CS 173: Discrete Structures

Eric Shaffer
Office Hour: Wed. 12-1, 2215 SC
shaffer1@illinois.edu

Announcements

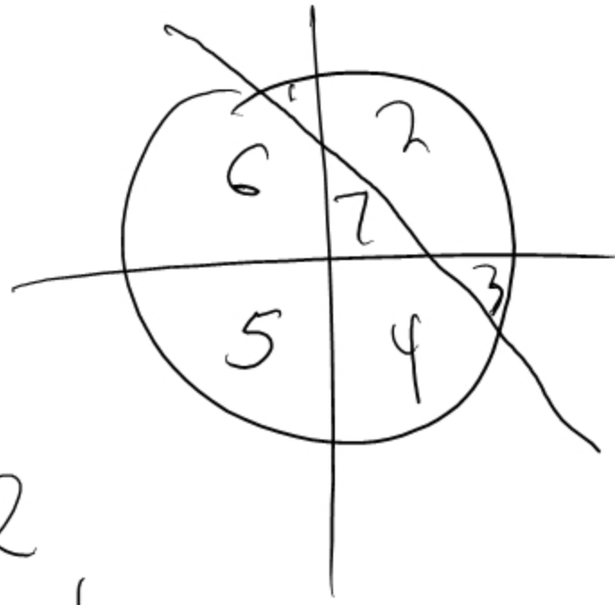
- Friday 3/13 class cancelled (due to EOH)
- HW still due 3/13
  - Hand-in at 3229 Siebel at before noon at the latest
- Quiz **Next** Wed. 3/18
  - Further updates as we get closer to that date..
- Today:
  - Measuring growth of functions
    - Section 3.2

Quick look back at recurrence relations

T(n) = number of pieces of pizza as a function of the number of cuts

Pizza cutting rules:

1. Every new cut intersects all previous cuts
2. Only two cuts intersect at one point
3. All intersections occur inside the pizza boundary (imagine an infinite pizza…)



$T(0) = 1$

$T(1) = 2$

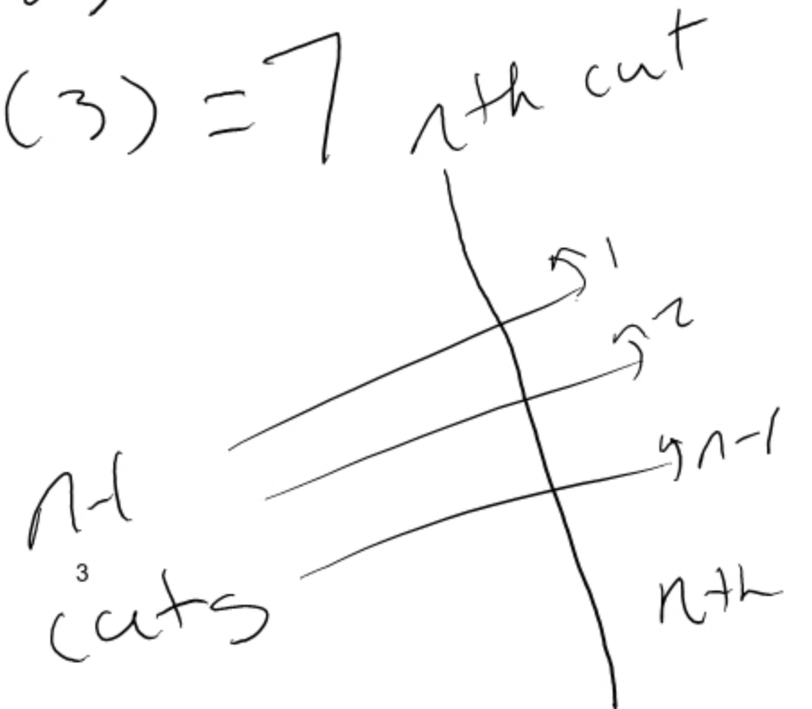
$T(2) = 4$

$T(3) = 7$

What is ~~T(n)~~?

$$T(n) = T(n-1) + n$$

$$T(0) = 1$$



Putting things together

By May (preferably sooner) you will be able to:

analyze the computational efficiency of algorithms

What this means:

we describe efficiency as a function

f(n) = number of operations an algorithm will perform

n is the "size" of the input

Tell me what f(n) is

You will be able to use recurrence relations to

Describe the efficiency of recursive algorithms

Algorithm Analysis

Algorithm analysis is concerned with:

- Typically, analyzing the efficiency of an algorithm
  - How much time will it take?
  - How much memory will it use?
- Often, efficiency is related the size of the input
  - How many comparisons to find max of n numbers?
- Typically ignore constant factors
  - different machines have different speed/cycle
- Interested in large values of n
  - Like asymptotic analysis
- We'll use "big-Oh" notation to describe functional growth
  - Created by Bachmann (1892) introduced to CS by Knuth (1960s?)

Algorithm analysis versus performance analysis

Algorithm analysis is distinct from **performance analysis**

Performance analysis is "practical" (e.g. constants often matter)

- Where are the bottlenecks?
- What part of the code is worth optimizing?
- What are the effects of the memory hierarchy?

Algorithm analysis is theoretical

- It is one, but not the only important tool, for producing efficient code
- A theoretically "good" algorithm is necessary but not sufficient

Familiar functions

Polynomials:

$$f(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x^1 + a_nx^0$$

Ex: $f(x) = x^3 - 2x^2 + 15$

Exponentials:

$$f(x) = c^{dx}$$

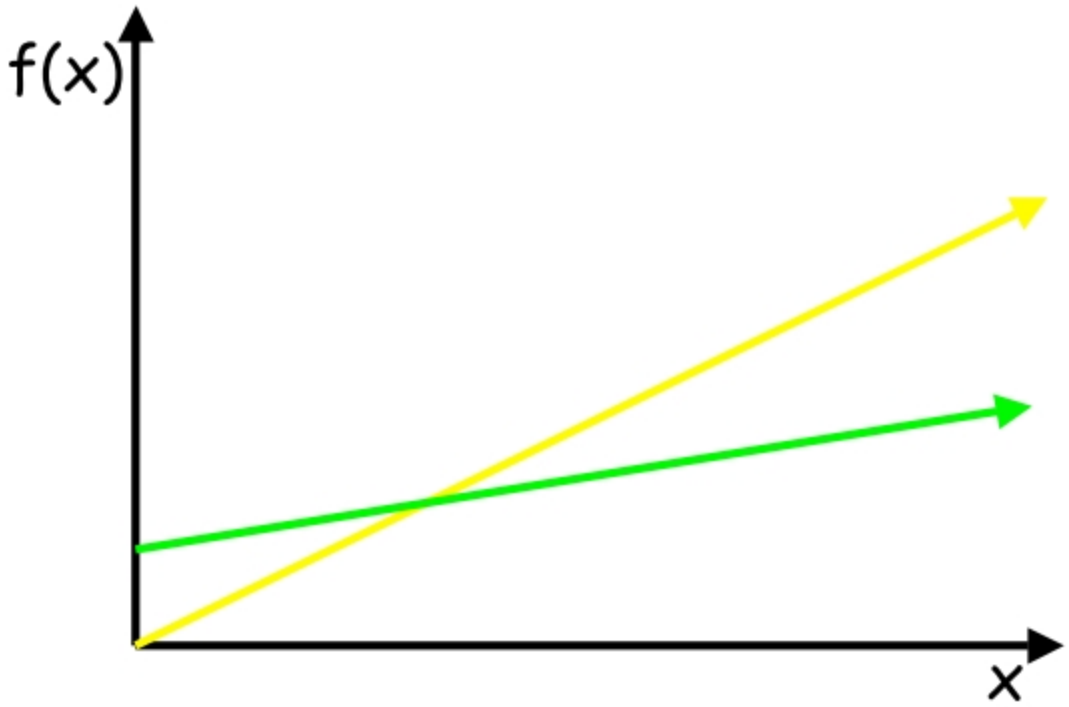
Ex: $f(x) = 3^{10x}$, $f(x) = e^x$

Logarithms:

$$\log_2 x = y, \text{ where } 2^y = x.$$

Growth of functions

Describe these functions:

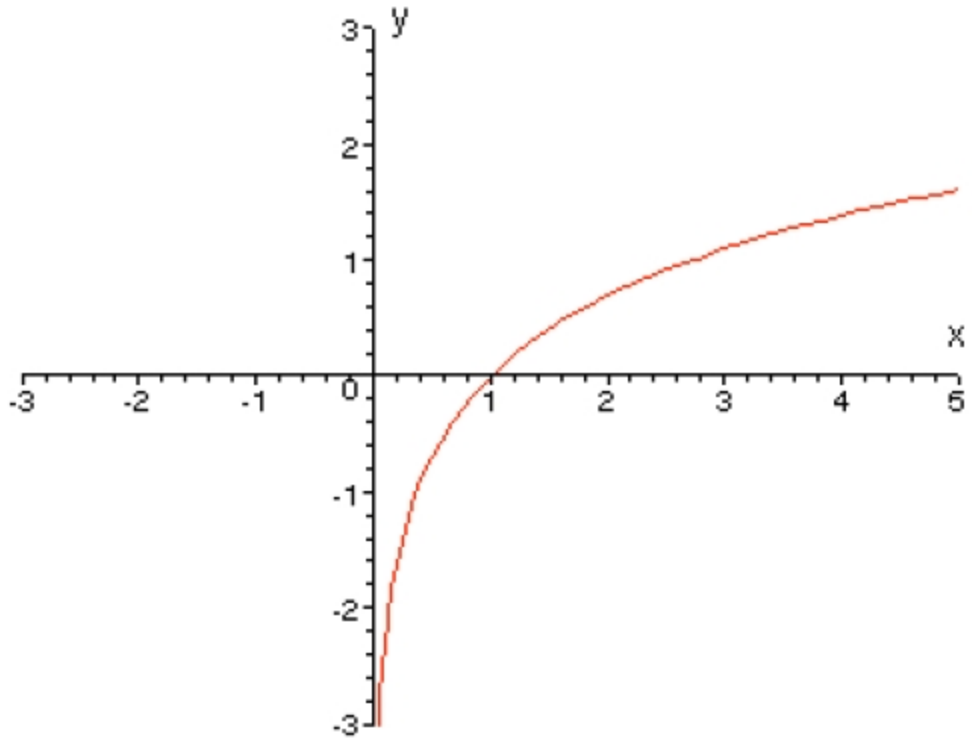


Linear functions

Every unit increase in x results in the same increase in f(x).

Growth of Functions

Describe this function:

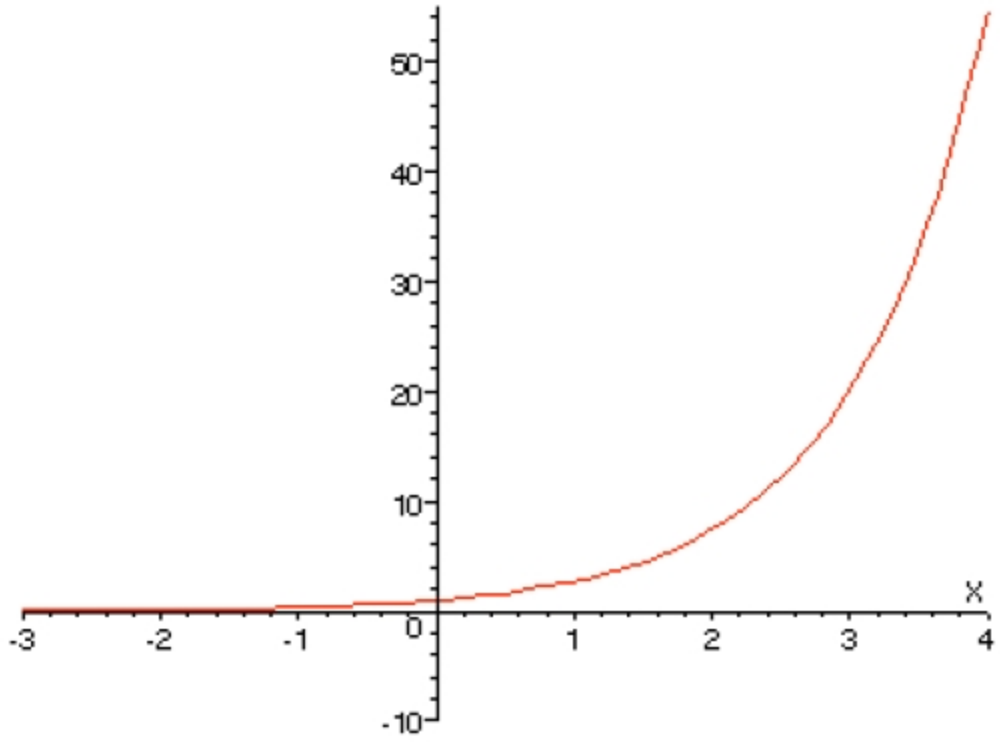


Logarithmic function

Very slow growing.

Growth of Functions

Describe this function:

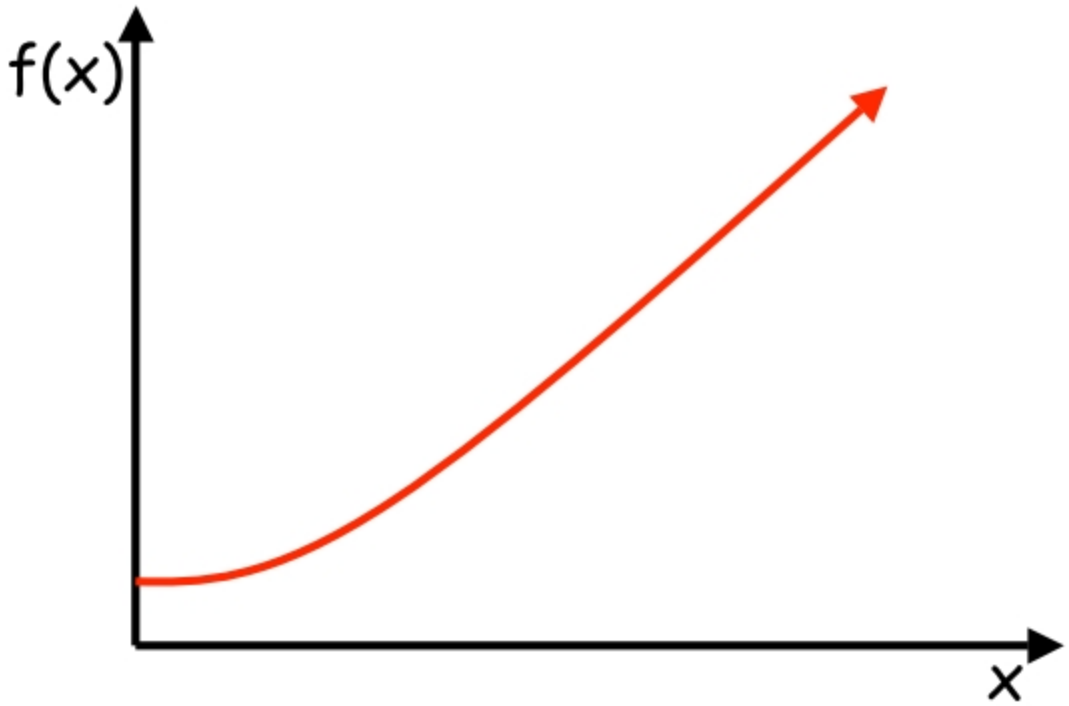


Exponential function

Very fast growing.

Growth of Functions

Describe this function:



Quadratic function

x^2

Polynomial

Growth of functions

Important definition:

For functions f and g we write

$f(x) = O(g(x))$

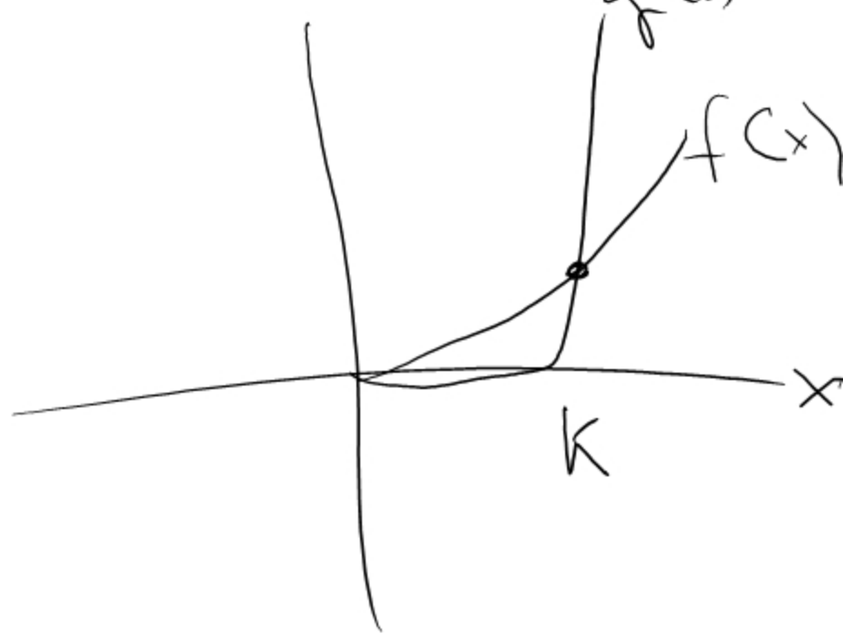
We say "f(x) is big O of g(x)"

to denote

$\exists\ c,k$ so that $\forall\ x>k$, $f(x) \le c \cdot g(x)$

Recipe for proving $f(x) = O(g(x))$: find a c and k so that the inequality holds.

$f(x) \stackrel{?}{=} g(x)$ to have co-domain of $\mathbb{R}^{\geq 0}$

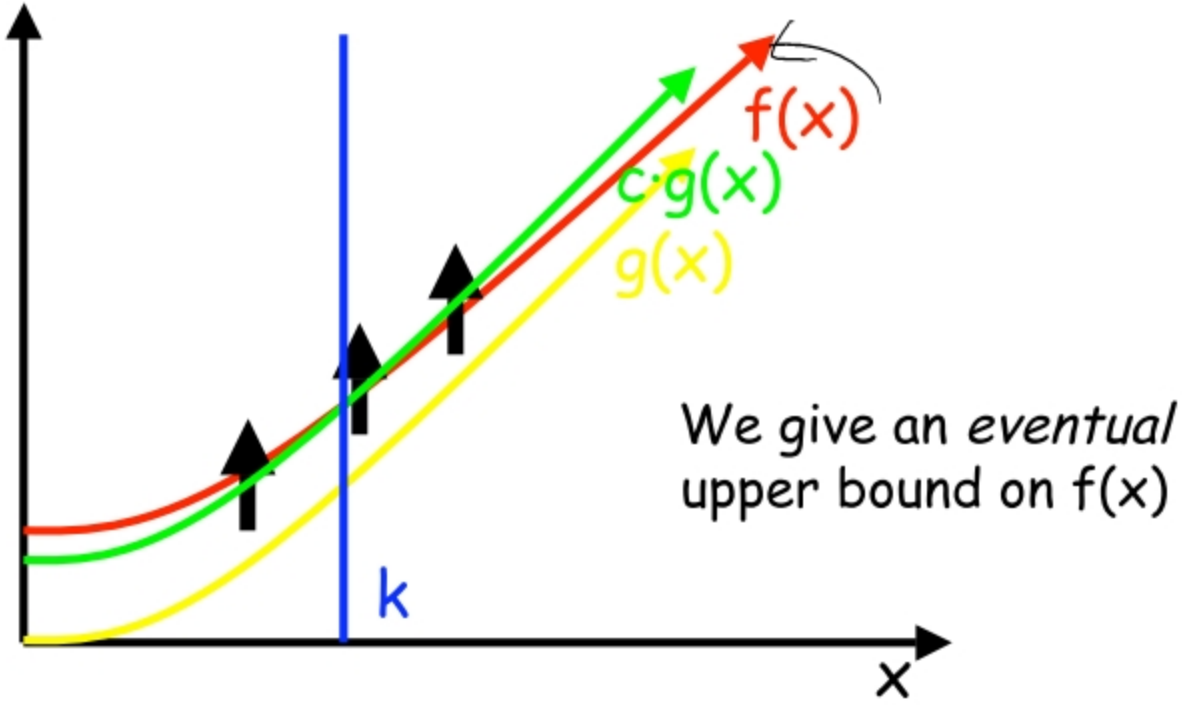


Growth of functions

$f(x) = O(g(x))$

iff

$\exists\ c,k$ so that $\forall\ x>k,\ f(x) \leq c \cdot g(x)$



We give an *eventual* upper bound on f(x)

Growth of functions (examples)

$f(x) = O(g(x))$
iff
$\exists\ c,k$ so that $\forall\ x>k,\ f(x) \le c \cdot g(x)$

$3n = O(15n)$ since $\forall\ n>0,\ 3n \le 1 \cdot 15n$

k c

Growth of functions (examples)

$f(x) = O(g(x))$
iff
$\exists\ c,k$ so that $\forall\ x>k,\ f(x) \leq c \cdot g(x)$

$15n = O(3n)$ since $\forall\ n>$ 0, $15n \leq$ 5 $\cdot 3n$
k c

Growth of functions (examples)

$f(x) = O(g(x))$
iff
$\exists\ c,k$ so that $\forall\ x>k,\ f(x) \le c \cdot g(x)$

$x^2 = O(x^3)$ since $\forall\ x > \underline{1}$, $x^2 \le x^3$

$k = 1 \quad c = 1$

Growth of functions (examples)

$f(x) = O(g(x))$
iff
$\exists\ c,k$ so that $\forall\ x>k,\ f(x) \le c \cdot g(x)$

$1000x^2 = O(x^2)$ since $\forall\ x >$ 0, $1000x^2 \le$ 1000$\cdot x^2$

k C

Growth of functions (examples)

$f(x) = O(g(x))$
iff
$\exists\ c,k$ so that $\forall\ x>k,\ f(x) \le c \cdot g(x)$

Prove that $x^2 + 100x + 100 = O((1/100)x^2)$

$x^2 + 100x + 100 \le 201x^2$ when $x > 1$

$\le 20100 \cdot (1/100)x^2$

$k = 1,$
$c = 20100$

$|x^2| + |100x| + |100| \le$
$|x^2| + |100x^2| + |100x^2|$
$201x^2$

Growth of functions (examples)

Prove that $5x + 100 = O(x/2)$

Similar problem, different technique.

Need $\forall x > \_\_\_\_, 5x + 100 \leq \_\_\_\_ \cdot x/2$

Try c = 10

$\forall x > \_\_\_\_, 5x + 100 \leq 10 \cdot x/2$

Nothing works for k

Try c = 11

$\forall x > \_\_\_\_, 5x + 100 \leq 5x + x/2$

$\forall x > 200, 100 \leq x/2$

k = 200,
c = 11

Growth of functions

Guidelines:

- In general, only the largest term in a sum matters.

$$a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x^1 + a_nx^0 = O(x^n)$$

- n dominates lg n.

$$n^5 \lg n = O(n^6)$$

- List of common functions in increasing O() order:

$$1\ ,\ \lg n,\ n\ ,(n \lg n)\ ,\ n^2\ ,\ n^3\ ,\ \ldots\ ,\ 2^n\ ,\ n!$$

Growth of functions

$$n! = n \cdot (n-1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1 \leq n \cdot n \cdot \ldots \cdot n \cdot n \cdot n = n^n$$

$$n! = O(n^n)$$

So

$$\log n! \leq \log n^n = n \log n,$$

$$\log n! = O(n \log n)$$

Growth of functions - a theorem

If $f_1(x) = O(g_1(x))$ and $f_2(x)=O(g_2(x))$, then
$f_1(x) + f_2(x) = O(\max\{g_1(x),g_2(x)\})$

Proof: Let $h(x) = \max\{g_1(x),g_2(x)\})$...

Need to find constants c and k so that
$\forall x>k$, $f_1(x) + f_2(x) \le c \cdot h(x)$.

Know $f_1(x) \le c_1 \cdot g_1(x)$
and $f_2(x) \le c_2 \cdot g_2(x)$.

So, $f_1(x) + f_2(x) \le c_1 \cdot g_1(x) + c_2 \cdot g_2(x)$
$\le c_1 \cdot h(x) + c_2 \cdot h(x)$
$= (c_1+c_2) \cdot h(x)$

"C"

$c = c_1+c_2$,
$k = \max\{k_1,k_2\}$

Growth of functions - a corollary

We know: If $f_1(x) = O(g_1(x))$ and $f_2(x)=O(g_2(x))$, then $f_1(x) + f_2(x) = O(\max\{g_1(x),g_2(x)\})$

Corollary: If $f_1(x) = O(g(x))$ and $f_2(x)=O(g(x))$, then $f_1(x) + f_2(x) = O(g(x))$.

This is just the case where $g_1(x) = g_2(x) = g(x)$

Growth of functions

If $f_1(x) = O(g_1(x))$ and $f_2(x)=O(g_2(x))$, then
$f_1(x)\cdot f_2(x) = O(g_1(x)\cdot g_2(x))$

Proof: we know $f_1(x) \le c_1 g_1(x)$ for $x > k_1$

$f_2(x) \le c_2 g_2(x)$ for $x > k_2$

$f_1(x)\, f_2(x) \le c_1 c_2\, g_1(x)\, g_2(x)$

for

$x > \max(k_1, k_2)$

Growth of functions

Prove that x^3 is not $O(7x^2)$

Proof: By contradiction

Assume $x^3 = O(7x^2)$

So $x^3 \leq c7x^2$ for $x > K$

$x \leq c7$ for $x > K$

But $x \not\leq c7$ for all $x > K$ because eventually $x > c7$ so $x^3 \neq O(7x^2)$

Growth of functions -lower bounds

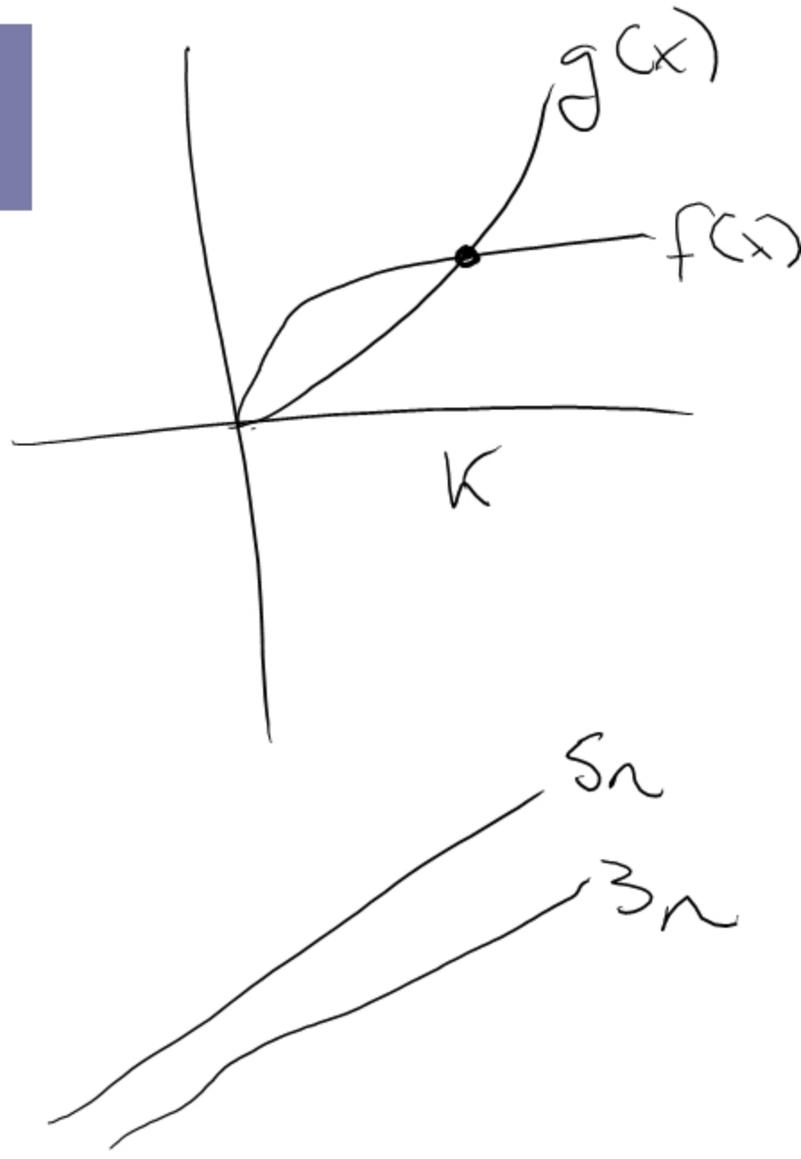
If $f(x) = O(g(x))$ then we write $g(x) = \Omega(f(x))$.

"g is big-omega of f"

What does this mean?

If $\exists$ c,k so that $\forall$ x>k, $f(x) \le c \cdot g(x)$, then by a little algebra: $\exists$ k,c' so that $\forall$ x>k, $g(x) \ge c' f(x)$

$c' = 1/c$



If $f(x) = O(g(x))$, and $f(x) = \Omega(g(x))$, then $f(x) = \Theta(g(x))$

"f is big-theta of g"

When we write f=O(g), it is like $f \le g$
When we write f= Ω(g), it is like $f \ge g$
When we write f= Θ(g), it is like f = g.

Growth of functions - other estimates

For functions f and g, f = o(g) if

$\forall c>0\ \exists k$ so that $\forall\ n>k$, $f(n) \leq c \cdot g(n)$,

"f is little-o of g"

What does this mean?

No matter how tiny c is, cg eventually dominates f.

Example: Show that $n^2 = o(n^2 \log n)$

Proof foreshadowing: find a k (possibly in terms of c) that makes the inequality hold.

Growth of functions - other estimates

For functions f and g, f = o(g) if

$\forall c>0\ \exists k$ so that $\forall\ n>k$, $f(n) \le c \cdot g(n)$,

"f is little-o of g"

Example: Show that $n^2 = o(n^2 \log n)$

Proof foreshadowing: find a k (possibly in terms of c) that makes the inequality hold.

Choose c arbitrarily. How large does n have to be so that $n^2 \le c\, n^2 \log n$?

$1 \le c \log n$

$1/c \le \log n$

$2^{1/c} \le n$

This inequality holds when $n > 2^{1/c}$.

So, $k = 2^{1/c}$.

Growth of functions - other estimates

For functions f and g, f = o(g) if

$\forall c>0\ \exists k$ so that $\forall\ n>k$, $f(n) \leq c \cdot g(n)$,

"f is little-o of g"

Example: Show that $10n^2 = o(n^3)$

Proof foreshadowing: find a k (possibly in terms of c) that makes the inequality hold.

Choose c arbitrarily. How large does n have to be so that $10n^2 \leq c\ n^3$?

$10/c \leq n$

This inequality holds when $n > 10/c$.

So, $k = 10/c$.

Growth of functions - other estimates

For functions f and g, if $f = o(g)$ then $g = \omega(f)$

"g is little-omega of f"

A thought to ponder:

What if $f = o(g)$ and $f = \omega(g)$?