(record)

CS3000: Algorithms & Data
Drew van der Poel

Lecture 2

- Finish Lecture 1 (Induction)
- Stable Matching: the Gale-Shapley Algorithm

May 11, 2021

Our First Proof

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T([n/2])$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$

$$T(2^m) = 3m + 2$$

Proof by Induction

- We will prove our claim using **induction**

- **Induction:**
  - Used to prove a claim *H* is true for every natural number *i* starting at a first value (usually *0* or *1*) – *H(i)* is true ∀ *i*
  - How:
    - 1. Base case – prove directly for *H(1)* (or whatever the base case(s) is/are)
    - 2. Inductive step – For general *k*, show that if *H(k-1)* is true, then *H(k)* is true. The assumption that *H(k-1)* is true is the **inductive hypothesis (IH).**

$$\underbrace{H(1)}_{\text{Base Case}} \xrightarrow{\text{Ind. step}} H(2) \xrightarrow{\text{Ind. step}} H(3 \rightarrow \ldots \rightarrow H(100)$$

  - Why:
    - Suppose we want to prove *H(100)*. First, we can use the base case to show *H(1)* holds. Then, because *H(1)* is true, *H(2)* is true via inductive step, and then *H(3)* is true, and so on, all the way to *H(100)* (or whatever value!).

- Problems: counting students

- Alg. techniques:

- Analysis:

- Proof techniques: **induction**

Proof by Induction

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$

- **Induction:** "automatically" prove for every $\boldsymbol{m}$
  - Let $\underline{H(m)}$ be the statement $\underline{T(2^m)} = \underline{3m + 2}$
  - **Base Case:** Show $\underline{H(0)}$ is true
  - **Inductive Step:** For every $m \geq 1$, can assume $\underline{H(m-1)}$ is true to show $\underline{H(m)}$ is true
  - **Conclusion:** statement is true for every m ✓

Proof by Induction

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$
- Let $H(m)$ be the statement $T(2^m) = 3m + 2$
- **pf. Base:** need to show *H(0)* is true,

  $H(0)$: $T(2^0) = 3 \cdot 0 + 2 = 2$ ✓

  $2 = T(1) =$

  IH $\rightarrow$ $T(2^{m-1}) = 3(m-1) + 2 = 3m - 3 + 2 = 3m - 1$

  Inductive: can assume $H(m-1)$ is true to show $H(m)$ is true, that $T(2^m) = 3m + 2$

  $$T(2^m) = 3 + T\left(\left\lceil \frac{2^m}{2} \right\rceil\right) = 3 + T(2^{m-1})$$

  $$= 3 + 3m - 1 \quad (IH)$$

  $$= 3m + 2$$

  □

Proof by Induction

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$
- Let $H(m)$ be the statement $T(2^m) = 3m + 2$
- **pf. Base:** need to show *H(0)* is true,

$H(0)$: $T(2^0) = 3(0) + 2$

$T(1) = 2$ **;** so *H(0) is true*

Inductive: can assume $H(m-1)$ is true to show $H(m)$ is true, that $T(2^m) = 3m + 2$

$T(2^m)$ *= 3 +* $T(\lceil 2^m/2 \rceil)$ *= 3 +* $T(2^{m-1})$

$= 3 + 3(m-1) + 2$ *= 3m + 2;* *so H(m) is true*

Comparing to Simple Counting

- **# of steps in RecCount:** $T(n) = \underline{3m + 2}$ for every number of students $\underline{n = 2^m}$

$\log_2 n = m$

- $m = \log_2 n$ → $\boxed{T(n) = 3\log_2 n + 2}$

Running Time

- **Simple counting:** $T(n) = 3n$ steps
- **Recursive counting:** $T(n) = 3\log_2 n + 2$ steps

- In January 2020:
  SimpleCount - ~1.228 seconds/student
  RecursiveCount - ~1.516 seconds/student

- Which requires more steps?

Simple counting

Running Time

- **Simple counting:** $T(n) = 3n$ steps
- **Recursive counting:** $T(n) = 3\log_2 n + 2$ steps

- Simple counting had more steps, but was faster???

steps aren't well defined,

don't take exact same amount of time

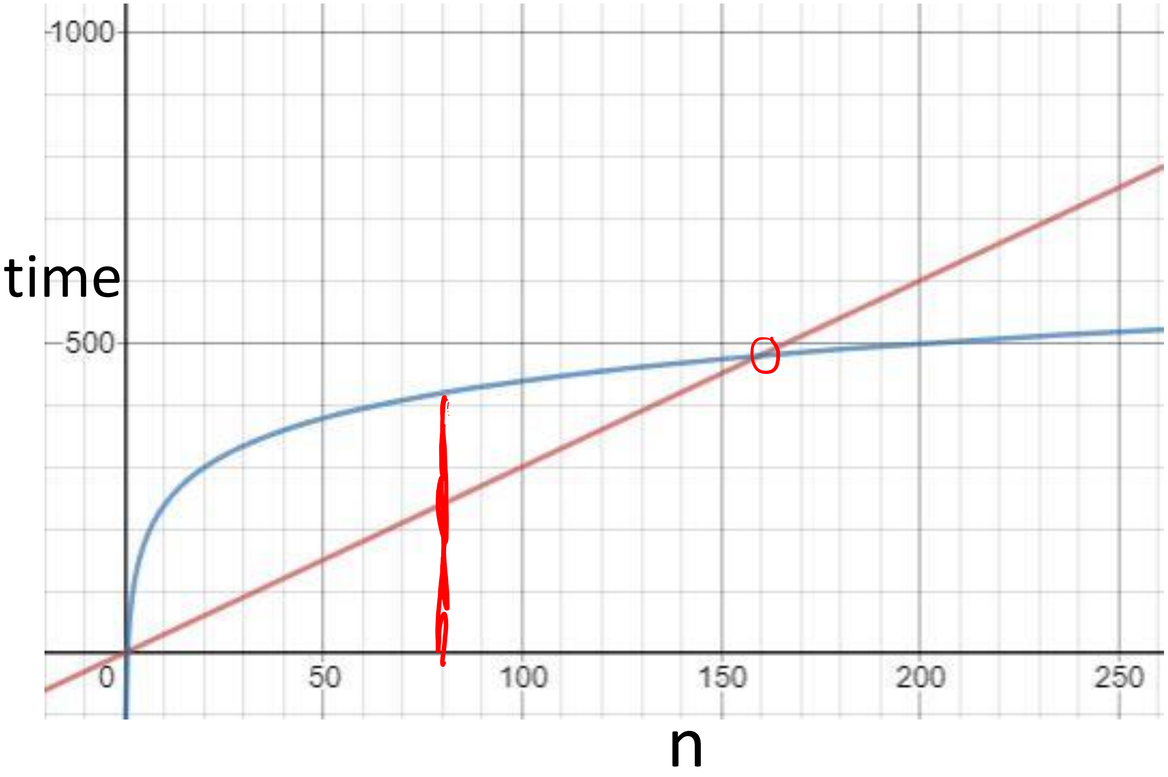
Running Time

- **Simple counting:**

  $3n$ time

- **Recursive counting:**

  $60\log_2 n + 40$ time



- **Compare algorithms by asymptotics!**
  - Log-time beats linear-time as $n \to \infty$

Induction Practice

$2^0 + 2^1 + 2^2 + \ldots + 2^{n-1}$

- **Claim:** For every $n \geq 1$, $\sum_{i=0}^{n-1} 2^i = 2^n - 1$

$H(k): \sum_{i=0}^{k-1} 2^i = 2^k - 1$

Ex. $H(4): \sum_{i=0}^{3} 2^i = 2^4 - 1$

- **Proof by Induction:**

LHS RHS

Base: $H(1): \sum_{i=0}^{0} 2^i = 2^0 = 1 \qquad 2^1 - 1 = 1$ ✓

Ind: $\overset{K}{H(k-1)} \rightarrow \overset{K+1}{H(k)}$

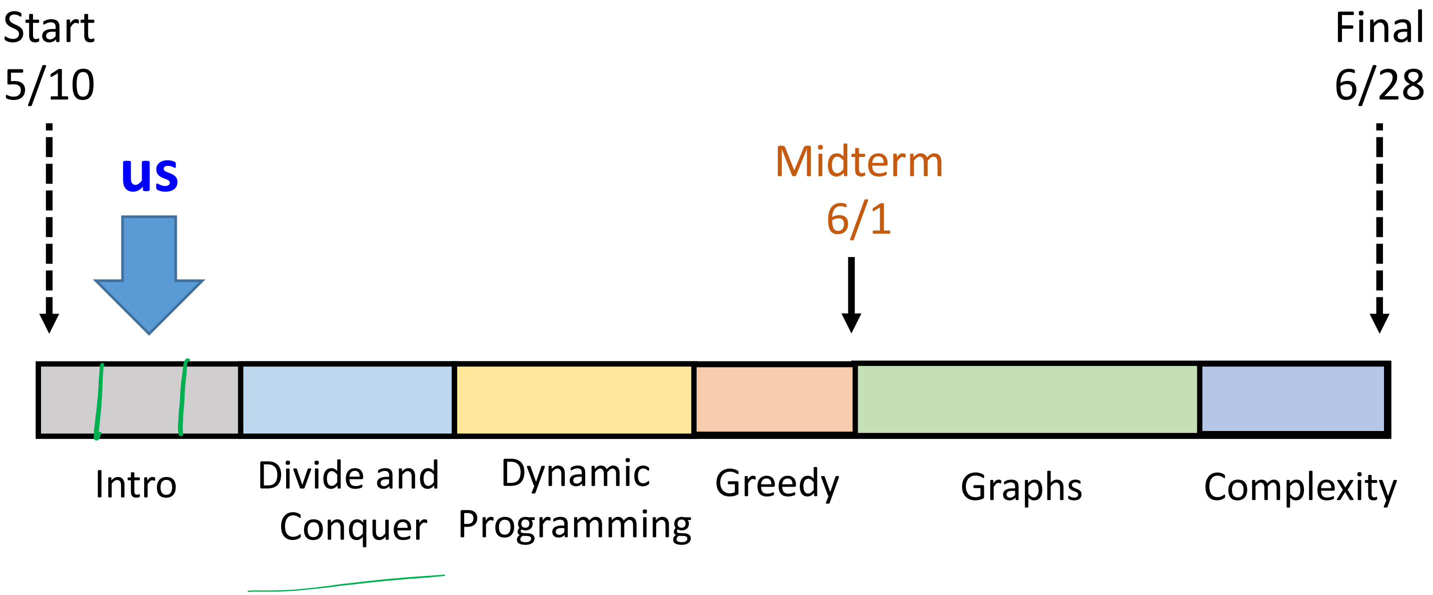
IH: $\sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$

$\sum_{i=1}^{5} i = \underbrace{1+2+3+4}+5 = \sum_{i=1}^{4} i + 5$

$\sum_{i=0}^{k-1} 2^i = \sum_{i=0}^{k-2} 2^i + 2^{k-1} = \underbrace{2^{k-1} - 1}_{IH} + 2^{k-1}$

$= 2 \cdot 2^{k-1} - 1 = 2^k - 1$ ✓

Outline



Last class: student counting, proof by induction

Next class: asymptotic analysis

Labor Markets

- Most labor markets are frustrating
  - Not everyone can get their favorite job/candidate
  - The market is decentralized
  - This leads to potential **chaos**

- Decentralized labor markets are confusing

  - You get an offer from your 2nd choice – what should you do to maximize your happiness?
    - Accept -> Could have been happier if #1 offers
    - Decline -> Could never get #1 offer, would have been happier w/ #2

Centralized Labor Markets

- What if we could just assign jobs?
  - What information would we want?
    - # of employees
    - # of jobs
    - list of preferences (both employees + employers)
  - How could we prevent the earlier chaos?

stable assignment – no (employee, employer) pair prefer each other to their current match

↗ possible

Input

- We are given the following information
  - n doctors $d_1 \dots d_n$
  - n hospitals $h_1 \dots h_n$
  - each doctor's ranking of hospitals (e.g. $d_1 : h_2 \succ h_3 \succ h_1$)
  - each hospital's ranking of doctors (e.g. $h_1 : d_1 \succ d_3 \succ d_2$)

Same

$n = 5$

	1st	2nd	3rd	4th	5th
MGH	Bob	Alice	Dorit	Ernie	Clara
BW	Dorit	Bob	Alice	Clara	Ernie
BID	Bob	Ernie	Clara	Dorit	Alice
MTA	Alice	Dorit	Clara	Bob	Ernie
CH	Bob	Dorit	Alice	Ernie	Clara

	1st	2nd	3rd	4th	5th
Alice	CH	MGH	BW	MTA	BID
Bob	BID	BW	MTA	MGH	CH
Clara	BW	BID	MTA	CH	MGH
Dorit	MGH	CH	MTA	BID	BW
Ernie	MTA	BW	CH	BID	MGH

Matchings

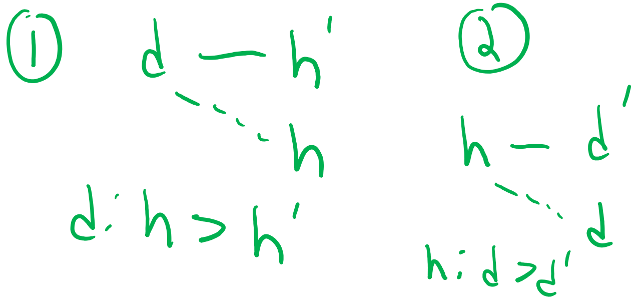
- A matching M is a (non-empty) set of doctor-hospital pairs where no doctor/hospital appears twice
  - $e.g.\ M = \{\,(d_1, h_2), (d_2, h_3)\,\}$

  - perfect matching: every doctor/hospital appears once

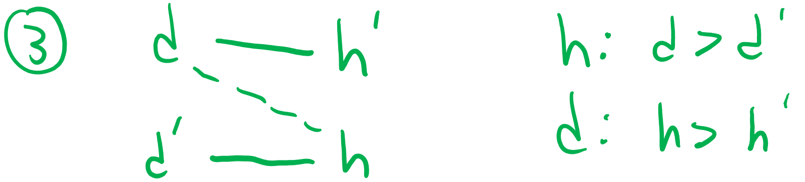
Terminology:

- "d is matched to h": $(d, h) \in M$
- "d is matched": $(d, h) \in M$ for some *h*

Stable Matchings



- A matching M is unstable if some doctor-hospital pair prefer one another to their mate in M



- Instabilities
  1. d, h such that d is matched to h', h is unmatched, but $d : h \succ h'$
  2. d, h such that h is matched to d', d is unmatched, but $h : d \succ d'$
  3. d, h such that d is matched to h', h is matched to d', but $d : h \succ h'$ and $h : d \succ d'$

If a matching M is perfect and not unstable it is **stable**

- Problems: counting students, **stable matching**

- Alg. techniques:

- Analysis:

- Proof techniques: induction

Ask the Audience

- Either find a stable matching or convince yourself that there is no stable matching

	1st	2nd	3rd
MGH	Alice	Bob	Clara
BW	Bob	Clara	Alice
BID	Alice	Clara	Bob

	1st	2nd	3rd
Alice	BW	BID	MGH
Bob	BID	MGH	BW
Clara	MGH	BID	BW

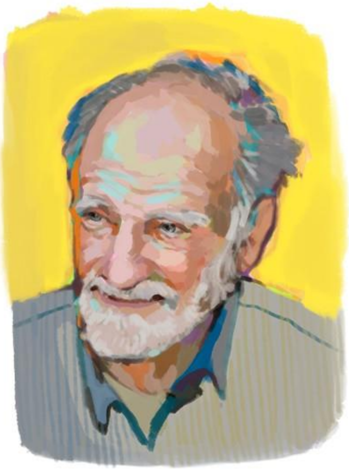
- Solution:

{ (MGH, C), (BW, A), (BID, B) } ✓

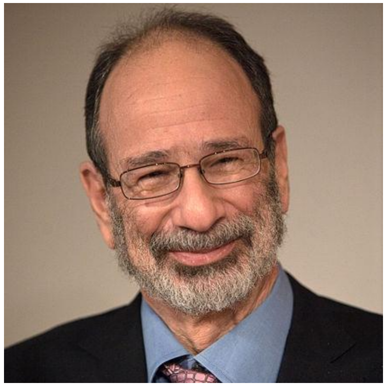
Gale-Shapley Algorithm

- National system for matching US medical school graduates to medical residencies
  - Roughly 40,000 doctors per year
  - Assignment is almost entirely algorithmic

David Gale (1921-2008)
PROFESSOR, UC BERKELEY



Lloyd Shapley
PROFESSOR EMERITUS, UCLA



Alvin Roth
PROFESSOR, STANFORD

Gale-Shapley Algorithm

```
• Let M be empty
• While (some hospital h is unmatched):
    • If (h has offered a job to everyone): break
    • Else: let d be the highest-ranked doctor to
      which h has not yet offered a job
    • h makes an offer to d:
        • If (d is unmatched):
            • d accepts, add (d,h) to M
        • ElseIf (d is matched to h′ & d: h′ > h):
            • d rejects, do nothing
        • ElseIf (d is matched to h′ & d: h > h′):
            • d accepts, remove (d,h′) from M and
              add (d,h) to M
• Output M
```

“job offer”

Gale-Shapley Demo

{ (MGH, A), (BW, C), (BID, B), (MTA, E), (CH, D) }

	1st	2nd	3rd	4th	5th
MGH	~~Bob~~	Alice	Dorit	Ernie	Clara
BW	~~Dorit~~	~~Bob~~	~~Alice~~	Clara	Ernie
BID	Bob	Ernie	Clara	Dorit	Alice
MTA	~~Alice~~	~~Dorit~~	~~Clara~~	~~Bob~~	Ernie
CH	~~Bob~~	Dorit	Alice	Ernie	Clara

	1st	2nd	3rd	4th	5th
Alice	CH	MGH	BW	MTA	BID
Bob	BID	BW	MTA	~~MGH~~	CH
Clara	BW	BID	MTA	CH	MGH
Dorit	MGH	CH	~~MTA~~	BID	~~BW~~
Ernie	MTA	BW	CH	BID	MGH

Observations (for proofs later on)

- Hospitals make offers in descending order
- Doctors that get a job never become unemployed
- Doctors accept offers in ascending order

Gale-Shapley Algorithm

- Questions about the Gale-Shapley Algorithm:
    - Will this algorithm terminate? And how long will it take?
    - Does it output a perfect matching?
    - Does it output a stable matching?
    - How do we implement this algorithm efficiently?

GS Algorithm: Termination

- **Claim:** The GS algorithm terminates after at most n^2 iterations of the main loop (n^2 job offers).

Gale-Shapley Algorithm

- Questions about the Gale-Shapley Algorithm:
  - Will this algorithm terminate? Yes!
  - Does it output a perfect matching?
  - Does it output a stable matching?
  - How do we implement this algorithm efficiently?

GS Algorithm: Perfect Matching

- **Claim:** The GS algorithm returns a perfect matching (all doctors/hospitals are matched)

Proof by Contradiction

- **Important:** No claim/proposition can be both true and false
- Assume the claim *C* that you want to prove true is *false* (not-*C* is true)
- Then show the claim being false implies **contradictory** assertions (that both an assertion *Q* and not-*Q* are true)
- Since *Q* and not-*Q* cannot both be true, *C* must be true

"one of a mathematician's finest weapons" – G. H. Hardy

- Problems: counting students, stable matching

- Alg. techniques:

- Analysis:

- Proof techniques: induction, **contradiction**

GS Algorithm: Perfect Matching

- **Claim:** The GS algorithm returns a perfect matching (all doctors/hospitals are matched)

Gale-Shapley Algorithm

- Questions about the Gale-Shapley Algorithm:
  - Will this algorithm terminate? Yes!
  - Does it output a perfect matching? Yes!
  - Does it output a stable matching?
  - How do we implement this algorithm efficiently?

GS Algorithm: Stable Matching

- **Stability:** GS algorithm outputs a stable matching
- Proof by contradiction:
  - Suppose there is an instability *(d, h'), (d', h)*
  - That is, given a matching which includes *(d, h'), (d', h), d* prefers *h* to *h'* and *h* prefers *d* to *d'*

GS Algorithm: Stable Matching

- **Stability:** GS algorithm outputs a stable matching
- Proof by contradiction:
  - Suppose there is an instability $(d, h'), (d', h)$
    - $h: d > d'$
    - $d: h > h'$
- We know h made an offer to d before d' (by obs. 1)
  - Case 1
  - Case 2

GS Algorithm: Stable Matching

- **Stability:** GS algorithm outputs a stable matching
- Proof by contradiction:
  - Suppose there is an instability *(d, h'), (d', h)*
    - *h: d > d'*
    - d: h > h'

  We know *h* made an offer to *d* before *d'*
  - Case 1 – *d* rejected the offer

GS Algorithm: Stable Matching

- **Stability:** GS algorithm outputs a stable matching
- Proof by contradiction:
  - Suppose there is an instability *(d, h’), (d’, h)*
    - *h: d > d’*
    - d: h > h’

  We know *h* made an offer to *d* before *d’*
  - Case 2 – *d* accepted the offer

Gale-Shapley Algorithm

- Questions about the Gale-Shapley Algorithm:
  - Will this algorithm terminate? Yes!
  - Does it output a perfect matching? Yes!
  - Does it output a stable matching? Yes!
  - How do we implement this algorithm efficiently?

GS Algorithm: Running Time

- **Running Time:**
  - A straightforward implementation requires $\approx n^3$ operations in the worst case, $\approx n^2$ space
  - (≈ -> dropping constants & lower-order terms)

GS Algorithm: Running Time

```
• Let M be empty
• While (some hospital h is unmatched):
    • If (h has offered a job to everyone): break
    • Else: let d be the highest-ranked doctor to
      which h has not yet offered a job
    • h makes an offer to d:
        • If (d is unmatched):
            • d accepts, add (d,h) to M
        • ElseIf (d is matched to h′ & d: h′ > h):
            • d rejects, do nothing
        • ElseIf (d is matched to h′ & d: h > h′):
            • d accepts, remove (d,h′) from M and
              add (d,h) to M
• Output M
```

“job offer”

GS Algorithm: Running Time

```
• Let M be empty
• While (some hospital h is unmatched):
    • If (h has offered a job to everyone): break
    • Else: let d be the highest-ranked doctor to
      which h has not yet offered a job
    • h makes an offer to d:
        • If (d is unmatched):
            • d accepts, add (d,h) to M
        • ElseIf (d is matched to h′ & d: $\underline{h' > h}$):
            • d rejects, do nothing
        • ElseIf (d is matched to h′ & d: $\underline{h > h'}$):
            • d accepts, remove (d,h′) from M and
              add (d,h) to M
• Output M
```

“job offer”

- Loop runs ≤ n^2 times; ≤ *n* operations to find *h, h'* in *d*'s preferences
- n^2 offers * *n* operations = n^3 total operations

GS Algorithm: Running Time

- **Running Time:**
  - A careful implementation requires just $\approx n^2$ operations in the worst case and $\approx n^2$ space

GS Algorithm: Running Time

- **Running Time:**
  - A careful implementation requires just $\approx n^2$ operations in the worst case and $\approx n^2$ space
  - Create an array of doctor x hospital in n^2 steps

	1st	2nd	3rd	4th	5th
Alice	CH	MGH	BW	MTA	BID
Bob	BID	BW	MTA	MGH	CH
Clara	BW	BID	MTA	CH	MGH
Dorit	MGH	CH	MTA	BID	BW
Ernie	MTA	BW	CH	BID	MGH

→

	MGH	BW	BID	MTA	CH
Alice					
Bob					
Clara					
Dorit					
Ernie					

GS Algorithm: Running Time

- **Running Time:**
  - A careful implementation requires just $\approx n^2$ operations in the worst case and $\approx n^2$ space
  - Create an array of doctor x hospital in n^2 steps

	1st	2nd	3rd	4th	5th
Alice	CH	MGH	BW	MTA	BID
Bob	BID	BW	MTA	MGH	CH
Clara	BW	BID	MTA	CH	MGH
Dorit	MGH	CH	MTA	BID	BW
Ernie	MTA	BW	CH	BID	MGH

	MGH	BW	BID	MTA	CH
Alice	2nd	3rd	5th	4th	1st
Bob	4th	2nd	1st	3rd	5th
Clara	5th	1st	2nd	3rd	4th
Dorit	1st	5th	4th	3rd	2nd
Ernie	5th	2nd	4th	1st	3rd

GS Algorithm: Running Time

- **Running Time:**
  - A careful implementation requires just $\approx n^2$ operations in the worst case and $\approx n^2$ space
  - n^2 operations to convert doctor x rank -> doctor x hospital
  - Loop runs ≤ n^2 times; *2* operations to find *h* & *h'* in *d*'s preferences

  - $\approx n^2$ total operations

Real World Impact

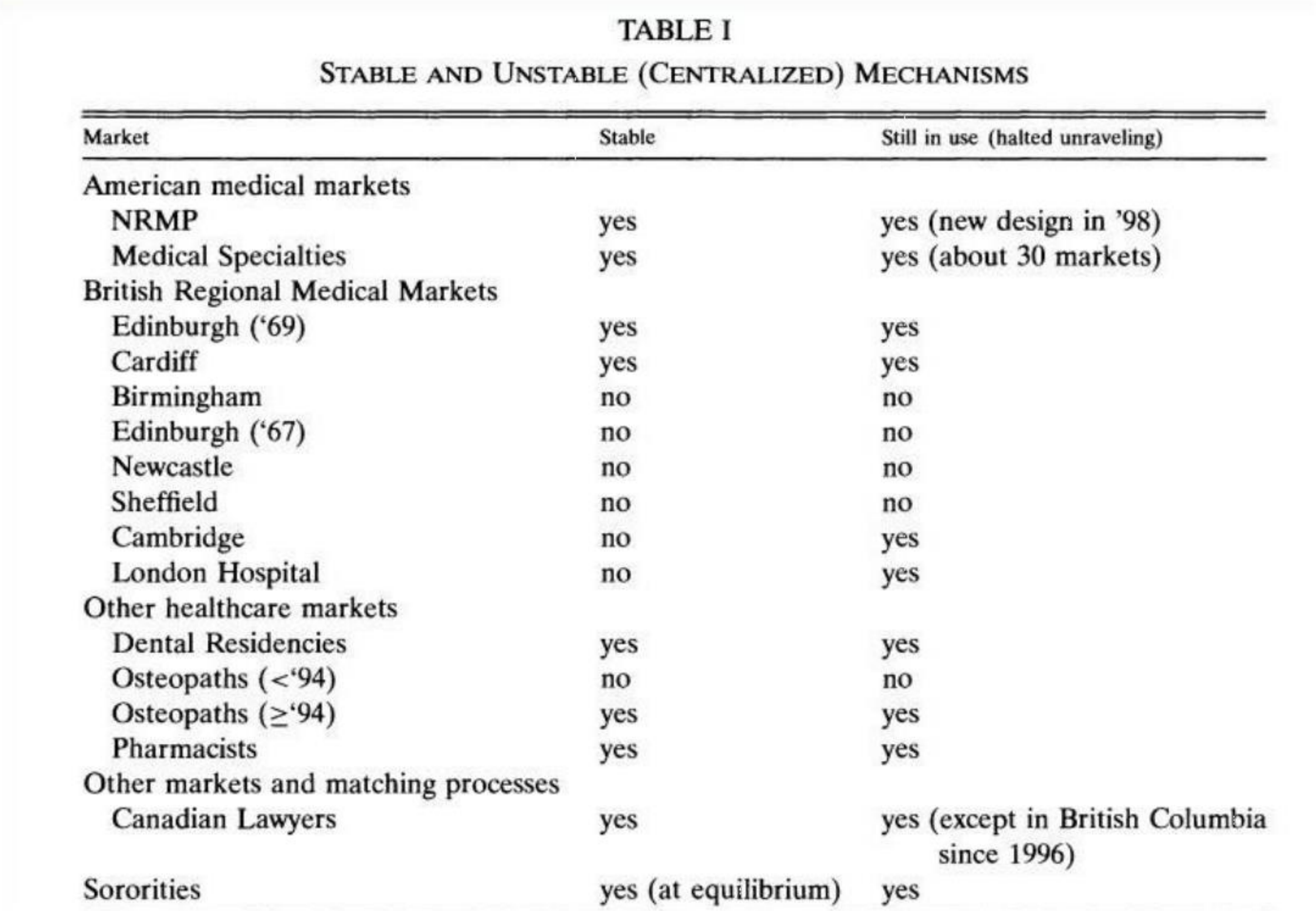


TABLE I

STABLE AND UNSTABLE (CENTRALIZED) MECHANISMS

Market	Stable	Still in use (halted unraveling)
American medical markets		
NRMP	yes	yes (new design in '98)
Medical Specialties	yes	yes (about 30 markets)
British Regional Medical Markets		
Edinburgh ('69)	yes	yes
Cardiff	yes	yes
Birmingham	no	no
Edinburgh ('67)	no	no
Newcastle	no	no
Sheffield	no	no
Cambridge	no	yes
London Hospital	no	yes
Other healthcare markets		
Dental Residencies	yes	yes
Osteopaths (<'94)	no	no
Osteopaths (≥'94)	yes	yes
Pharmacists	yes	yes
Other markets and matching processes		
Canadian Lawyers	yes	yes (except in British Columbia since 1996)
Sororities	yes (at equilibrium)	yes

Table 1. Reproduced from Roth (2002, Table 1).

Real World Impact

- **Doctors ⟷ Hospitals**
  - Have to deal with two-body problems
  - Have to make sure doctors do not game the system
- **Kidneys ⟷ Patients**
  - Not all matches are feasible (blood types)
  - Certain pairs must be matched
- **Students ⟷ Public Schools**
  - Siblings, walking zones, diversity

2012 Nobel Prize to Lloyd Shapley and Al Roth!