

CSCI 256 Lecture 2:

Stable Matching

Announcements

- Hand in Exercise #1, Pick up Exercise #2 and Handout
- Reminder: Assignment 0 due Wed at 10 pm
- **Office hours (TCL 304):**
 - Mon & Wed 1.30 - 3 pm
 - Tues 10am - noon
- TA hours: still getting sorted out, will post by tomorrow

An Illustrative Example:

The Stable Matching Problem

Applications

- Assigning first year students to advisors
- Pairing job candidates with employers
- Matching doctors to hospitals

Fundamental Problem

- Given preferences of both sides, find a matching that is resilient against opportunistic swapping

State the Problem

- Two groups: hospitals and medical students
- Students have preferences over hospitals
- Hospitals have preferences over students

Goal. Match hospitals to students that is “stable”, that is, no pair has an incentive to break their match!

The Redesign of the Matching Market for American Physicians: Some Engineering Aspects of Economic Design

By ALVIN E. ROTH AND ELLIOTT PERANSON*

We report on the design of the new clearinghouse adopted by the National Resident Matching Program, which annually fills approximately 20,000 jobs for new physicians. Because the market has complementarities between applicants and between positions, the theory of simple matching markets does not apply directly. However, computational experiments show the theory provides good approximations. Furthermore, the set of stable matchings, and the opportunities for strategic manipulation, are surprisingly small. A new kind of “core convergence” result explains this; that each applicant interviews only a small fraction of available positions is important. We also describe engineering aspects of the design process. (JEL C78, B41, J44)

THE MATCH
NATIONAL RESIDENT MATCHING PROGRAM®

National Resident
Matching Program

Matching Med-Students to Hospitals

Input. A set H of n hospitals, a set S of n students and their preferences (each hospital ranks each student, each students ranks each hospital)

	1st	2nd	3rd
MA	Aamir	Beth	Chris
NH	Beth	Aamir	Chris
OH	Aamir	Beth	Chris

	1st	2nd	3rd
Aamir	NH	MA	OH
Beth	MA	NH	OH
Chris	MA	NH	OH

Perfect Matchings

Definition. A matching M is a set of ordered pairs (h, s) where $h \in H$ and $s \in S$ such that

- Each hospital h is in at most one pair in M
- Each student s is in at most one pair in M

A matching M is **perfect** if each hospital is matched to exactly one student and vice versa (i.e., $|M| = |H| = |S|$)

	1st	2nd	3rd
MA	Aamir	Beth	Chris
NH	Beth	Aamir	Chris
OH	Aamir	Beth	Chris

	1st	2nd	3rd
Aamir	NH	MA	OH
Beth	MA	NH	OH
Chris	MA	NH	OH

Unstable Pairs

Definition. A perfect matching M is stable if there are **no blocking pairs**.

A blocking pair is a pair $(h, s) \in H \times S$, such that,

- h prefers s to its current match in M
- s prefers h to its current match in M

Question. Can you point out a blocking pair in this matching?

	1st	2nd	3rd
MA	Aamir	Beth	Chris
NH	Beth	Aamir	Chris
OH	Aamir	Beth	Chris

	1st	2nd	3rd
Aamir	NH	MA	OH
Beth	MA	NH	OH
Chris	MA	NH	OH

Unstable Pairs

Definition. A perfect matching M is stable if there are **no blocking pairs**.

A blocking pair is pair of hospital and student $(h, s) \in H \times S$, such that,

- h prefers s to its current match in M
- s prefers h to its current match in M

Question. Can you point out a blocking pair in this matching?

	1st	2nd	3rd
MA	Aamir	Beth	Chris
NH	Beth	Aamir	Chris
OH	Aamir	Beth	Chris

	1st	2nd	3rd
Aamir	NH	MA	OH
Beth	MA	NH	OH
Chris	MA	NH	OH

Stable Matching Problem

Problem. Given the preference lists of n hospitals and n students, find a stable matching, that is a matching with no blocking pairs.

Question. Does such a matching always exist?

This does not seem obvious!

	1st	2nd	3rd
MA	Aamir	Beth	Chris
NH	Beth	Aamir	Chris
OH	Aamir	Beth	Chris

	1st	2nd	3rd
Aamir	NH	MA	OH
Beth	MA	NH	OH
Chris	MA	NH	OH

How Can We Show This?

- Want to prove: a stable matching always exists
- One way:
 - Give an algorithm to find a stable matching
 - Prove that it is always successful
 - Constructive method



False Starts

Proceed greedily in rounds until matched. In each round,

- Each hospital makes offer to its top available candidate
- Each student accepts its top offer (irrecoverable contract) and rejects others

What goes wrong?

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

False Starts

Proceed greedily in rounds until matched.

- (Round 1) MA $\rightarrow$ Aamir, NH $\rightarrow$ Aamir, OH $\rightarrow$ Chris

-

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

False Starts

Proceed greedily in rounds until matched.

- (Round 1) MA $\rightarrow$ Aamir, NH $\rightarrow$ Aamir, OH $\rightarrow$ Chris
- (Round 1) Aamir rejects MA, accepts NH, Chris accepts OH

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

False Starts

Proceed greedily in rounds until matched.

- (Round 1) MA $\rightarrow$ Aamir, NH $\rightarrow$ Aamir, OH $\rightarrow$ Chris
- (Round 1) Aamir rejects MA, accepts NH, Chris accepts OH
- (Round 2) Only Beth and MA left, and must match

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

False Starts

Proceed greedily in rounds until matched.

- (Round 1) MA $\rightarrow$ Aamir, NH $\rightarrow$ Aamir, OH $\rightarrow$ Chris
- (Round 1) Aamir rejects MA, accepts NH, Chris accepts OH
- (Round 2) Only Beth and MA left, and must match

Is this a stable matching?

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

False Starts

Proceed greedily in rounds until matched.

- (Round 1) MA $\rightarrow$ Aamir, NH $\rightarrow$ Aamir, OH $\rightarrow$ Chris
- (Round 1) Aamir rejects MA, accepts NH, Chris accepts OH
- (Round 2) Only Beth and MA left, and must match

Is this a stable matching?

- Unstable pair: (MA, Chris). What could have avoided it?

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

Gale Shapely Deferred Acceptance

Proceed in rounds until all hospitals matched. In each round,

- Each free hospital offers to its top choice among candidates it hasn't offered yet
- Each free student retains but defers accepting top offer, rejects others
- If a student receives a better offer than currently retained, they reject current and retain new offer (trade up)

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

Gale Shapely Deferred Acceptance

Proceed in rounds until all hospitals matched. In each round,

- Each free hospital offers to its top choice among candidates it hasn't offered yet
- Each free student retains but defers accepting top offer, rejects others
- If a student receives a better offer than currently retained, they reject current and retain new offer (trade up)

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

Gale Shapely Deferred Acceptance

Proceed in rounds until all hospitals matched. In each round,

- Each free hospital offers to its top choice among candidates it hasn't offered yet
- Each free student retains but defers accepting top offer, rejects others
- If a student receives a better offer than currently retained, they reject current and retain new offer (trade up)

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

Gale-Shapely Deferred Acceptance Algorithm

Proceed in rounds until all hospitals matched. In each round,

- Each free hospital offers to its top choice among candidates it hasn't offered yet
- Each free student *retains but defers accepting* **top offer**, rejects others
- If a student receives a better offer than currently retained, they reject current and retain new offer (trade up)

	1st	2nd	3rd
MA	Aamir	Chris	Beth
NH	Aamir	Beth	Chris
OH	Chris	Beth	Aamir

	1st	2nd	3rd
Aamir	OH	NH	MA
Beth	MA	OH	NH
Chris	MA	NH	OH

Gale-Shapely Algorithm

GALE-SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h 's list to whom h has not yet proposed.

IF (s is unmatched)

 Add $h-s$ to matching M .

ELSE IF (s prefers h to current partner h')

 Replace $h'-s$ with $h-s$ in matching M .

ELSE

s rejects h .

RETURN stable matching M .

Analyzing Gale-Shapely

Correctness:

- Does it match everyone? (produce a perfect matching)
- Does it produce a stable matching?

Efficiency:

- How long does it take to produce a matching?
- We will review Big Oh before we analyze this

Useful Invariants

- Iterative algorithms make "progress" which each iteration
- To analyze them, useful to think about its **invariants**
- Invariants are properties that are true before and after iterations
- Suppose a hospital is about to propose to i th student on its list
 - What can we say about students $1, \dots, i - 1$ on its list?
- Suppose a student has received $k > 0$ proposals so far
 - What can we say about their match status?

Analyzing the Algorithm: Correctness

Does it match everyone? (Perfect matching)

- Once a student gets an offer, they are guaranteed to a match
- Equivalently, if a student is unmatched, then no hospital has offered them
 - Some hospital has not exhausted its preference lists
- When the algorithm terminates, everyone is matched (i.e., it produces a perfect matching).

Does it produce a stable matching?

- Key idea: students always 'trade up'
- s breaks match with h in favor of h' only if s prefers h' to h

Analyzing the Algorithm: Correctness

Lemma 1. Gale Shapley algorithm always produces a stable matching.

Proof. (By contradiction) Let M be the resulting matching. Suppose $\exists(h, s)$ such that $(h, s'), (h', s) \in M$ and

- h prefers s over s' and s prefers h over h'

Thus h must have offered to s before s'

- Either s broke the match to h at some point for some h'' , or s already had a match h'' that s preferred over h

But students always trade up, so s must prefer final match h' over h'' , which they prefer over h . ($\Rightarrow \Leftarrow$) ■

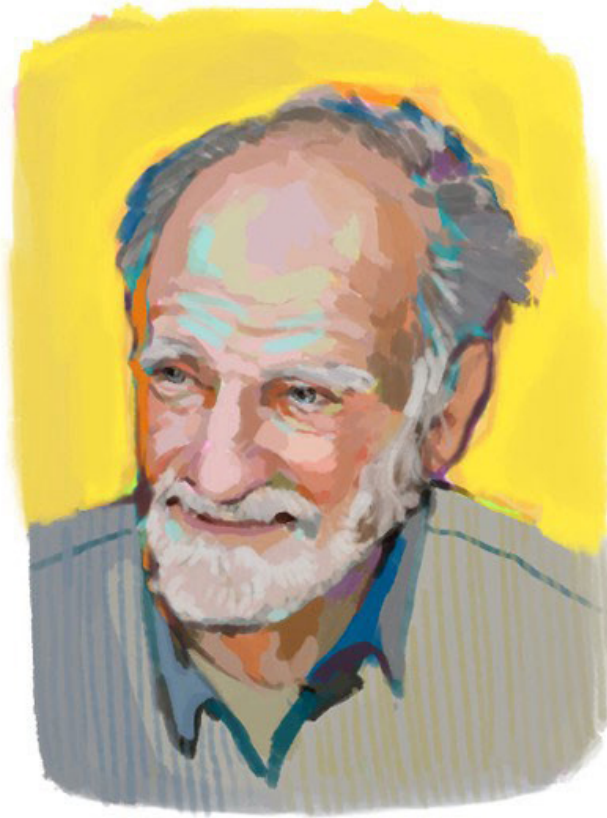
Does Who Proposes Matters?

- Can get different matchings depending on which side proposes
- Stability holds irrespective of who proposes
 - An input can have multiple stable matchings
- **Question.** Does Gale-Shapely produce the “best matching” for hospitals or students?
- Turns out hospital-proposing algorithm produces a unique matching that is **hospital optimal** and **student pessimal**
 - Matches hospital to “best achievable” student and student to “worst-achievable” hospital among all stable matchings

Nobel Prize 2012: Shapley & Roth



David Gale
PROFESSOR, UC BERKELEY



Lloyd Shapley
PROFESSOR EMERITUS, UCLA

Stable matching: Theory, evidence, and practical design

*This year's Prize to **Lloyd Shapley** and **Alvin Roth** extends from abstract theory developed in the 1960s, over empirical work in the 1980s, to ongoing efforts to find practical solutions to real-world problems. Examples include the assignment of new doctors to hospitals, students to schools, and human organs for transplant to recipients. Lloyd Shapley made the early theoretical contributions, which were unexpectedly adopted two decades later when Alvin Roth investigated the market for U.S. doctors. His findings generated further analytical developments, as well as practical design of market institutions.*

Analyzing Gale-Shapely

Correctness:

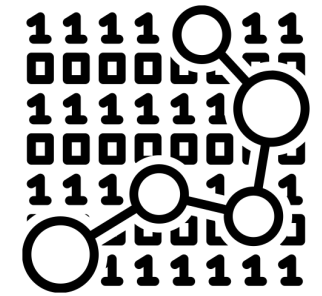
- Does it match everyone? (produce a perfect matching)
- Does it produce a stable matching?

Efficiency:

- How long does it take to produce a matching?
- We will review Big Oh before we analyze this

Algorithm Analysis Model

- Analyze performance: number of steps
- Measured as a function of **input size** (typically n)
 - Care about scalability



$$|x| = n$$

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

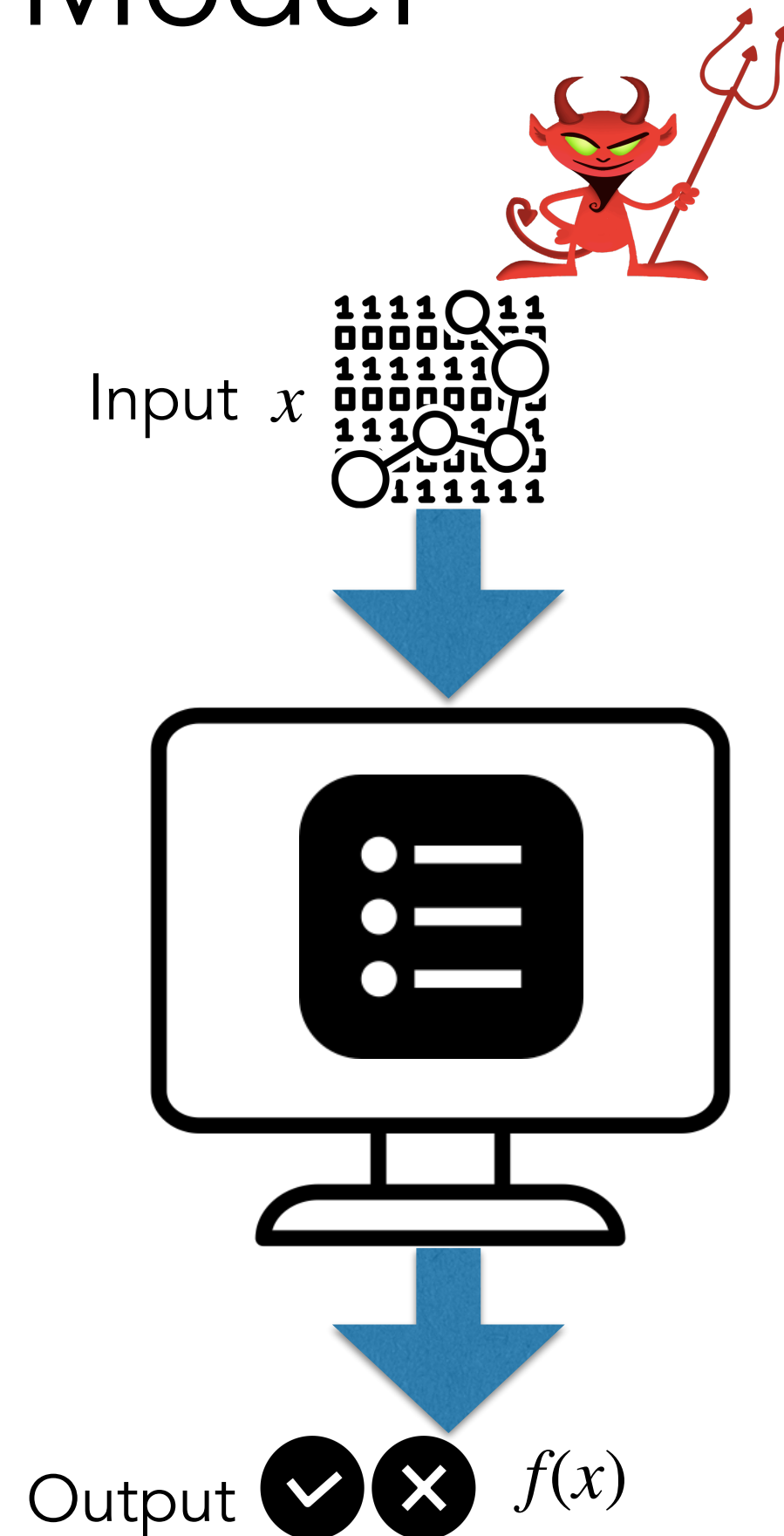
	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1,000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10,000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100,000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1,000,000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Polynomials are Scalable

- **Scalability.** When the input size doubles the algorithm should slow down by at most some constant factor C
- Examples
 - $f(n) = n^k$, then $f(2n) = 2^k n^k = cn^k$ for any fixed k
 - $g(n) = \log n$, then $g(2n) = \log 2 + \log n \approx \log n$
- But not for these functions
 - $f(n) = 2^n$, then $f(2n) = 2^{2n} = (2^n)^2$
 - $g(n) = n!$, then $g(2n) = 2n! \geq n^n \cdot n!$

Algorithm Analysis Model

- For any algorithm, some inputs are easy, others are hard
- **Worst-case paradigm:** Performance measured as the **maximum number of steps** taken on any input of size n
 - Adversarial analysis



Why Give Worst Case Guarantees?

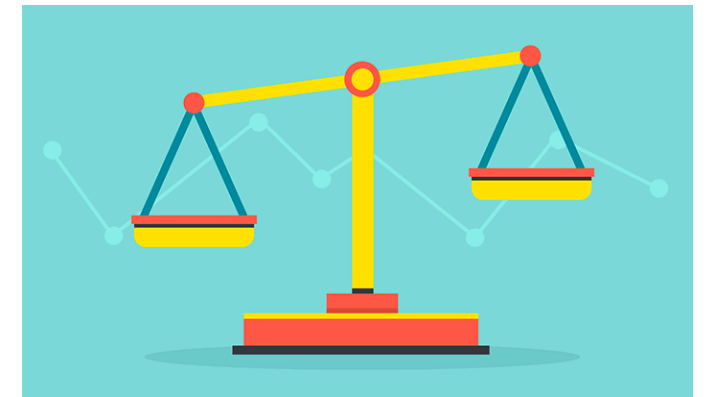
- Worst-case analysis: dominant algorithm design paradigm



Powerful guarantee



Universal



Mathematically
compare algorithms

- **Downside:** Can be too pessimistic on typical (non-worst-case) instances

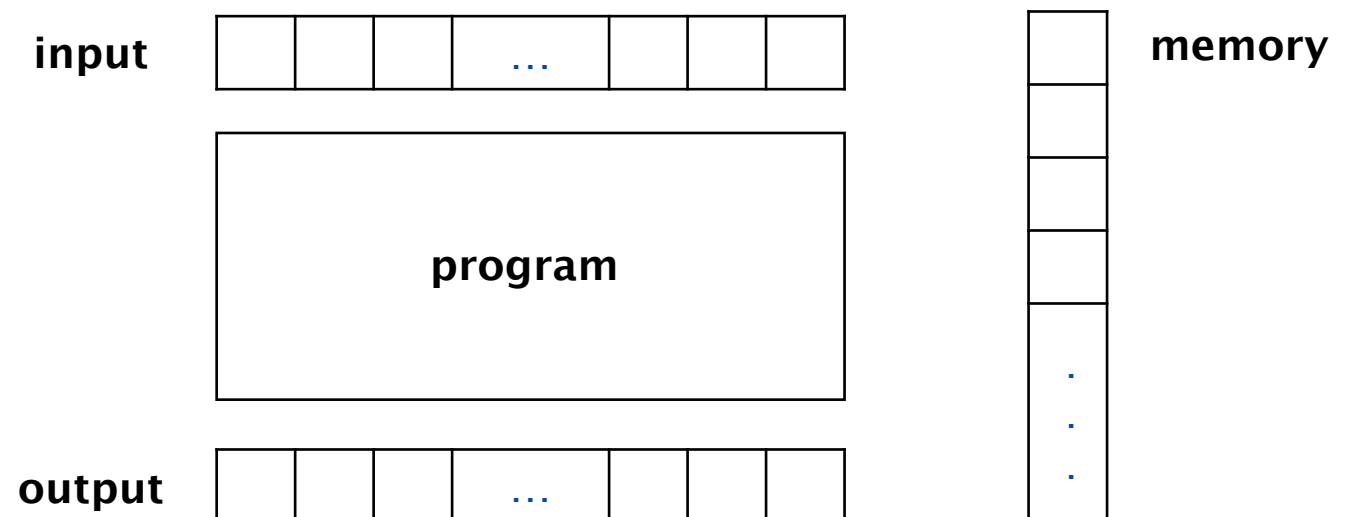
Beyond-Worst-Case Analyses

- **Probabilistic.** Expected running time of a randomized algorithm
 - e.g., the expected running time of quicksort
- **Amortized.** Worst-case running time for any sequence of n operations
 - Some operations can be expensive but may make future operations fast (doing well on average)
- **New. Algorithms with predictions.**
 - Obtain a prediction about the input instance or solution
 - Running time proportional to error in predicted instances



How to Measure Cost

- “Word RAM” model of computation
- Every primitive operation costs 1 unit of time:
 - Adding/multiplying/dividing/etc two ints or floats costs 1
 - An if statement costs 1
 - A comparison costs 1
 - Array access costs 1
- Etc.



Model of Computation Details

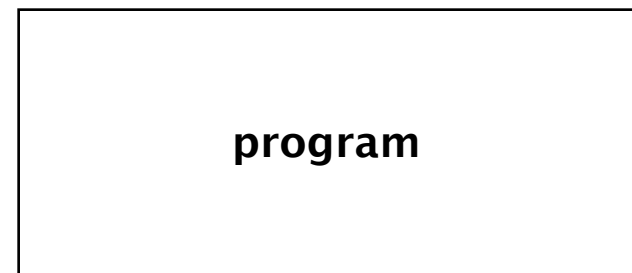
- Word RAM model
 - Each memory location and input/output cell stores a "**word**" which is a w -bit integer (assume $w \geq \log_2 n$)
 - **Primitive operations:** arithmetic operations, read/write memory, array indexing, following a pointer etc. are constant time
- **Running time:** number of primitive operations
- **Space:** number of memory cells utilized

Space is measured in "words" (ints, floats, chars, etc) not bits

input



program



output

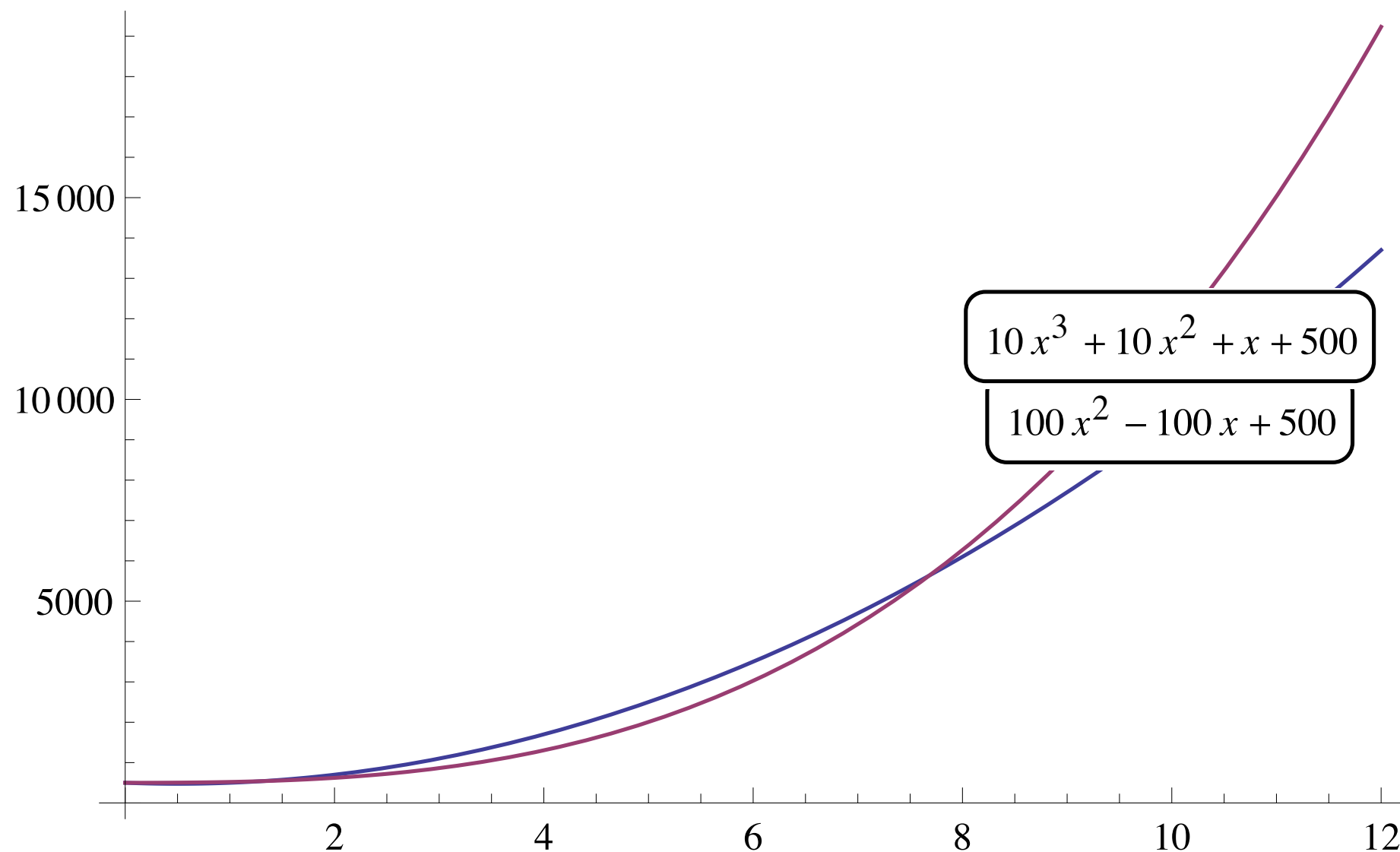


memory



Asymptotic Growth

Running time as a function of n as “as n gets large”



Asymptotic Upper Bounds

Definition. $f(n)$ is $O(g(n))$ if there exists constants $c > 0$ and $n_0 \geq 0$ such that

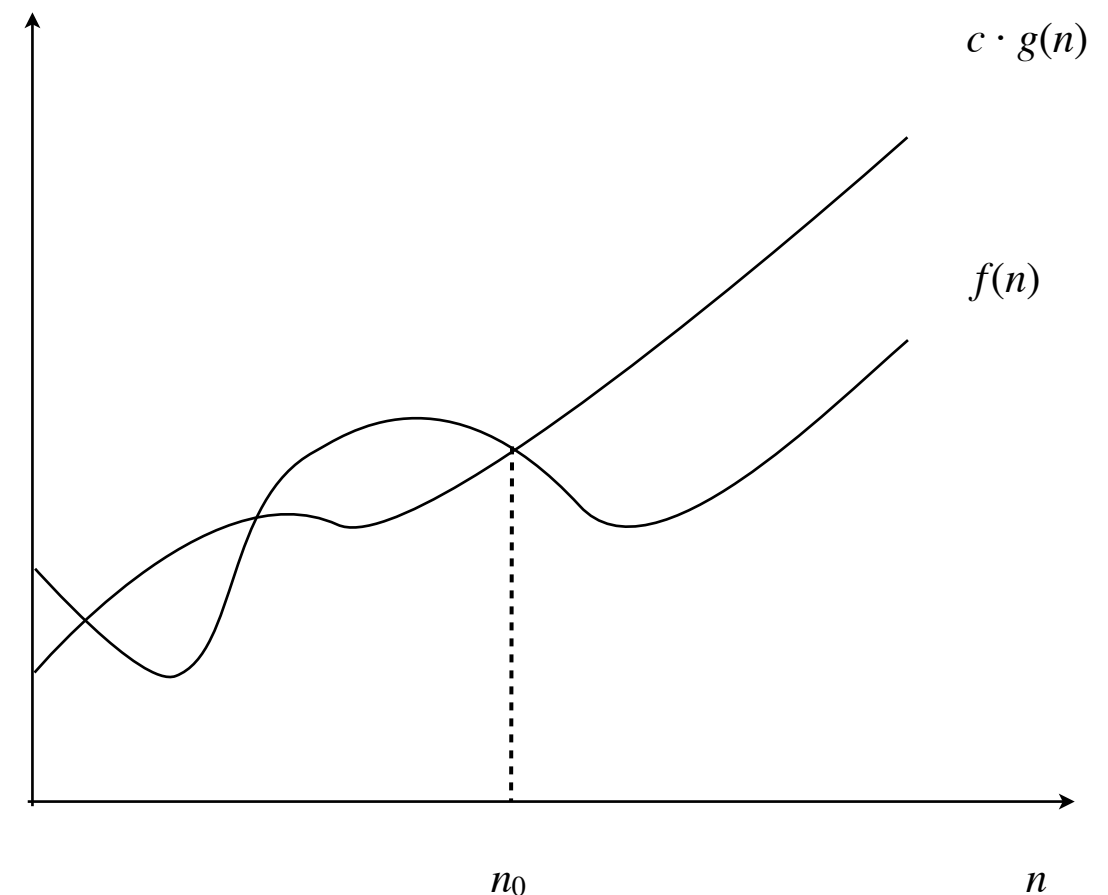
$$0 \leq f(n) \leq c \cdot g(n) \quad \text{for all } n \geq n_0$$

That is, for sufficiently large n , $f(n)$ is asymptotically bounded above by $g(n)$

Examples

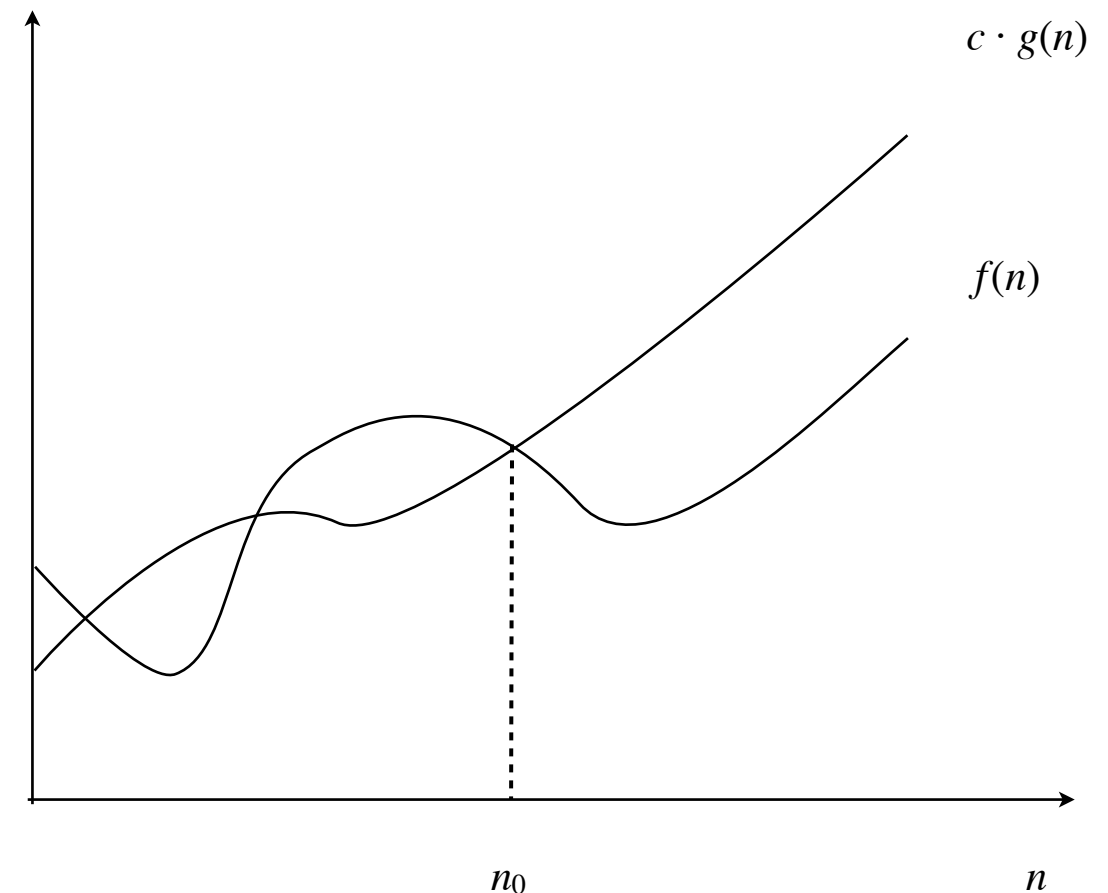
- $100n^2 = O(n^2)$
- $n \log n = O(n^2)$
- $5n^3 + 2n + 1 = O(n^3)$

Typical usage. Insertion sort makes $O(n^2)$ compares to sort n elements



Big Oh- Notational Abuses

- $O(g(n))$ is actually a set of functions, but the CS community writes $f(n) = O(g(n))$ instead of $f(n) \in O(g(n))$
- For example
 - $f_1(n) = O(n \log n) = O(n^2)$
 - $f_2(n) = O(3n^2 + n) = O(n^2)$
 - But $f_1(n) \neq f_2(n)$
- Okay to abuse notation in this way



Class Quiz

Let $f(n) = 3n^2 + 17n \log_2 n + 1000$. Which of the following are true?

- A. $f(n)$ is $O(n^2)$.
- B. $f(n)$ is $O(n^3)$.
- C. Both A and B.
- D. Neither A nor B.

Something Missing

- Big Oh is like $\leq$
- So one can accurately say “merge sort requires $O(2^n)$ time,” but it’s not very meaningful
- Can we get terminology like big- Ω that lower bounds?
- Or that shows two functions are “equal” (up to constants and for large values of n)?

Asymptotic Lower Bounds

Definition. $f(n)$ is $\Omega(g(n))$ if there exists constants $c > 0$ and $n_0 \geq 0$

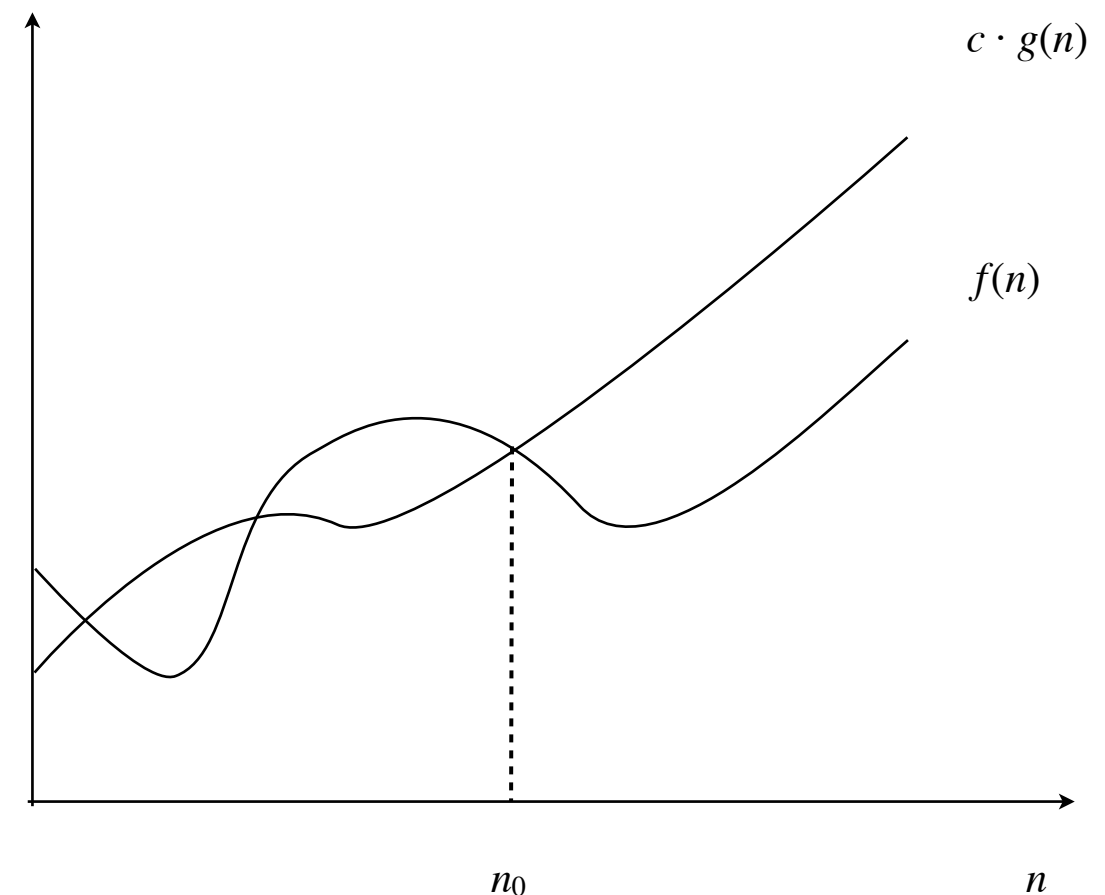
$$f(n) \geq c \cdot g(n) \geq 0 \quad \text{for all } n \geq n_0$$

That is, for sufficiently large n , $f(n)$ is asymptotically bounded below by $g(n)$

Examples

- $100n^2 = \Omega(n^2) = \Omega(n)$
- $n \log n = \Omega(n)$

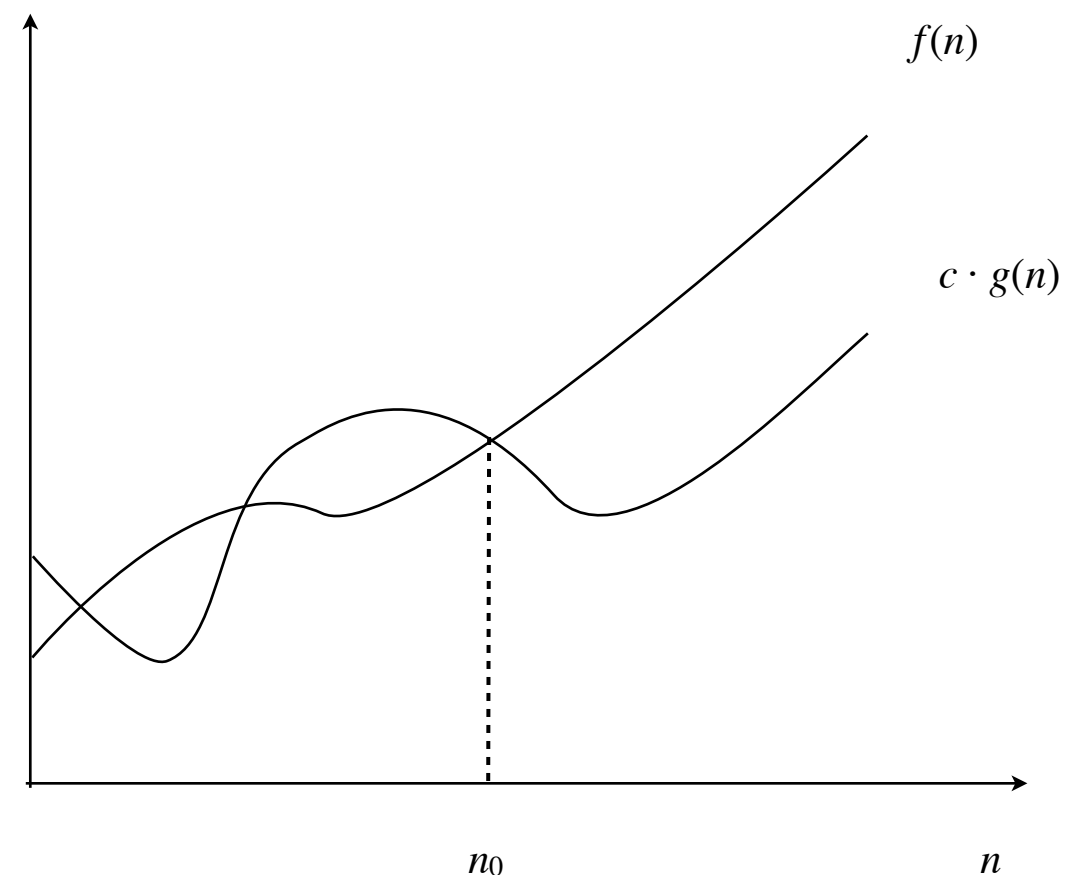
Typical usage. Insertion sort makes $O(n^2)$ compares to sort n elements



Why Lower Bounds?

Show that an algorithm performs at least so many steps

- Searching an unordered list of n items: $\Omega(n)$ steps in some cases
- Insertion sort take $\Omega(n^2)$ steps in some cases
- Mergesort takes $\Omega(n \log n)$ steps in all cases



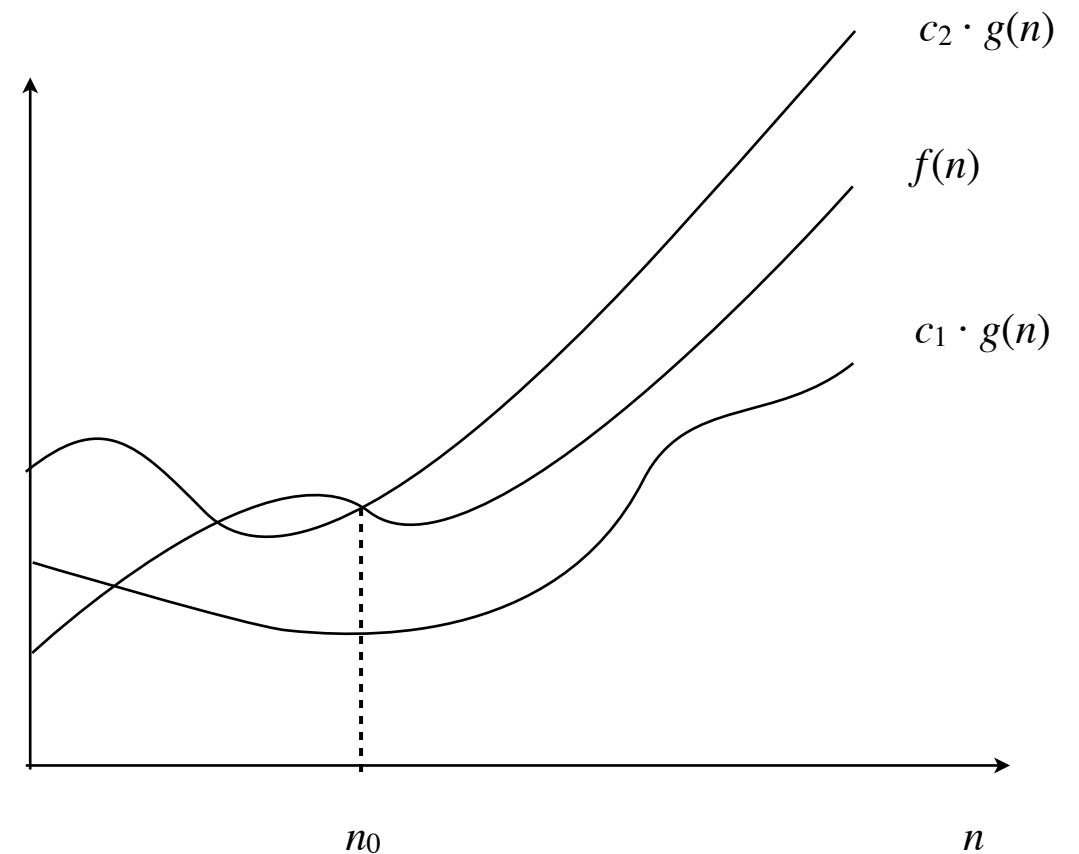
Asymptotically Tight Bounds

Definition. $f(n) = \Theta(g(n))$ if

$$f(n) = O(g(n)) \quad \text{and} \quad f(n) = \Omega(g(n))$$

Examples

- $5n^3 + 2n + 1 = \Theta(n^3)$
- $\log_{100} n = \Theta(\log_2 n)$



Tools for Comparing Asymptotics

- Logs grow slowly than any polynomial:
 - $\log_a n = O(n^b)$ for every $a > 1$, $b > 0$
- Exponentials grow faster than any polynomial:
 - $n^d = O(r^n)$ for every $d > 1$, $r > 0$
- Using limits:
 - If $\lim_{n \rightarrow \infty} \frac{f(x)}{g(x)} = 0$, then $f(x) = O(g(x))$
 - If $\lim_{n \rightarrow \infty} \frac{f(x)}{g(x)} = c$ for some constant $0 < c < \infty$, then $f(x) = \Theta(g(x))$

Tools for Comparing Asymptotics

- Sometimes running time functions may need to be simplified
 - For example, compare $3^{\log n}$ vs 2^n
- Simplifying using properties of logs/exponents can help

Playing with Logs: Properties

- In this class, $\log n$ means $\log_2 n$, $\ln n = \log_e n$
- Constant base doesn't matter: $\log_b(n) = \frac{\log n}{\log b} = O(\log n)$
- Properties of logs:
 - $\log(n^m) = m \log n$
 - $\log(ab) = \log a + \log b$
 - $\log(a/b) = \log a - \log b$

$$a^{\log_a n} = n$$

We will use this a lot!

Exponents

$$n^a \cdot n^b = n^{a+b}$$

$$(n^a)^b = n^{ab}$$

Tools for Comparing Asymptotics

- Sometimes running time functions may need to be simplified
 - For example, compare $3^{\log n}$ vs 2^n
- Simplifying using properties of logs/exponents can help
- Write $3^{\log n} = (2^{\log_2 3})^{\log_2 n} = (2^{\log_2 n})^{\log_2 3} = n^{\log_2 3} = n^c$
 - Now easy to compare n^c to 2^n

Class Exercise

- Answer True/False by simplifying the function

- Is $n^{1/\log n} = O(1)$?

- Is $\log \sqrt{4^n} = O(n^2)$?

- Is $n = O(2^{\log_4 n})$?

Class Exercise

- When comparing two functions, helpful to simplify first
- Is $n^{1/\log n} = O(1)$?
 - Simplify $n^{1/\log n} = (2^{\log n})^{1/\log n} = 2 : \mathbf{True}$
- Is $\log \sqrt{4^n} = O(n^2)$?
 - Simplify $\log \sqrt{2^{2n}} = \log 2^n = n \log 2 = O(n) : \mathbf{True}$
- Is $n = O(2^{\log_4 n})$?
 - Simplify $2^{\log_4 n} = 2^{\frac{\log_2 n}{\log_2 4}} = 2^{(\log_2 n)/2} = 2^{\log_2 \sqrt{n}} = \sqrt{n} : \mathbf{False}$

Next Time: Analyzing
Gale Shapley Running Time