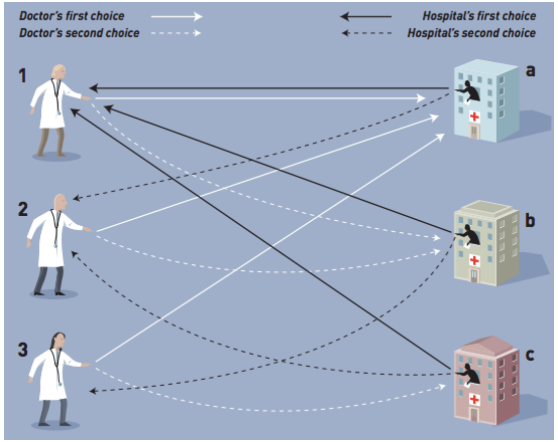
Lecture 1

Stable Matching



What is Algorithm Design?

What are some complex problems we may write a computer program to solve?

- ❖ Computing similarity between DNA sequences
- ❖ Routing packets on the Internet
- ❖ Scheduling final exams at a college
- ❖ Assign medical residuals to hospitals
- ❖ Find all occurrences of a phrase in a large collection of documents
- ❖ Finding the smallest number of coffee shops that can be built in the US such that everyone is within 20 minutes of a coffee shop

DNA Sequence Similarity

- **Input**: two n-length strings s_1 and s_2
  - $s_1 = AGGCTACC$
  - $s_2 = CAGGCTAC$
- **Output**: minimum number of insertions/deletions to transform s_1 into s_2
- **Algorithm**: ???
- Even if the objective is precisely defined, we are often not ready to start coding right away!

What is Algorithm Design?

1. Formulate the problem precisely
2. Design an algorithm
3. Prove the algorithm is correct
4. Analyze its running time

This is an iterative process!

❖ Sometimes we’ll redesign an algorithm to prove that it is correct

Stable Matching

Matching applicants to medical residency programs:

- m applicants
- m slots at hospitals
- Applicants have preferences over hospitals and vice versa

What is a good way to match?

- Matching should be *stable*; participates have no incentive to switch
- One way to identify a good matching!
- Gale-Shapley algorithm

Stable Matching

- ❖ Work on developing and applying a solution to this problem won the 2012 Nobel Prize in Economics
- ❖ Lloyd Shapley developed Stable matching theory and the Gale-Shapley algorithm
- ❖ Alvin Roth applied Gale-Shapley to matching residents with hospitals, students with schools, and organ doners with patients

Problem Formulation

- **Input**:
  - n residents
  - n hospitals
  - preference lists
- **Output**:
  - A stable matching
- A **matching** is an assignment of residents to hospitals
  - a set M of resident-hospital pairs, each resident/hospital in exactly one pair
- **Goal**: output a stable matching

Stable Matching

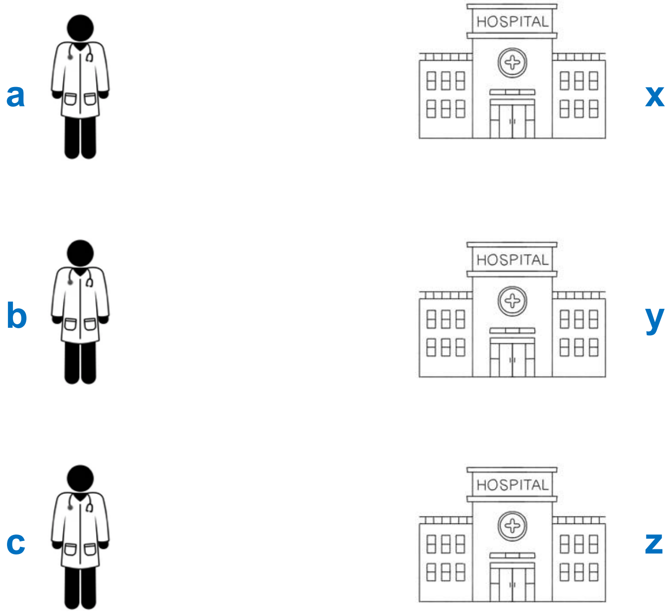
What does *stable* even mean?

❖ "participates have no incentive to switch"

❖ Matching has no unstable pair

An **unstable pair** is an unmatched pair that prefer each other to their assigned matches

Can we determine is a matching is stable just from the matching?



Stable Matching

What does *stable* even mean?

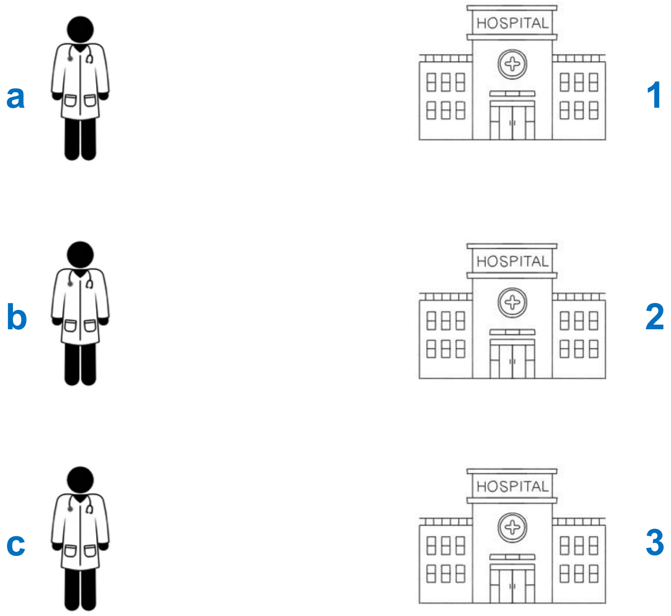
❖ "participates have no incentive to switch"

❖ Matching has no unstable pair

An **unstable pair** is an unmatched pair that prefer each other to their assigned matches

Can we determine is a matching is stable just from the matching?

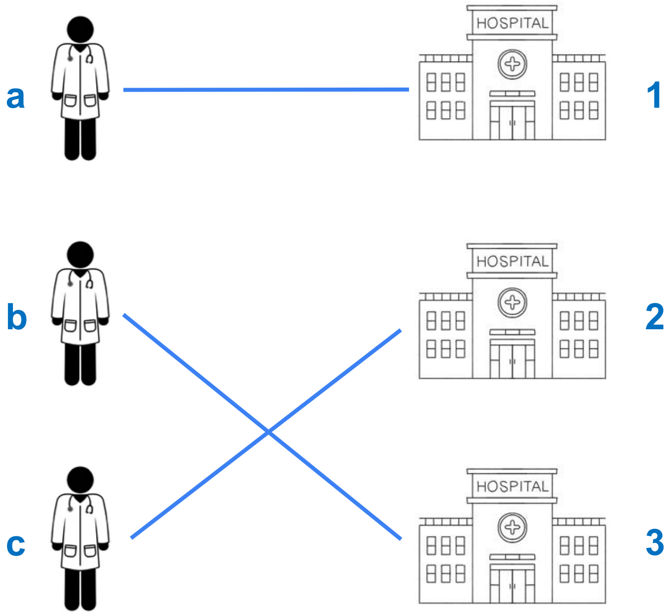
❖ No, we need to know their preferences!



Exercise 1

a:	**1**	2	3
b:	2	1	**3**
c:	1	**2**	3

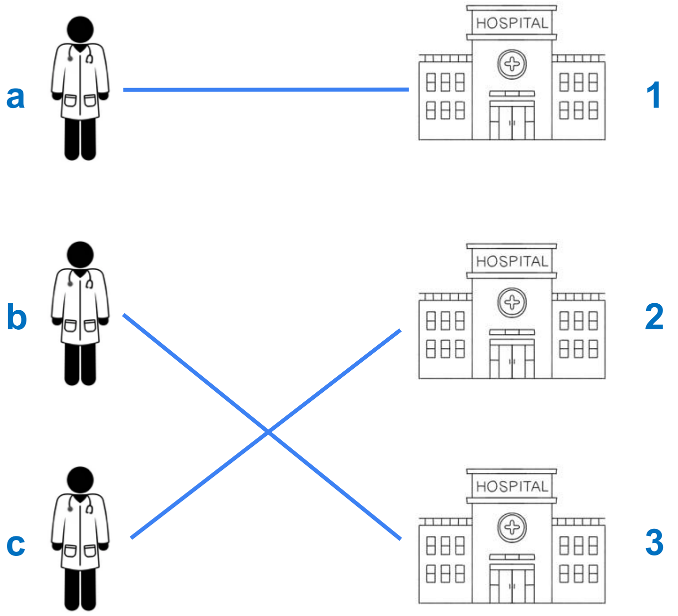
1:	b	**a**	c
2:	a	b	**c**
3:	a	**b**	c



Exercise 1

a:	**1**	2	3
b:	2	1	**3**
c:	1	**2**	3

1:	b	**a**	c
2:	a	b	**c**
3:	a	**b**	c



Which pair is an unstable pair in this matching?

i. (a, 2)

ii. (b,1)

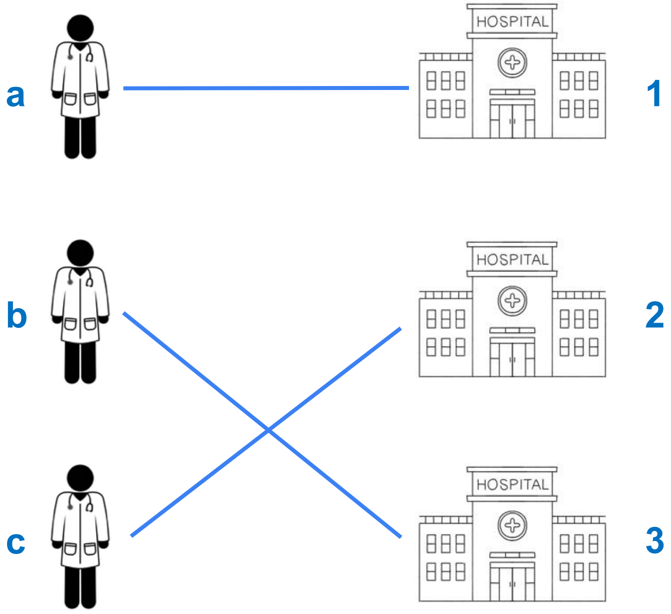
iii. (b,3)

iv. None of the above

Exercise 1

a:	**1**	2	3
b:	2	1	**3**
c:	1	**2**	3

1:	b	**a**	c
2:	a	b	**c**
3:	a	**b**	c



Which pair is an unstable pair in this matching?

i. (a, 2)

ii. (b,1)

iii. (b,3)

iv. None of the above

Example: Universal preferences

a:	1	2
b:	1	2

1:	a	b
2:	a	b

- $M = \{(a, 1), (b, 2)\}$? stable
- $M = \{(a, 2), (b, 1)\}$? not stable

Exercise 2: Inconsistent preferences

a:	1	2
b:	2	1

1:	b	a
2:	a	b

a


1

b


2

Which matching is stable?

i. $M = \{(a, 2), (b, 1)\}$

ii. $M = \{(a, 1), (b, 2)\}$

iii. Neither

iv. Both

Exercise 2: Inconsistent preferences

a:	1	2
b:	2	1

1:	b	a
2:	a	b

Which matching is stable?

i. $M = \{(a, 2), (b, 1)\}$

ii. $M = \{(a, 1), (b, 2)\}$

iii. Neither

iv. Both

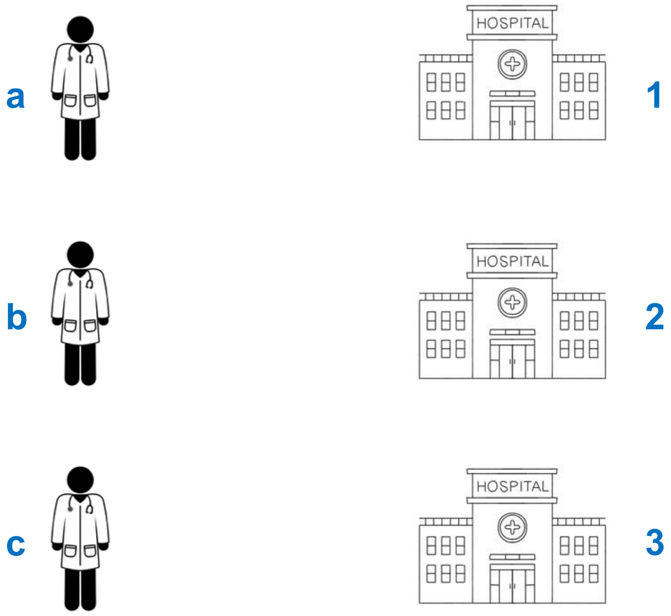
There can be multiple stable matchings for a given problem!

Designing an Algorithm

a:	1	2	3
b:	2	1	3
c:	1	3	2

1:	c	a	b
2:	a	b	c
3:	a	b	c

- ❖ Let's try building M incrementally

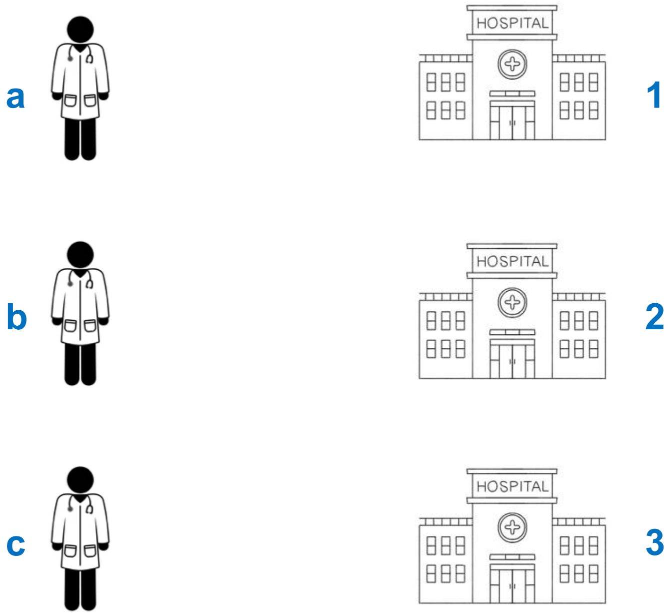


Designing an Algorithm

a:	1	2	3
b:	2	1	3
c:	1	3	2

1:	c	a	b
2:	a	b	c
3:	a	b	c

- Let's try building M incrementally
- Unmatched hospitals take turns offering to students and propose in order of preference
- Students take first offer then "trade up" if they receive better offer



Propose-and-Reject (Gale-Shapley) Algorithm

Initially all residents and hospitals are free

while some hospital is free and hasn't made offers to every resident **do**

    Choose a hospital h

    Let r be the highest ranked resident to whom h has no offered

    if r is free **then**

        r and h become matched

    else if r is matched to h' but prefers h to h' **then**

        h' becomes unmatched

        h and r become matched

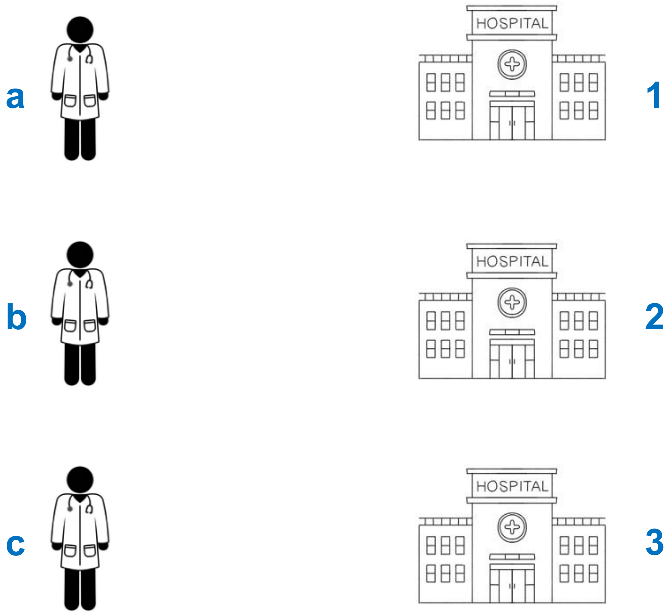
    else

        r rejects h and h remains free

Running the Propose-and-Reject algorithm

a:	1	2	3
b:	2	1	3
c:	1	3	2

1:	c	a	b
2:	a	b	c
3:	a	b	c

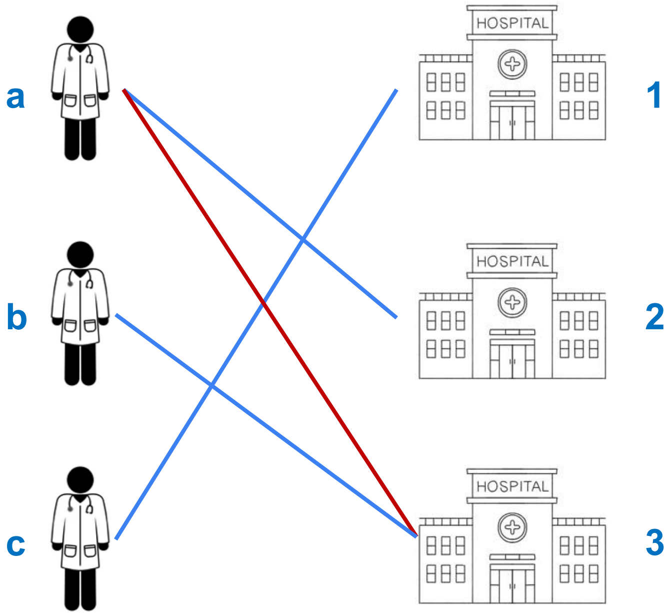


Running the Propose-and-Reject algorithm

a:	1	2	3
b:	2	1	3
c:	1	3	2

1:	c	a	b
2:	a	b	c
3:	a	b	c

- 1 matches with c
- 2 matches with a
- 3 proposes to a but is rejected
- 3 matches with b



Stability

Does the algorithm return a stable matching?

- Suppose (r, h) is an unstable pair
  - r is matched to h' but prefers h to h'
  - h is matched to r' but prefers r to r'
- Did h offer to r? Yes, by F2, since h offered to r' who is ranked lower
- Did r accept the offer from h? Maybe initially, but r must eventually reject h for another hospital, and, by F1, r prefers final college h' to h ($\rightarrow\leftarrow$)

Analyzing the Algorithm

Goal: prove that the algorithm always returns a stable matching

Observations:

- ❖ (F1) Residents accept their first offer, after which they stay matched and only improve their match during the algorithm
- ❖ (F2) Hospitals propose to residents sequentially in order of preferences

Termination

Does the algorithm terminate?

- ❖ In each round, some hospital proposes to a new resident in their list (by F2)
- ❖ Each hospital makes at most n proposals
- ❖ Then, there are at most n^2 proposals
- ❖ Implies there are at most n^2 rounds

Validity

Does the algorithm return a valid matching?

❖ For contradiction, suppose that resident r and hospital h are unmatched at the end of the algorithm

❖ Implies r was never matched during the algorithm (by F1)

❖ But h proposed to every student (by F2 and termination)

❖ When h proposed to r, she was unmatched but must have rejected h ($\rightarrow\leftarrow$)

Validity

Does the algorithm return a valid matching?

- ❖ For contradiction, suppose that resident r and hospital h are unmatched at the end of the algorithm
- ❖ Implies r was never matched during the algorithm (by F1)
- ❖ But h proposed to every student (by F2 and termination)
- ❖ When h proposed to r, she was unmatched but must have rejected h $(\rightarrow\leftarrow)$

Key idea: proof by contradiction

Symmetry

What if we had the residents propose rather than the hospitals?

Symmetry

What if we had the residents propose rather than the hospitals?

- ❖ May obtain a different stable matching
- ❖ When hospitals propose, we best satisfy the hospitals' preferences
- ❖ When residents propose, we best satisfy the residents' preferences

Next Time

- Begin looking at tools for analyzing algorithms, e.g., Big-O notation