants c and such that

$$f(n) \geq cg(n)$$

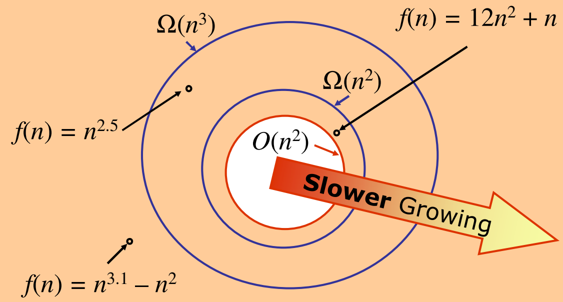
for all $n \geq n_0$.

Difference from O – this was $\leq$

Where is $\Omega(n^2)$?



Inside-Out



Survey Results Summary

See course web site for more detailed results

Honor Questions

- How much faith do you think we should put in the honor system for this class?
  - 30 Should have complete trust in honor system
  - 43 Enough to have take-home exams
  - 6 A little, but don't trust take-home exams
  - 1 Don't trust the students at all, need to police everything

73/80

Exam 1 will be take home

Honor Disadvantage?

- Do you feel you are at a disadvantage if you follow the course honor policy strictly?
  - 69 no
  - 11 yes

I hope the majority answer here will help convince the 11 “yes” answered they are not really at a disadvantage. Long term, being honorable is **never** a disadvantage.

Honor Reporting

If you observed a classmate cheating on a take-home exam, what would you do?

36 Report the student anonymously to the course staff
20 Confront the student
11 Report the student to the course staff
8 Nothing
3 Initiate an honor charge
2 No Selection

Course Pledge disallows this now.
If you can’t handle this, don’t sign the course pledge and take a different class.

Course Pledge

- Read this carefully – you are expected to know it and follow it
- Only pledge you need to sign this semester
- Requires:
  - No lying, cheating, or stealing
  - Helping your classmates learn
  - No toleration of dishonorable behavior
  - Helping the course staff improve the course

Programming Self-Rating

- 4 among best
- 26 above average
- 41 about average
- 8 a little below, far below

The programming you will do in this class is different enough from what you have done previously, that you probably don’t really know.

Everyone should be confident you will do well in this class. You don’t need to be a super code hacker to ace this class.

Programming Languages

- **Python:**
  - **74 Not at all**
  - 6 Some familiarity
- **Any assembly language:**
  - **72 Not at All**
  - 7 Some familiarity
  - 1 Lots of experience

Very few of you have experience with the language we use in this class (that is part of why we use them). You should not be worried about this!

Survey Results

- More results (as well as my answers to the questions you asked) are posted on the course web page

Charge

- Sections meet today and tomorrow
  - Order Notation practice
  - Recurrence Relations
- Wednesday: Levels of Abstraction, Introducing Lists
  - Read Chapter 3 in the textbook