CS3000: Algorithms & Data
Paul Hand

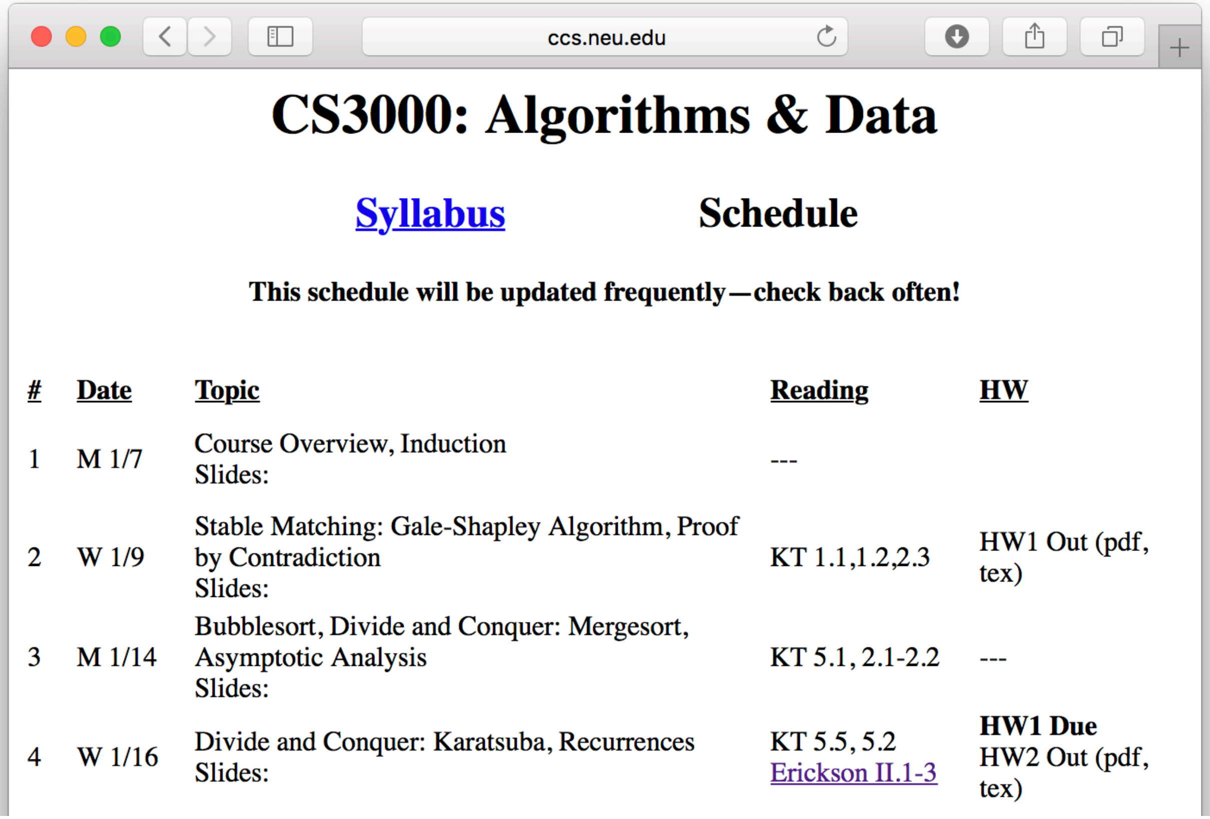
Lecture 2:
- Finish Induction
- Stable Matching: the Gale-Shapley Algorithm

Jan 9, 2019

Course Website

http://~~www.ccs~~.neu.edu/home/hand/teaching/cs3000-spring-2018/

khoury



CS3000: Algorithms & Data

Syllabus Schedule

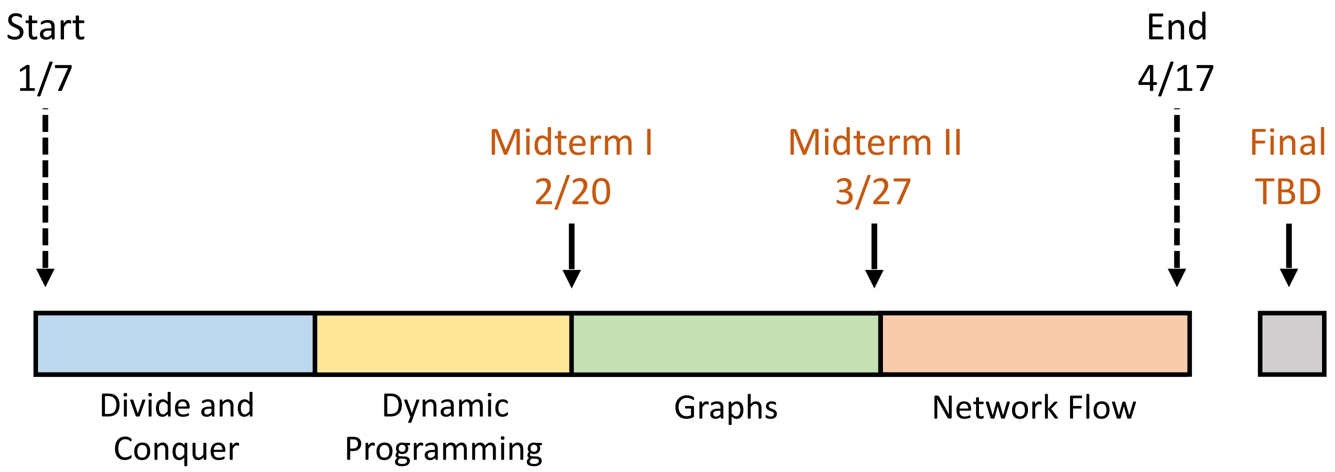
This schedule will be updated frequently—check back often!

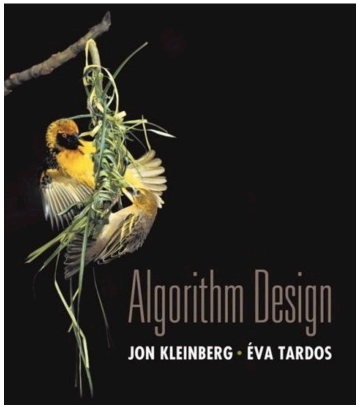
#	Date	Topic	Reading	HW
1	M 1/7	Course Overview, Induction Slides:	---	
2	W 1/9	Stable Matching: Gale-Shapley Algorithm, Proof by Contradiction Slides:	KT 1.1,1.2,2.3	HW1 Out (pdf, tex)
3	M 1/14	Bubblesort, Divide and Conquer: Mergesort, Asymptotic Analysis Slides:	KT 5.1, 2.1-2.2	---
4	W 1/16	Divide and Conquer: Karatsuba, Recurrences Slides:	KT 5.5, 5.2 Erickson II.1-3	**HW1 Due** HW2 Out (pdf, tex)

Homework Policies

- Homework will be submitted on Gradescope!
    - More details on Wednesday
    - Entry Code:**MKKEW2**
    - https://www.gradescope.com/courses/36055

Course Structure





Textbook:
Algorithm Design by Kleinberg and Tardos

More resources on the course website

Proof by induction:

You want to prove a statement $H(m)$ is true for all $m = 0, 1, 2, 3, \ldots$

Steps:

- Prove base case. Show $H(0)$ is true.
- Inductive Step. Assume $H(m)$. Show $H(m+1)$ is true.

  "inductive hypothesis"

Exercise

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

$$\begin{array}{r} 1000000 \\ -1 \\ \hline 111111 \end{array}$$

- **Proof by Induction:**

Base case: $n=1$, $\sum_{i=0}^{0} 2^i = 2^1 - 1 = 1$

General case

Assume $\sum_{i=0}^{n-1} 2^i = 2^n - 1$

$$\sum_{i=0}^{n} 2^i = \left(\sum_{i=0}^{n-1} 2^i + 2^n\right) = 2^n - 1 + 2^n$$
$$= 2^{n+1} - 1.$$

Stable Matching Problem and the Gale-Shapley Algorithm

Process for solving computational problems with algorithms

- Formulate problem and questions
- Play around
- Devise algorithm
- Determine how long it takes to run
- Determine if algorithm is correct
- Determine appropriate data structures

Stable Matching Problem

- Many job candidates (eg. doctors). Many jobs (eg. residency programs). You are to assign candidates to jobs. How should you do it?

Problem formulation

What information do you need? inputs to alg

What makes an output good?

What reasonable simplifications can you make?

In case of Stable Matching problem

Info: job candidate's preferences (over jobs)
employer's preferences (over candidates)
which candidate is qualified to work which job
can jobs be held simultaneously?

Simplifications: all candidates rank all jobs
all jobs rank all candidates
only one job can be taken

Good output: no (candidate, job) pair prefers each other over what they have

Problem formulation

What information do you need?

What makes an output good?

What reasonable simplifications can you make?

In case of Stable Matching problem

Info:

Simplifications:

Good output :

Problem formulation

What information do you need?

What makes an output good?

What reasonable simplifications can you make?

In case of Stable Matching problem

Info: job candidate's preferences (over jobs)
employer's preferences (over candidates)
which candidate is qualified to work which job
can jobs be held simultaneously?

Simplifications: all candidates rank all jobs
all jobs rank all candidates
only one job can be taken

Good output: no (candidate, job) pair prefers each other over what they have

Stable Matching problem – What makes an output good?

No candidate-job pair prefers each other over what they have.

Candidates $\{c_1, c_2, \cdots c_n\}$

jobs $\{j_1, j_2, \cdots, j_n\}$

pair $(\underbrace{c_1}_{\text{first person}}, \underbrace{j_5}_{\text{5th job}})$

Stable Matching - Questions

- For any set of preferences, does a stable matching exist?
- Can there be more than one stable matching?
- How can you find one if it exists?

Stable Matching - Introduce Formalism

A matching M is a set of candidate-job pairs $M = \{(c_1, j_3), (c_2, j_2), \cdots\}$ where no candidate or job appears more than once.

~~some~~ ppl have a job
not everyone

A matching is perfect if every candidate and job appears exactly once

"belongs to"

"c_1 is matched" means $(c_1, j) \in M$ for some job j.

"c_1 is matched to j_3" means $(c_1, j_3) \in M$

Stable Matching - Introduce Formalism

A matching is stable if it has no instabilities

An instability is any of the following

- $(c, j) \in M$, j' unmatched, and $c: j' > j$ "prefers"
- $(c, j) \in M$, c' unmatched, and $j: c' > c$
- $(c, j) \in M$ & $(c', j') \in M$ but $c: j' > j$, $j': c > c'$

note: prefs of j & c' dont matter

Activity: Consider the following preferences and matching. Is it stable? If not, find two different matchings that are.

Candidates: 1, 2, 3
Jobs: A, B, C

Find a (c,j) pair that both prefers each other

Candidates:

1 : $\boxed{B} \succ A \succ C$

2 : $\boxed{A} \succ C \succ B$

3 : $\boxed{C} \succ B \succ A$

Jobs

A : $\boxed{1} \succ 2 \succ 3$

B : $\boxed{3} \succ 2 \succ 1$

C : $\boxed{2} \succ 1 \succ 3$

Matching: (1, A)
(2, B)
(3, C)

$C: 2 > 3$

$2: C > B$

$M_1 = (\{(1,B), (2,A), (3,C)\})$

$M_2 = \{(1,A), (3,B), (2,C)\}$

Devise algorithm

Idea: Go through list of candidates in any order

Assign best job (according to candidate) that
prefers them to what that job has now

Repeat

Gale-Shapley Algorithm

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

Gale-Shapley Demo

	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

Activity: What are the first 4 steps of G-S algorithm?

(Assume it steps through jobs in order 1-4, afterwards starting over with 1 if necessary)

- Jobs: 1,2,3,4
- Candidates: A,B,C,D

Jobs' Preferences

1 : $\cancel{\boxed{A}} > B > C > D$

2 : $\boxed{B} > D > A > C$

3 : $\boxed{A} > B > C > D$

4 : $\boxed{D} > A > B > C$

Candidates' Preferences

A : $4 > 3 > 1 > 2$

B : $1 > 4 > 2 > 3$

C : $3 > 4 > 1 > 2$

D : $1 > 4 > 2 > 3$

Observations

- At all steps, the state of the algorithm is a matching
- Jobs make offers in descending order
- Candidates that get a job never become unemployed
- Candidates accept offers in ascending order

Gale-Shapley Algorithm

- Questions about the Gale-Shapley Algorithm:
  - Will this algorithm terminate? After how long?
  - 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 $\overset{\text{at most}}{\checkmark}$ n^2 iterations of the main loop, where n is number of candidates/jobs.

At most n^2 possible offers

At each iter, an offer is made. None repeated

So $\leq n^2$ iterations

GS Algorithm: Perfect Matching

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

GS Algorithm: Stable Matching

- **Stability:** GS algorithm outputs a stable matching
- Proof by contradiction:
    - Suppose there is an instability

GS Algorithm: Running Time

- **Running Time:**
  - A straightforward implementation requires at $\approx n^3$ operations, $\approx n^2$ space (memory).

GS Algorithm: Running Time

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

GS Algorithm: Running Time

- **Running Time:**
  - A careful implementation requires just $\approx n^2$ time and $\approx n^2$ space

GS Algorithm: Running Time

- **Running Time:**
  - A careful implementation requires just time and space

	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 $\approx n^2$ time and $\approx n^2$ space

Notes for instructor
Students may ignore
because they are repeated
elsewhere

Proofs:

Termination:

Each loop makes ~~at most~~ one new offer.
Only n^2 total possible offers

Perfect Matching:

Suppose a job is unmatched.

- Job offer was made to all candidates
- All candidates have a job
- So some candidate is matched with this job

Contradiction

Suppose a candidate is unmatched.

- Some job is unmatched. Contradiction

Stability:

As matching is perfect, only possible instability
is $(c, j) \in M$ and $c: j' \succ j$
$(c', j') \in M$ $\quad j': c \succ c'$

At some point, j' offered to c. c had a job at least as good as j'. c has a job at least as good as j'. Contradiction.