

CSE 421

Introduction to Algorithms

Lecture 1: Intro & Stable Matching

<https://cs.washington.edu/421>

Instructor

Paul Beame [he/him]

beame@cs

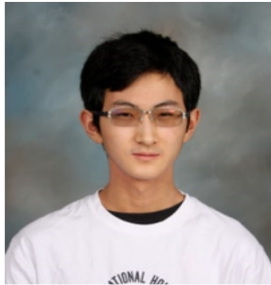
Specialty: **Complexity and Applications**

<https://homes.cs.washington.edu/~beame>

Office: CSE 668



A Dedicated Team of TAs



Daniel Gao



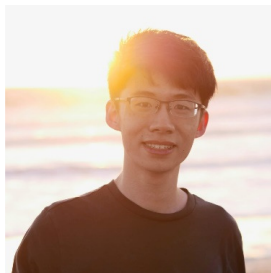
Ajay Harilal



Owen Boseley



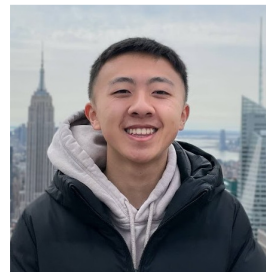
Edward Qin



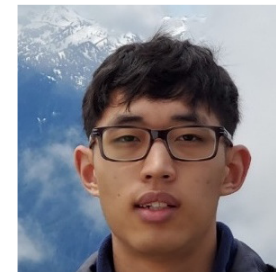
Glenn Sun



George King



Ben Zhang



Paul Han

See <https://cs.washington.edu/421/staff.html> to learn more about their backgrounds and interests!

Getting Started (Your TODO List)

- Make sure you are on Ed (a.k.a. EdStem)!
 - Check your inbox – and maybe your SPAM filter – for an invitation
- Attend your first Quiz Section tomorrow!
- Homework 1 will be out tonight
 - You will have enough to start on it after section tomorrow
 - Start thinking about it right away after that
- Get all the credit you deserve: Sign up for CSE 490Z
- Attend lecture and participate
 - Students who participate do better on average

Coursework

- 8 homework assignments roughly (due Wednesdays)
 - Typically 1 mechanical problem
3 long-form problems
- See the Homework Guide linked on the course website
- Start early to reduce amount of time you need to concentrate on them
 - Use your brain's background processing
- OK to talk with fellow students but solution write-up must be your own
 - See syllabus <https://cs.washington.edu/421/syllabus.html>
- Use of outside resources for solutions **forbidden** (see syllabus)
 - Generative AI does worse than almost anyone in the class would on their own...

Late Problem Days

- Late days per problem rather than for the whole assignment
 - Each problem is a separate Gradescope submission
 - Max 2 late days per problem; limit on total # of late problem days
 - You should submit anything that is done as soon as you are finished with it
 - See the syllabus for details

Exam dates

Midterm: Monday Nov 4 (In the evening, here, to give you more time for the same problems)

Final Exam: Standard exam time and place:
Monday Dec 9, 2:30-4:20 here

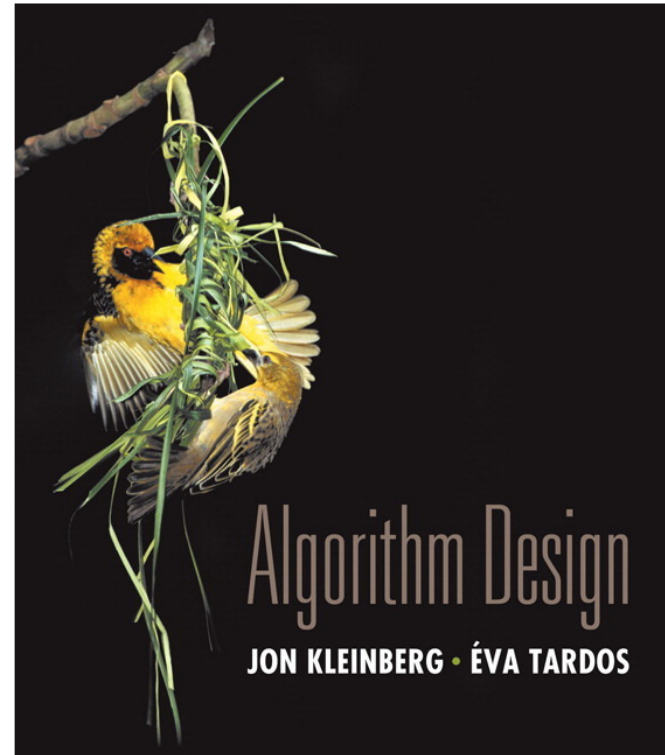
Grading scheme

- Homework 55%
- Midterm 15-20%
- Final Exam 25-30%

Textbook

Kleinberg-Tardos: Algorithm Design

- International Edition just as good
- Plus supplements on website
- Worth reading
 - Good for reading sequentially and learning how to think like an algorithm designer
 - Not as good for random access
- Not required
 - All required content will be on slides in lectures and quiz section



Introduction to Algorithms

- Basic techniques for the design and analysis of algorithms.
 - Develop a toolkit of ways to find efficient algorithms to solve problems
 - Prove that the algorithms are correct
 - Analyze their efficiency properties
 - Communicate these algorithms and their properties to others

On efficiency

- Originally, efficiency was important for many reasons but partly because computers were weak
- Now we have powerful computers but
 - Data has grown to be enormous
 - We need *even more* efficient algorithms at this scale
 - Computation has an *energy cost* and represents a significant part of society's total energy use
 - Efficient computing is essential to reducing that cost
 - Additional power is of little help for inefficient (e.g. brute force) solutions

Introduction to Algorithms

- Stable Matching

Matching Medical Residents to Hospitals

Goal: Given a set of preferences among hospitals and medical school residents (graduating medical students), design a *self-reinforcing* admissions process.

Unstable pair: applicant x and hospital y are *unstable* if:

- x prefers y to their assigned hospital.
- y prefers x to one of its admitted residents.

Stable assignment. Assignment with no unstable pairs.

- Natural and desirable condition.
- Individual self-interest will prevent any applicant/hospital side deal from being made.

Simpler: Stable Matching Problem

Goal: Given two groups of n people each, find a "suitable" matching.

- Participants rate members from opposite group.
- Each person lists members from the other group in order of preference from best to worst.

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Group P Preference Profile

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Group R Preference Profile

Stable Matching Problem

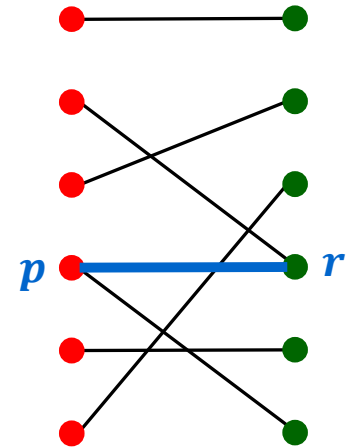
Perfect matching: everyone is matched to precisely one person from the other group

Stability: self-reinforcing, i.e. no incentive to undermine assignment by joint action.

- For a matching M , an unmatched pair $p-r$ from different groups is *unstable* if p and r prefer each other to current partners.
- Unstable pair $p-r$ could each improve by ignoring the assignment.

Stable matching: perfect matching with no unstable pairs.

Stable matching problem: Given the preference lists of n people from each of two groups, find a stable matching between the two groups if one exists.



Stable Matching Problem

Q: Is matching $(X,C), (Y,B), (Z,A)$ stable?

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Group P Preference Profile

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Group R Preference Profile

Stable Matching Problem

Q: Is matching $(X,C), (Y,B), (Z,A)$ stable?

A: No. B and X prefer each other.

	favorite ↓ 1 st	2 nd	least favorite ↓ 3 rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Group P Preference Profile

	favorite ↓ 1 st	2 nd	least favorite ↓ 3 rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Group R Preference Profile

Stable Matching Problem

Q: Is matching $(X,A), (Y,B), (Z,C)$ stable?

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Group P Preference Profile

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Group R Preference Profile

Stable Matching Problem

Q: Is matching $(X,A), (Y,B), (Z,C)$ stable?

A: Yes

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Group P Preference Profile

	favorite ↓		least favorite ↓
	1 st	2 nd	3 rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Group R Preference Profile

Variation: Stable Roommate Problem

Q. Do stable matchings always exist?

A. Not obvious a priori.

Stable roommate problem:

- $2n$ people; each person ranks others from 1 to $2n - 1$.
- Assign roommate pairs so that no unstable pairs.

	<i>1st</i>	<i>2nd</i>	<i>3rd</i>
<i>A</i>	B	C	D
<i>B</i>	C	A	D
<i>C</i>	A	B	D
<i>D</i>	A	B	C

$(A,B), (C,D) \Rightarrow$ B-C unstable
 $(A,C), (B,D) \Rightarrow$ A-B unstable
 $(A,D), (B,C) \Rightarrow$ A-C unstable

Observation: Stable matchings do not always exist for stable roommate problem.

Propose-And-Reject Algorithm

Propose-and-reject algorithm: [Gale-Shapley 1962]

Intuitive method that guarantees to find a stable matching.

- Members of one group P make *proposals*, the other group R receives proposals

```
Initialize each person to be free
while (some  $p$  in  $P$  is free) {
  Choose some free  $p$  in  $P$ 
   $r = 1^{\text{st}}$  person on  $p$ 's preference list to whom  $p$  has not yet proposed
  if ( $r$  is free)
    tentatively match  $(p, r)$  //  $p$  and  $r$  both engaged, no longer free
  else if ( $r$  prefers  $p$  to current tentative match  $p'$ )
    replace  $(p', r)$  by  $(p, r)$  //  $p$  now engaged,  $p'$  now free
  else
     $r$  rejects  $p$ 
}
```

Proof of Correctness: Termination (not obvious from the code)

Observation 1: Members of P propose in decreasing order of preference.

Claim: The Gale-Shapley Algorithm terminates after at most n^2 iterations.

Proof: Proposals are never repeated (by Observation 1) and there are only n^2 possible proposals. ■

It could be nearly that bad...

General form of this example will take $n(n - 1) + 1$ proposals.

	1 st	2 nd	3 rd	4 th	5 th
V	A	B	C	D	E
W	B	C	D	A	E
X	C	D	A	B	E
Y	D	A	B	C	E
Z	A	B	C	D	E

Preference Profile for P

	1 st	2 nd	3 rd	4 th	5 th
A	W	X	Y	Z	V
B	X	Y	Z	V	W
C	Y	Z	V	W	X
D	Z	V	W	X	Y
E	V	W	X	Y	Z

Preference Profile for R

Proof of Correctness: Perfection

Observation 2: Once a member of R is matched, they never become free; they only "trade up."

Claim: Everyone gets matched.

Proof:

- If no proposer is free then everyone is matched.
- After some p proposes to the last person on their list, all the r in R have been proposed to by someone (by p at least).
- By Observation 2, every r in R is matched at that point.
- Since $|P| = |R|$ every p in P is also matched. ■

Proof of Correctness: Stability

Claim: No unstable pairs in the final Gale-Shapley matching M

Proof: Consider a pair $p-r$ not matched by M

Case 1: p never proposed to r .

$\Rightarrow p$ prefers M -partner to r .

$\Rightarrow p-r$ is not unstable for M .

Case 2: p proposed to r .

$\Rightarrow r$ rejected p (right away or later when trading up)

$\Rightarrow r$ prefers M -partner to p .

$\Rightarrow p-r$ is not unstable for M . ■

Summary

Stable matching problem: Given n people in each of two groups, and their preferences, find a stable matching if one exists.

Stable: No pair of people both prefer to be with each other rather than with their assigned partner

Gale-Shapley algorithm: Guarantees to find a stable matching for *any* problem instance.

⇒ Stable matching always exists!