CSE 421 Section 1

Stable Matching

Administrivia & Introductions

Homework

- Submissions
    - LaTeX (highly encouraged)
        - overleaf.com
        - template and LaTeX guide posted on course website!
    - Word Editor that supports mathematical equations
    - Handwritten (not preferred, but fine as long as clear)
- All homeworks will be turned in via Gradescope
- Homeworks due on Wednesdays at 11.59pm
- You get **12 late <u>problems</u>** for the quarter and can use a maximum of 2 on any problem. If you submit multiple problems late, you use one late problem for each one (we hope you'll submit whatever you have finished before the deadline to make sure we start grading early)

Announcements & Reminders

- Section Materials
  - Handouts will be provided in at each section
  - Worksheets and sample solutions will be available on the class website later this evening

- HW1
  - Due Wednesday 4/8 at 11.59pm

Stable Matching

Stable Matching

Given n riders and n horses with preference lists, how can we find a stable matching so that all riders have horses and all horses have riders?

Perfect Matching:

- Each rider is paired with exactly one horse
- Each horse is paired with exactly one rider

Stability (in words): “No incentive to exchange partners”

Unstable pair: An unmatched pair (r,h) is unstable if they both prefer each other to their current matches

Stable Matching: A perfect matching with no unstable pairs

Gale-Shapley Algorithm

Algorithm to find a stable matching:

```
Initially all r in R and h in H are free
while there is a free r
      Let h be highest on r's list that r has not proposed to
      if h is free
             match (r,h)
      else //h is not free
             Let r' be the current match of h
             if h prefers r to r'
                    unmatch (r',h)
                    match (r,h)
```

Problem 1 – Gale-Shapley

Consider the following stable matching instance:

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3

(r_1, h_3)

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3 — (r_1, h_3)

r_2 chooses h_2 — $(r_1, h_3), (r_2, h_2)$

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3	(r_1, h_3)
r_2 chooses h_2	$(r_1, h_3), (r_2, h_2)$
r_3 chooses h_2	(r_1, h_3), **(r_2, h_2)**, **(r_3, h_2)**

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3	(r_1, h_3)
r_2 chooses h_2	$(r_1, h_3), (r_2, h_2)$
r_3 chooses h_2	(r_1, h_3), ~~(r_2, h_2)~~, (r_3, h_2)

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3	(r_1, h_3)
r_2 chooses h_2	$(r_1, h_3), (r_2, h_2)$
r_3 chooses h_2	(r_1, h_3), ~~(r_2, h_2)~~, (r_3, h_2)
r_2 chooses h_1	$(r_1, h_3), (r_2, h_1), (r_3, h_2)$

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3	(r_1, h_3)
r_2 chooses h_2	(r_1, h_3), (r_2, h_2)
r_3 chooses h_2	(r_1, h_3), ~~(r_2, h_2)~~, (r_3, h_2)
r_2 chooses h_1	(r_1, h_3), (r_2, h_1), (r_3, h_2)
r_4 chooses h_3	**(r_1, h_3)**, (r_2, h_1), (r_3, h_2), **(r_4, h_3)**

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3	(r_1, h_3)
r_2 chooses h_2	$(r_1, h_3), (r_2, h_2)$
r_3 chooses h_2	(r_1, h_3), ~~(r_2, h_2)~~, (r_3, h_2)
r_2 chooses h_1	$(r_1, h_3), (r_2, h_1), (r_3, h_2)$
r_4 chooses h_3	$(r_1, h_3), (r_2, h_1), (r_3, h_2)$, ~~$(r_4, h_3)$~~

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3	(r_1, h_3)
r_2 chooses h_2	$(r_1, h_3), (r_2, h_2)$
r_3 chooses h_2	(r_1, h_3), ~~(r_2, h_2)~~, (r_3, h_2)
r_2 chooses h_1	$(r_1, h_3), (r_2, h_1), (r_3, h_2)$
r_4 chooses h_3	$(r_1, h_3), (r_2, h_1), (r_3, h_2)$, ~~$(r_4, h_3)$~~
r_4 chooses h_4	$(r_1, h_3), (r_2, h_1), (r_3, h_2), (r_4, h_4)$

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm with riders proposing on the instance above. When choosing which free rider to propose next, always choose the one with the smallest index (e.g., if r_1 and r_2 are both free, always choose r_1).

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

r_1 chooses h_3	(r_1, h_3)
r_2 chooses h_2	$(r_1, h_3), (r_2, h_2)$
r_3 chooses h_2	(r_1, h_3), ~~(r_2, h_2)~~, (r_3, h_2)
r_2 chooses h_1	$(r_1, h_3), (r_2, h_1), (r_3, h_2)$
r_4 chooses h_3	$(r_1, h_3), (r_2, h_1), (r_3, h_2)$, ~~$(r_4, h_3)$~~
r_4 chooses h_4	$(r_1, h_3), (r_2, h_1), (r_3, h_2), (r_4, h_4)$

$(r_1, h_3), (r_2, h_1), (r_3, h_2), (r_4, h_4)$

Problem 1 – Gale-Shapley

b) Run the Gale-Shapley Algorithm with riders proposing on the same instance. But now, when choosing which free rider to propose next, always choose the one with the largest index. Do you get the same result?

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

c) Now run the algorithm with horses proposing, breaking ties by taking the free horse with the smallest index. Do you get the same result?

Work on parts b and c of this problem with the people around you, and then we'll go over it together!

Problem 1 – Gale-Shapley

b) Run the Gale-Shapley Algorithm with riders proposing on the same instance. But now, when choosing which free rider to propose next, always choose the one with the largest index. Do you get the same result?

$\mathbf{r_1}$: h_3, h_1, h_2, h_4
$\mathbf{r_2}$: h_2, h_1, h_4, h_3
$\mathbf{r_3}$: h_2, h_3, h_1, h_4
$\mathbf{r_4}$: h_3, h_4, h_1, h_2

$\mathbf{h_1}$: r_4, r_1, r_3, r_2
$\mathbf{h_2}$: r_1, r_3, r_2, r_4
$\mathbf{h_3}$: r_1, r_3, r_4, r_2
$\mathbf{h_4}$: r_3, r_1, r_2, r_4

Problem 1 – Gale-Shapley

b) Run the Gale-Shapley Algorithm with riders proposing on the same instance. But now, when choosing which free rider to propose next, always choose the one with the largest index. Do you get the same result?

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

The steps of the Gale-Shapley Algorithm with the rider with highest index proposing first:

r_4 chooses h_3	(r_4, h_3)
r_3 chooses h_2	$(r_3, h_2),(r_4, h_3)$
r_2 chooses h_2	$(r_3, h_2),(r_4, h_3)$
r_2 chooses h_1	$(r_2, h_1),(r_3, h_2),(r_4, h_3)$
r_1 chooses h_3	$(r_1, h_3),(r_2, h_1),(r_3, h_2)$
r_4 chooses h_4	$(r_1, h_3),(r_2, h_1),(r_3, h_2),(r_4, h_4)$

We ended up with the same result!

Problem 1 – Gale-Shapley

c) Now run the algorithm with horses proposing, breaking ties by taking the free horse with the smallest index. Do you get the same result?

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

Problem 1 – Gale-Shapley

c) Now run the algorithm with horses proposing, breaking ties by taking the free horse with the smallest index. Do you get the same result?

r_1: h_3, h_1, h_2, h_4
r_2: h_2, h_1, h_4, h_3
r_3: h_2, h_3, h_1, h_4
r_4: h_3, h_4, h_1, h_2

h_1: r_4, r_1, r_3, r_2
h_2: r_1, r_3, r_2, r_4
h_3: r_1, r_3, r_4, r_2
h_4: r_3, r_1, r_2, r_4

The steps of the Gale-Shapley Algorithm with horses proposing:

h_1 chooses r_4	(r_4, h_1)
h_2 chooses r_1	$(r_1, h_2),(r_4, h_1)$
h_3 chooses r_1	$(r_1, h_3),(r_4, h_1)$
h_2 chooses r_3	$(r_1, h_3),(r_3, h_2),(r_4, h_1)$
h_4 chooses r_3	$(r_1, h_3),(r_3, h_2),(r_4, h_1)$
h_4 chooses r_1	$(r_1, h_3),(r_3, h_2),(r_4, h_1)$
h_4 chooses r_2	$(r_1, h_3),(r_2, h_4),(r_3, h_2),(r_4, h_1)$

No, the result is different when we have the horses propose as opposed to the riders.

Induction

Induction

- You will be writing lots of induction proofs in this class in order to prove that your algorithms are correct.

- The style requirements for proofs in this class are less stringent than the style requirements from 311
  - there is a **style guide** doc on the course website (here) about how 421 proofs are different than what you did in 311

Induction Template

Let $P(n)$ be "(whatever you're trying to prove)".
We show $P(n)$ holds for all n by induction on n.

Base Case: Show $P(b)$ is true.

Inductive Hypothesis: Suppose $P(k)$ holds for an arbitrary $k \geq b$

Inductive Step: Show $P(k+1)$ (i.e. get $P(k) \rightarrow P(k+1)$)

Conclusion: Therefore, $P(n)$ holds for all n by the principle of induction.

Problem 3 – Induction Review

Consider the following claim:

Let $P(n)$ be "Every tree with at least n nodes has at least two nodes of degree-one."

a) What is the correct "skeleton" of the inductive step (i.e., the right things to assume and the right target)?

b) Prove the claim by induction.

Problem 3 – Induction Review

Consider the following claim:

Let $P(n)$ be "Every tree with at least n nodes has at least two nodes of degree-one."

a) What is the correct "skeleton" of the inductive step (i.e., the right things to assume and the right target)?

Work on this problem with the people around you, and then we'll go over it together!

Problem 3 – Induction Review

Consider the following claim:

Let $P(n)$ be "Every tree with at least n nodes has at least two nodes of degree-one."

a) What is the correct "skeleton" of the inductive step (i.e., the right things to assume and the right target)?

Problem 3 – Induction Review

Consider the following claim:

Let $P(n)$ be "Every tree with at least n nodes has at least two nodes of degree-one."

a) What is the correct "skeleton" of the inductive step (i.e., the right things to assume and the right target)?

We must start with "Let T' be an arbitrary tree with $k+1$ nodes."
Our conclusion will be that T' has at least two nodes of degree-one, so $P(k+1)$ holds.

KEY Induction Concept

It might be really tempting to structure the inductive step of this problem as something like, “start with an arbitrary tree T of size k nodes, and then add a node to it, making tree T' with $k+1$ nodes.”

This is a ***BAD*** idea! Then we’d have to cover every possible way to add on a node (and prove that we had actually dealt with every possible case), making the overall proof way more complicated and unwieldly.

Instead, we ***ALWAYS*** want to start with the bigger thing (in this case, with the arbitrary tree T' of size $k+1$) and find the smaller thing inside of it.

Problem 3 – Induction Review

Consider the following claim:

Let $P(n)$ be "Every tree with at least n nodes has at least two nodes of degree-one."

b) Prove the claim by induction.

Work on this problem with the people around you, and then we'll go over it together!

Problem 3 – Induction Review

b) Prove the claim by induction.

Let P(n) be "Every tree with at least n nodes has at least two nodes of degree-one." We prove the claim by induction on n.

Base Case: $n = 3$. There is only one undirected tree with three nodes. It has two nodes of degree-one.

Inductive Hypothesis: Suppose $P(n)$ holds for $n = 3, \ldots, k$ for an arbitrary $k \geq 3$.

Problem 3 – Induction Review

b) Prove the claim by induction.

Inductive Step: Let T' be an arbitrary tree with $k + 1$ nodes. Let u be a vertex of T' of degree-one (this first vertex exists by the fact), and call its neighbor v. Let T'' be the tree created by deleting u from T'.

Observe that, since u was degree-one, the only simple paths that used (u, v) had u as an endpoint (as once we use (u, v) to arrive at/leave u we cannot reuse it to leave/arrive). Thus T'' is still a connected tree, and we can apply the IH to T'' to conclude there are at least two vertices w_1, w_2 of T'' that are degree-one.

We now find the two degree-one nodes in the original tree T'. We know that u has degree-one (and is not the same as w_1 or w_2 since u was deleted to create T''). Since u has degree-one, it can only attach to at most one of w_1, w_2, thus at least one (the other one) of w_1, w_2 is an additional node of degree-one, as required.

Therefore, T' has the required degree-one vertices. Since T' is an arbitrary tree with $k + 1$ vertices, we have shown $P(k + 1)$.

That’s All, Folks!

Thanks for coming to section this week!
Any questions?