CSE 421 Section 1

Stable Matching

Administrivia & Introductions

Your Section TAs

- TA 1
    - Anything you want to say about yourself
- TA 2
    - content

Homework

- Submissions
  - LaTeX
  - Word Editor that supports mathematical equations
- All homeworks will be turned in via Gradescope
- Homeworks typically due on Wednesdays at 11:59pm

- Late day policy – 5 late days. Maximum 2 days late per assignment. Otherwise 25% per day.

Announcements & Reminders

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

- HW1
  - Due Wednesday 1/10 @ 11:59pm

Stable Matching

Stable Matching

Given $2n$ people, in two groups, **M** and **W**, of n people, with each person having a preference list for members of the other group, how can we find a stable matching between them?

Perfect Matching:

- Each person **m** in **M** is paired with exactly one person **w** in **W**
- Each person **w** in **W** is paired with exactly one person **m** in **M**

Stability: No ability to exchange partners

Unstable: An unmatched pair **m-w** is unstable if they both prefer each other to current matches

Stable Matching: perfect matching with no unstable pairs

Gale-Shapley Algorithm

Algorithm to find a stable matching:

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

Problem 1 – Gale-Shapley

Consider the following stable matching instance:

$\mathbf{m_1}$: w_3, w_1, w_2, w_4
$\mathbf{m_2}$: w_2, w_1, w_4, w_3
$\mathbf{m_3}$: w_2, w_3, w_1, w_4
$\mathbf{m_4}$: w_3, w_4, w_1, w_2

$\mathbf{w_1}$: m_4, m_1, m_3, m_2
$\mathbf{w_2}$: m_1, m_3, m_2, m_4
$\mathbf{w_3}$: m_1, m_3, m_4, m_2
$\mathbf{w_4}$: m_3, m_1, m_2, m_4

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3 (m_1, w_3)

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3 — (m_1, w_3)

m_2 chooses w_2 — $(m_1, w_3), (m_2, w_2)$

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3	(m_1, w_3)
m_2 chooses w_2	$(m_1, w_3), (m_2, w_2)$
m_3 chooses w_2	(m_1, w_3), **(m_2, w_2)**, **(m_3, w_2)?**

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

$\mathbf{m_1}$: w_3, w_1, w_2, w_4
$\mathbf{m_2}$: w_2, w_1, w_4, w_3
$\mathbf{m_3}$: w_2, w_3, w_1, w_4
$\mathbf{m_4}$: w_3, w_4, w_1, w_2
$\mathbf{w_1}$: m_4, m_1, m_3, m_2
$\mathbf{w_2}$: m_1, m_3, m_2, m_4
$\mathbf{w_3}$: m_1, m_3, m_4, m_2
$\mathbf{w_4}$: m_3, m_1, m_2, m_4

m_1 chooses w_3	(m_1, w_3)
m_2 chooses w_2	$(m_1, w_3), (m_2, w_2)$
m_3 chooses w_2	(m_1, w_3), $\cancel{(m_2, w_2)}$, (m_3, w_2)

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3	(m_1, w_3)
m_2 chooses w_2	$(m_1, w_3), (m_2, w_2)$
m_3 chooses w_2	$(m_1, w_3), \cancel{(m_2, w_2)}, (m_3, w_2)$
m_2 chooses w_1	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3	(m_1, w_3)
m_2 chooses w_2	$(m_1, w_3), (m_2, w_2)$
m_3 chooses w_2	$(m_1, w_3), \cancel{(m_2, w_2)}, (m_3, w_2)$
m_2 chooses w_1	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$
m_4 chooses w_3	**(m_1, w_3)**, $(m_2, w_1), (m_3, w_2)$, **(m_4, w_3)?**

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3	(m_1, w_3)
m_2 chooses w_2	$(m_1, w_3), (m_2, w_2)$
m_3 chooses w_2	$(m_1, w_3), \cancel{(m_2, w_2)}, (m_3, w_2)$
m_2 chooses w_1	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$
m_4 chooses w_3	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$ (m_4, w_3) failed

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3	(m_1, w_3)
m_2 chooses w_2	$(m_1, w_3), (m_2, w_2)$
m_3 chooses w_2	$(m_1, w_3), \cancel{(m_2, w_2)}, (m_3, w_2)$
m_2 chooses w_1	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$
m_4 chooses w_3	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$ (m_4, w_3) failed
m_4 chooses w_4	$(m_1, w_3), (m_2, w_1), (m_3, w_2), (m_4, w_4)$

Problem 1 – Gale-Shapley

a) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the smallest index (e.g., if m_1 and m_2 are both free, always choose m_1).

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

m_1 chooses w_3	(m_1, w_3)
m_2 chooses w_2	$(m_1, w_3), (m_2, w_2)$
m_3 chooses w_2	$(m_1, w_3), \cancel{(m_2, w_2)}, (m_3, w_2)$
m_2 chooses w_1	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$
m_4 chooses w_3	$(m_1, w_3), (m_2, w_1), (m_3, w_2)$ (m_4, w_3) failed
m_4 chooses w_4	$(m_1, w_3), (m_2, w_1), (m_3, w_2), (m_4, w_4)$

$(m_1, r_3), (m_2, r_1), (m_3, r_2), (m_4, r_4)$

Problem 1 – Gale-Shapley

b) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the *largest* index (e.g., if m_1 and m_2 are both free, always choose m_2). Do you get the same result?

$\mathbf{m_1}$: w_3, w_1, w_2, w_4
$\mathbf{m_2}$: w_2, w_1, w_4, w_3
$\mathbf{m_3}$: w_2, w_3, w_1, w_4
$\mathbf{m_4}$: w_3, w_4, w_1, w_2

$\mathbf{w_1}$: m_4, m_1, m_3, m_2
$\mathbf{w_2}$: m_1, m_3, m_2, m_4
$\mathbf{w_3}$: m_1, m_3, m_4, m_2
$\mathbf{w_4}$: m_3, m_1, m_2, m_4

c) Now run the algorithm with the same preferences but with the roles of **M** and **W** reversed (that is the w_i do the proposing) breaking ties by taking the free w_i with the smallest index i. 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 on the instance above. When choosing which free m in **M** to propose next, always choose the one with the *largest* index (e.g., if m_1 and m_2 are both free, always choose m_2). Do you get the same result?

$\mathbf{m_1}$: w_3, w_1, w_2, w_4
$\mathbf{m_2}$: w_2, w_1, w_4, w_3
$\mathbf{m_3}$: w_2, w_3, w_1, w_4
$\mathbf{m_4}$: w_3, w_4, w_1, w_2

$\mathbf{w_1}$: m_4, m_1, m_3, m_2
$\mathbf{w_2}$: m_1, m_3, m_2, m_4
$\mathbf{w_3}$: m_1, m_3, m_4, m_2
$\mathbf{w_4}$: m_3, m_1, m_2, m_4

Problem 1 – Gale-Shapley

b) Run the Gale-Shapley Algorithm on the instance above. When choosing which free m in **M** to propose next, always choose the one with the *largest* index (e.g., if m_1 and m_2 are both free, always choose m_2). Do you get the same result?

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

The steps of the Gale-Shapley Algorithm with the free m in **M** with largest index proposing first:

m_4 chooses w_3	(m_4, w_3)
m_3 chooses w_2	$(m_3, w_2),(m_4, w_3)$
m_2 chooses w_2	$(m_3, w_2),(m_4, w_3)$ (m_2, w_2) failed
m_2 chooses w_1	$(m_2, w_1),(m_3, w_2),(m_4, w_3)$
m_1 chooses w_3	$(m_1, w_3),(m_2, w_1),(m_3, w_2),(\cancel{m_4, w_3})$
m_4 chooses w_4	$(m_1, w_3),(m_2, w_1),(m_3, w_2),(m_4, w_4)$

We ended up with the same result!

Problem 1 – Gale-Shapley

c) Now run the algorithm with the people in **W** proposing, breaking ties by taking the free w_i with the smallest index. Do you get the same result?

$\mathbf{m_1}$: w_3, w_1, w_2, w_4
$\mathbf{m_2}$: w_2, w_1, w_4, w_3
$\mathbf{m_3}$: w_2, w_3, w_1, w_4
$\mathbf{m_4}$: w_3, w_4, w_1, w_2

$\mathbf{w_1}$: m_4, m_1, m_3, m_2
$\mathbf{w_2}$: m_1, m_3, m_2, m_4
$\mathbf{w_3}$: m_1, m_3, m_4, m_2
$\mathbf{w_4}$: m_3, m_1, m_2, m_4

Problem 1 – Gale-Shapley

c) Now run the algorithm with the people in **W** proposing, breaking ties by taking the free w_i with the smallest index. Do you get the same result?

m_1: w_3, w_1, w_2, w_4
m_2: w_2, w_1, w_4, w_3
m_3: w_2, w_3, w_1, w_4
m_4: w_3, w_4, w_1, w_2

w_1: m_4, m_1, m_3, m_2
w_2: m_1, m_3, m_2, m_4
w_3: m_1, m_3, m_4, m_2
w_4: m_3, m_1, m_2, m_4

The steps of the Gale-Shapley Algorithm with the w in **W** proposing:

w_1 chooses p_4	(m_4, w_1)	
w_2 chooses p_1	$(m_1, w_2),(m_4, w_1)$	
w_3 chooses p_1	$(m_1, w_3),\cancel{(m_1, w_2)},(m_4, w_1)$	
w_2 chooses p_3	$(m_1, w_3),(m_3, w_2),(m_4, w_1)$	
w_4 chooses p_3	$(m_1, w_3),(m_3, w_2),(m_4, w_1)$	(m_3, w_4) failed
w_4 chooses p_1	$(m_1, w_3),(m_3, w_2),(m_4, w_1)$	(m_1, w_4) failed
w_4 chooses p_2	$(m_1, w_3),(m_2, w_4),(m_3, w_2),(m_4, w_1)$	

No, the result is different when we have the w in **W** propose as opposed to the m in **M**.

Problem 2 – True/False

Determine if the following statements are true or false:

a) True or false? In every instance of the stable matching problem, there is a stable matching containing a pair (m,w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.

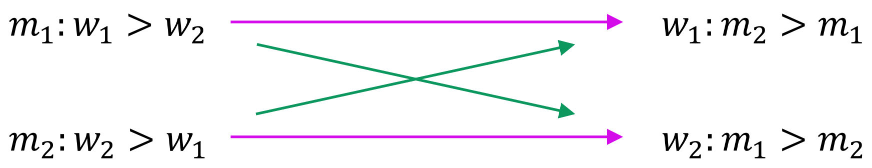
Work on this problem with the people around you, and then we'll go over it together!

Problem 2 – True/False

Determine if the following statements are true or false:

a) True or false? In every instance of the stable matching problem, there is a stable matching containing a pair (m,w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.

False. Note the following counterexample. In both purple and green matchings, this pair does not exist:



Problem 2 – True/False

Determine if the following statements are true or false:

b) Consider an instance of the stable matching problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then in every stable matching S for this instance, the pair (m,w) belong to S.

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

Problem 2 – True/False

Determine if the following statements are true or false:

b) Consider an instance of the stable matching problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then in every stable matching S for this instance, the pair (m,w) belong to S.

True. Let m be matched to $p \neq w$ and w be matched to $q \neq m$ in an arbitrary stable matching S. We see then that (m, w) is instable for S. Hence (m, w) always belongs to a stable matching S.

Induction

Induction

- You will be writing induction proofs in this class.
- The style requirements for proofs in this class are less stringent than the style requirements from 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.

<u>Base Case:</u> Show $P(b)$ is true.

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

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

<u>Conclusion:</u> 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 n nodes has $n - 1$ edges."

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 n nodes has $n-1$ edges."

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 n nodes has $n - 1$ edges."

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 n nodes has $n - 1$ edges."

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 n nodes has $n-1$ edges."

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 n nodes has n−1 edges." We prove the claim by induction on n.

Base Case: $n = 1$. This tree clearly has 1 – 1 = 0 edges.

Inductive Hypothesis: Suppose $P(n)$ holds for $n = 1, \ldots, k$ for an arbitrary $k \geq 1$.

Inductive Step: Let T be a tree with $k + 1$ nodes. Let d denote the node that is a leaf in the tree T. Let T' be the graph we get after we remove d and its connected edges. We see T' is still connected and undirected, and still acyclic as the removal of the node has not created a cycle. Hence, T' is a tree.

By IH, we see that T' has $k - 1$ edges. Let us add the node d back to the tree. Since it is a leaf node, it only has one edge, and therefore T has $(k - 1) + 1 = k$ edges.

Proof or Counterexample?

Prove or Disprove?

Often, you will be given a statement, and then asked to either prove or disprove it. This can be stressful! How do you know which you should start with?

The best way to begin, especially when you don’t know if the claim is even true, is to try to understand it better by producing some examples. This has two main benefits that will help, whether you end up proving or disproving the claim:

1) You get a better understanding of the statement so now you have a clear method of approach, OR
2) You find a counterexample, which allows you to easily write a quick proof that the statement is false!

Problem 2 – A Quick Proof

Is it possible to have a stable matching instance with more than 2 stable matchings? If so, give an instance and at least 3 stable matchings. If not, prove that every instance has at most 2 stable matchings.

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

Problem 2 – A Quick Proof

Is it possible to have a stable matching instance with more than 2 stable matchings? If so, give an instance and at least 3 stable matchings. If not, prove that every instance has at most 2 stable matchings.

Problem 2 – A Quick Proof

Is it possible to have a stable matching instance with more than 2 stable matchings? If so, give an instance and at least 3 stable matchings. If not, prove that every instance has at most 2 stable matchings.

Consider the following instance:

$\mathbf{m_1}$: w_1, w_2, w_3, w_4
$\mathbf{m_2}$: w_2, w_1, w_4, w_3
$\mathbf{m_3}$: w_3, w_4, w_1, w_2
$\mathbf{m_4}$: w_4, w_3, w_2, w_1

$\mathbf{w_1}$: m_2, m_1, m_4, m_3
$\mathbf{w_2}$: m_1, m_2, m_3, m_4
$\mathbf{w_3}$: m_4, m_3, m_2, m_1
$\mathbf{w_4}$: m_3, m_4, m_1, m_2

This instance has four stable matchings:

$(m_1, w_1),(m_2, w_2),(m_3, w_3),(m_4, w_4)$
$(m_1, w_1),(m_2, w_2),(m_3, w_4),(m_4, w_3)$
$(m_1, w_2),(m_2, w_1),(m_3, w_3),(m_4, w_4)$
$(m_1, w_2),(m_2, w_1),(m_3, w_4),(m_4, w_3)$

That’s All, Folks!

Thanks for coming to section this week!
Any questions?