

# Lecture 17

Gale-Shapley (Deferred Acceptance) Algorithm

# Help evaluate the Embedded Ethics Program!



<https://tinyurl.com/embeddedethicsw26>

10-15 minute survey, taking it (or not) won't impact your grade in the class in any way. The teaching team won't know who participates.

Option to provide your **Stanford email address** to receive a **\$10 gift card**, up to the first 800 participants. Compensation once per quarter (SUNet login required).

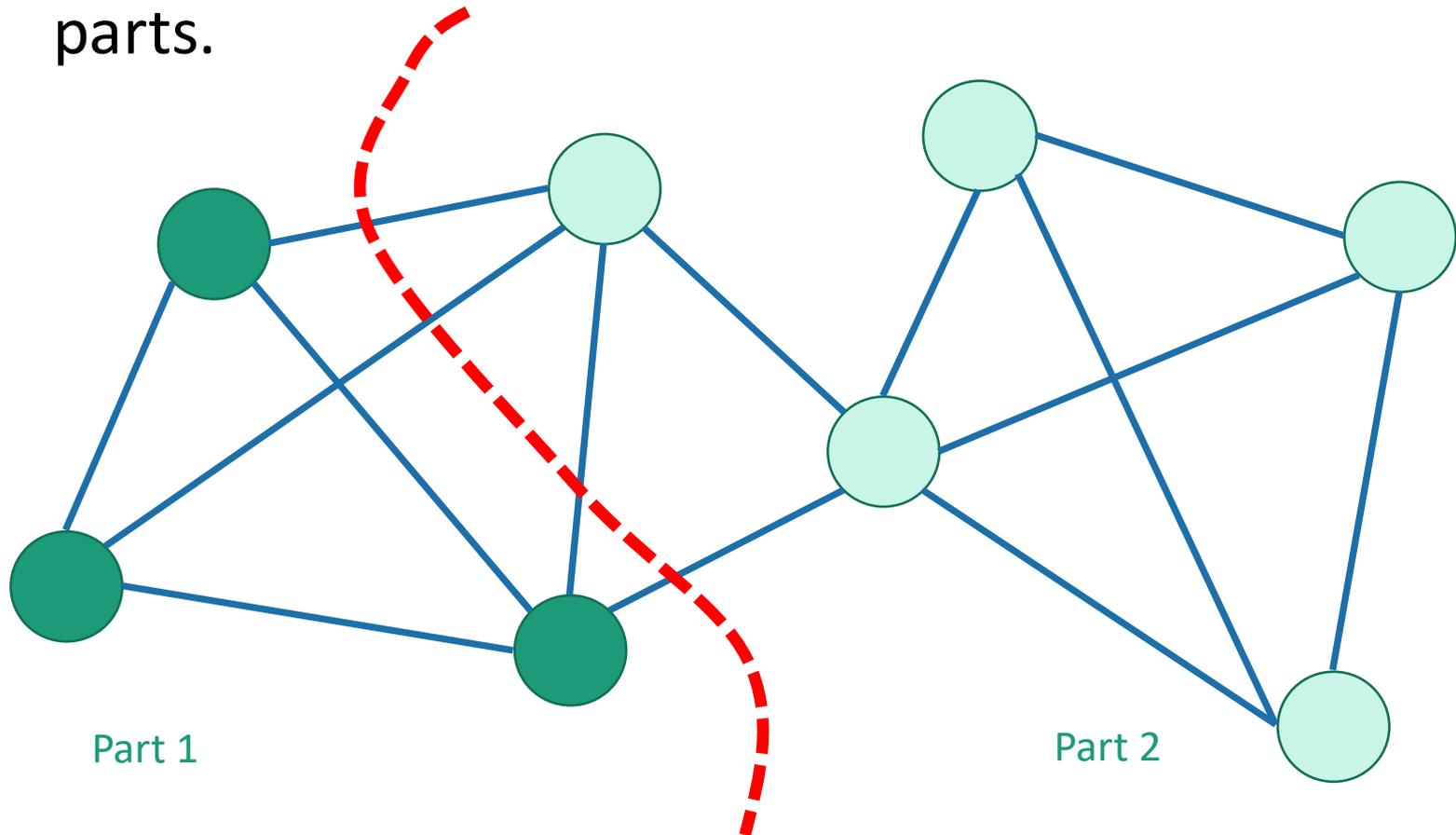
Questions? Email [embeddedethics@stanford.edu](mailto:embeddedethics@stanford.edu)

# Announcements

- HW8 due on Wednesday
- Material covered this week won't be on final
  - Today: stable matching algorithm
    - Peek inside the interface between economic and CS
    - Used to match medical students to residencies
    - Creators won Nobel prize for work including this algorithm
  - Wednesday: more “greatest hits” in algorithms

# Recap: min-cuts in graphs

- A cut is a partition of the vertices into two **nonempty** parts.



# Recap: One way to greedy algorithms

- Greedy algorithms

- Make a series of choices.
  - Choose this activity, then that one, ..
  - Never backtrack.
- Show (or hope) that your choice never rules out success.
  - At every step, there exists an optimal solution consistent with the choices we've made so far.
- At the end of the day:
  - you've built only one solution,
  - never having ruled out success,
  - **so your solution must be correct.**

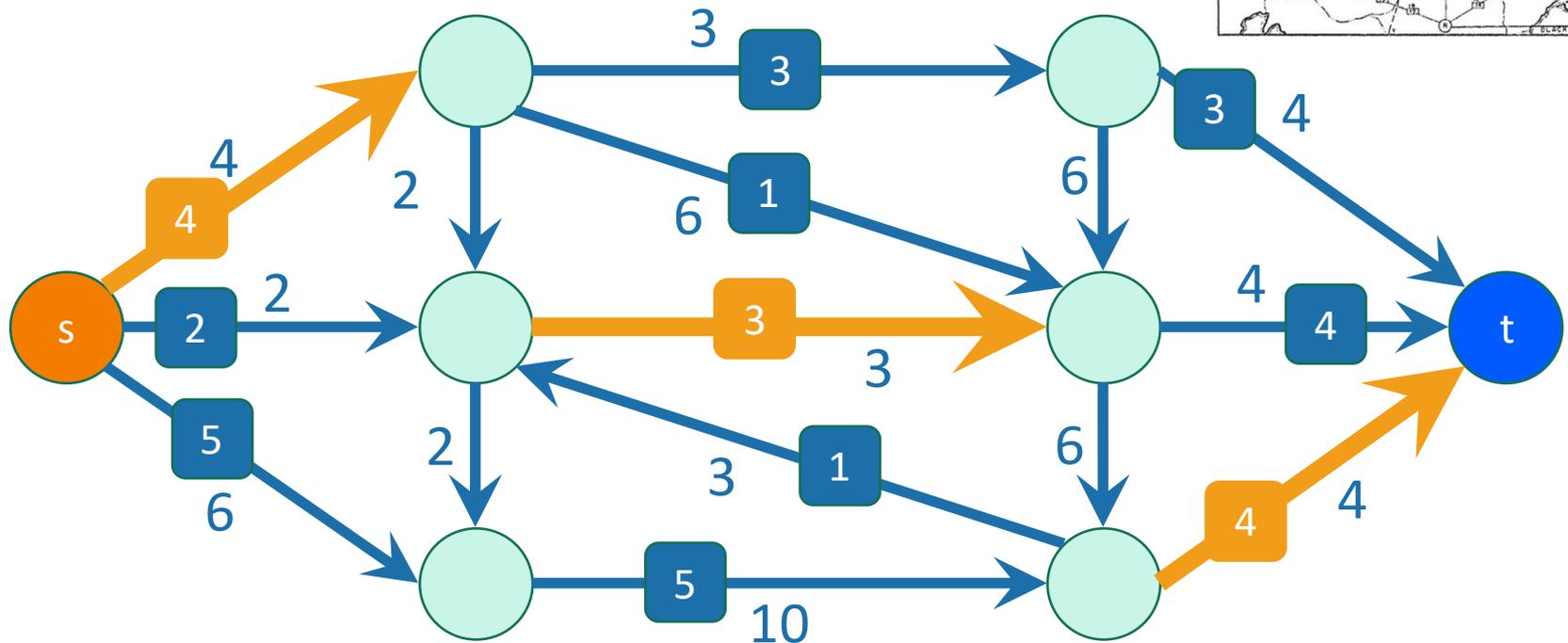
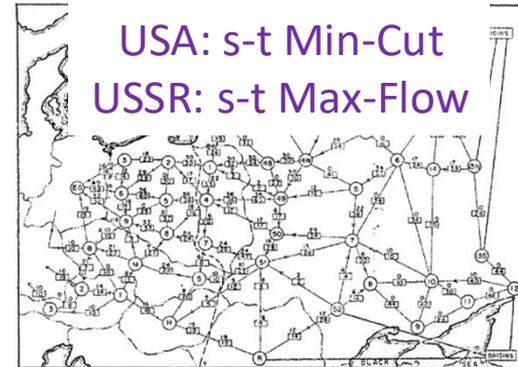
# Recap: A different approach to greedy

- Greedy algorithms

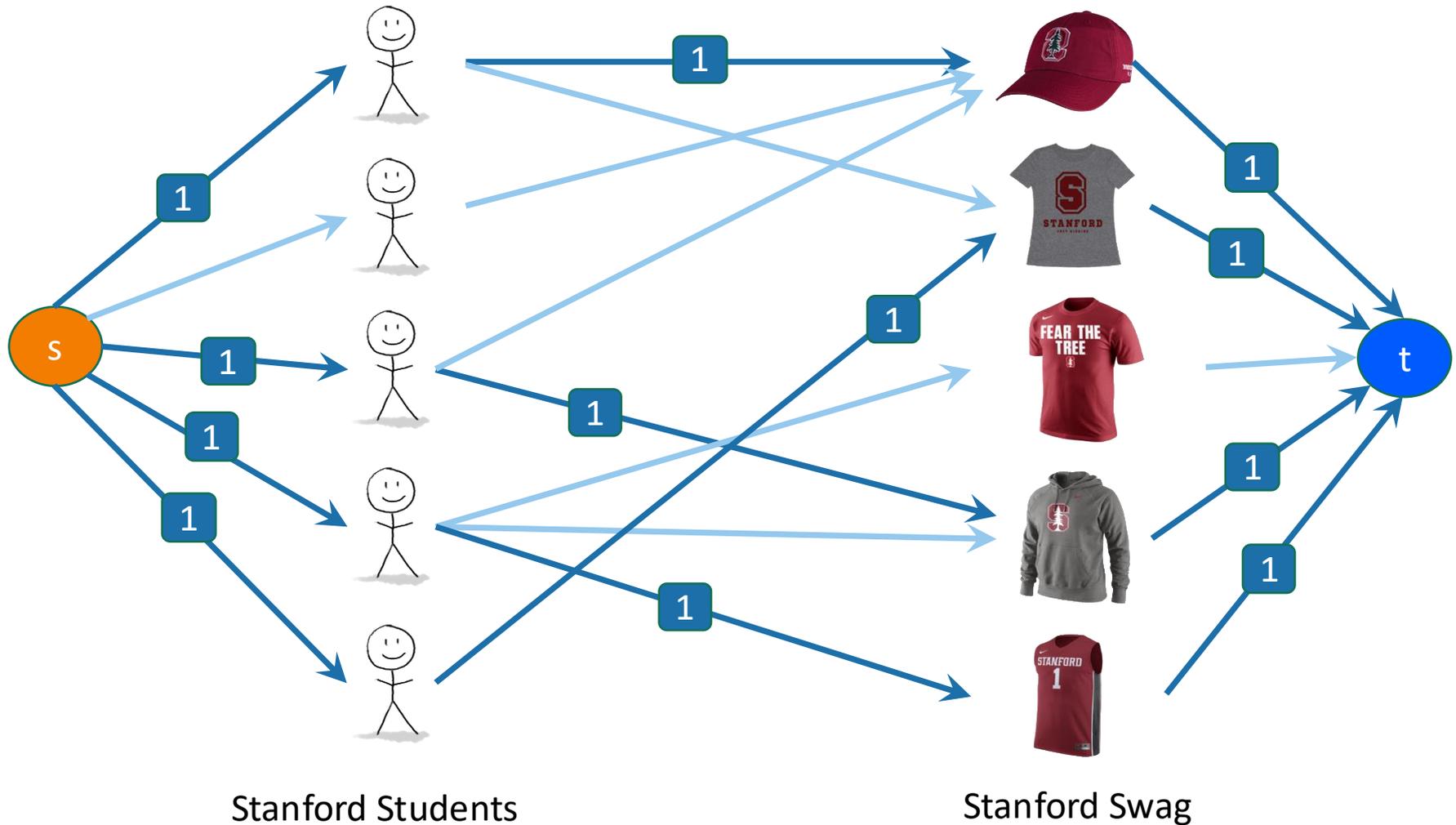
- Make a series of choices.
  - Choose this activity, then that one, ..
  - ~~• Never backtrack.~~
- **Instead:** At each step, free to revert any of the choices we've already made – as long as the solution is improving!

# Recap: Ford-Fulkerson algorithm for s-t min-cut / max-flow

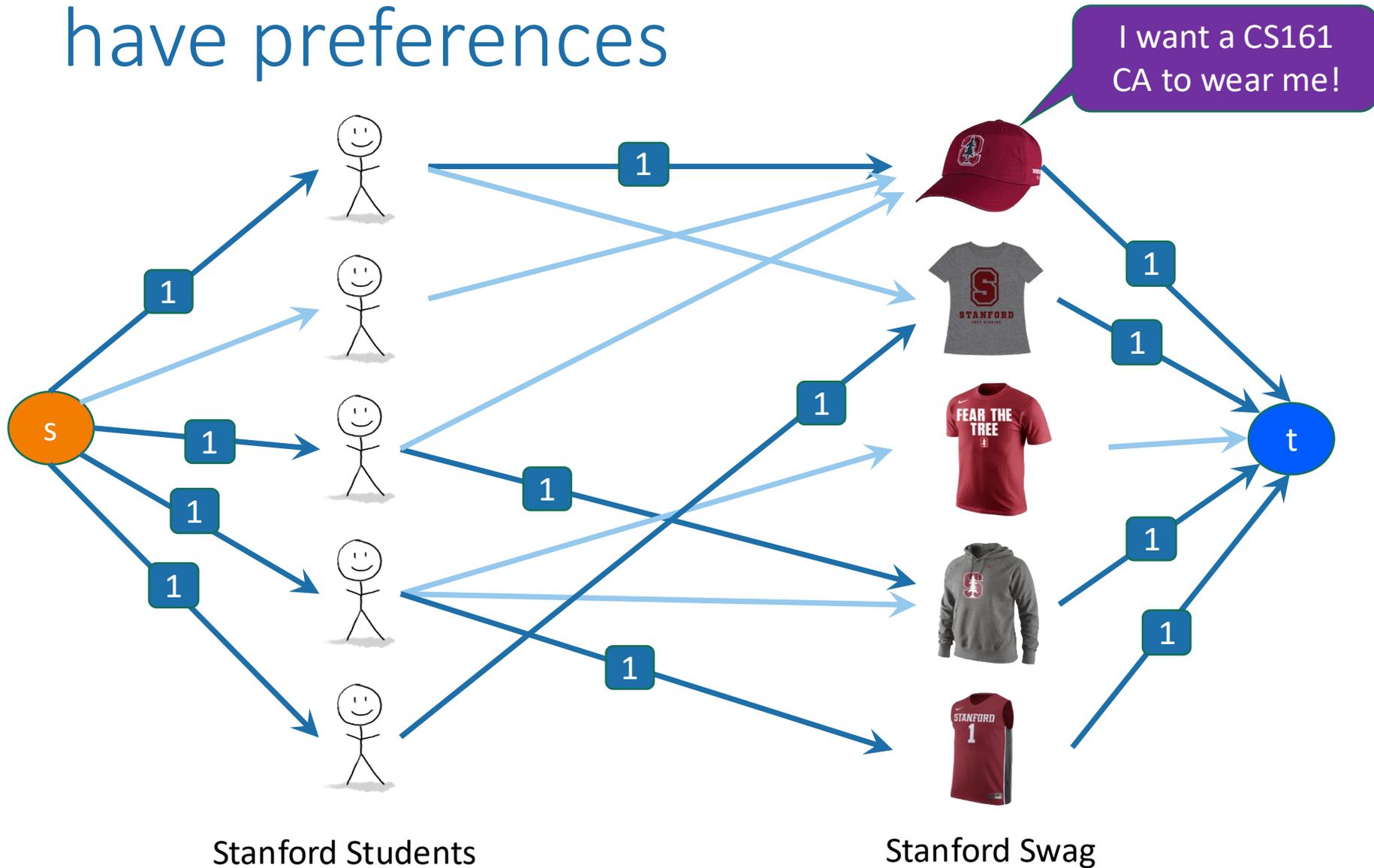
The value of a max flow from s to t is equal to the cost of a min s-t cut.



Recap: used s-t max-flow to solve assignment problems



# Today: matching when both sides have preferences



# Today

- Hospitals/residents problem



- **Stable matchings**

- Solve the hospitals/residents problem
- But can we find them?

- **Deferred Acceptance Algorithm**

- Find stable matchings!

- Discussion, applications and non-applications

# The hospital residency problem

- After completing 4 years of undergrad and 4 years of med school, med students are finally ready to start their internship (“residency”)
- Each **applicant** has a preference over different **residency programs**
- Each **program** has a preference over the **applicants**

How should you match applicants to residencies?

Simplifying assumption today:  
Each program has 1 slot

# The hospital residency problem

- After completing 4 years of undergrad and 4 years of med school, med students are finally ready to start their internship (“residency”)
- Each **doctor** has a preference over different **hospitals**
- Each **hospital** has a preference over the **doctors**

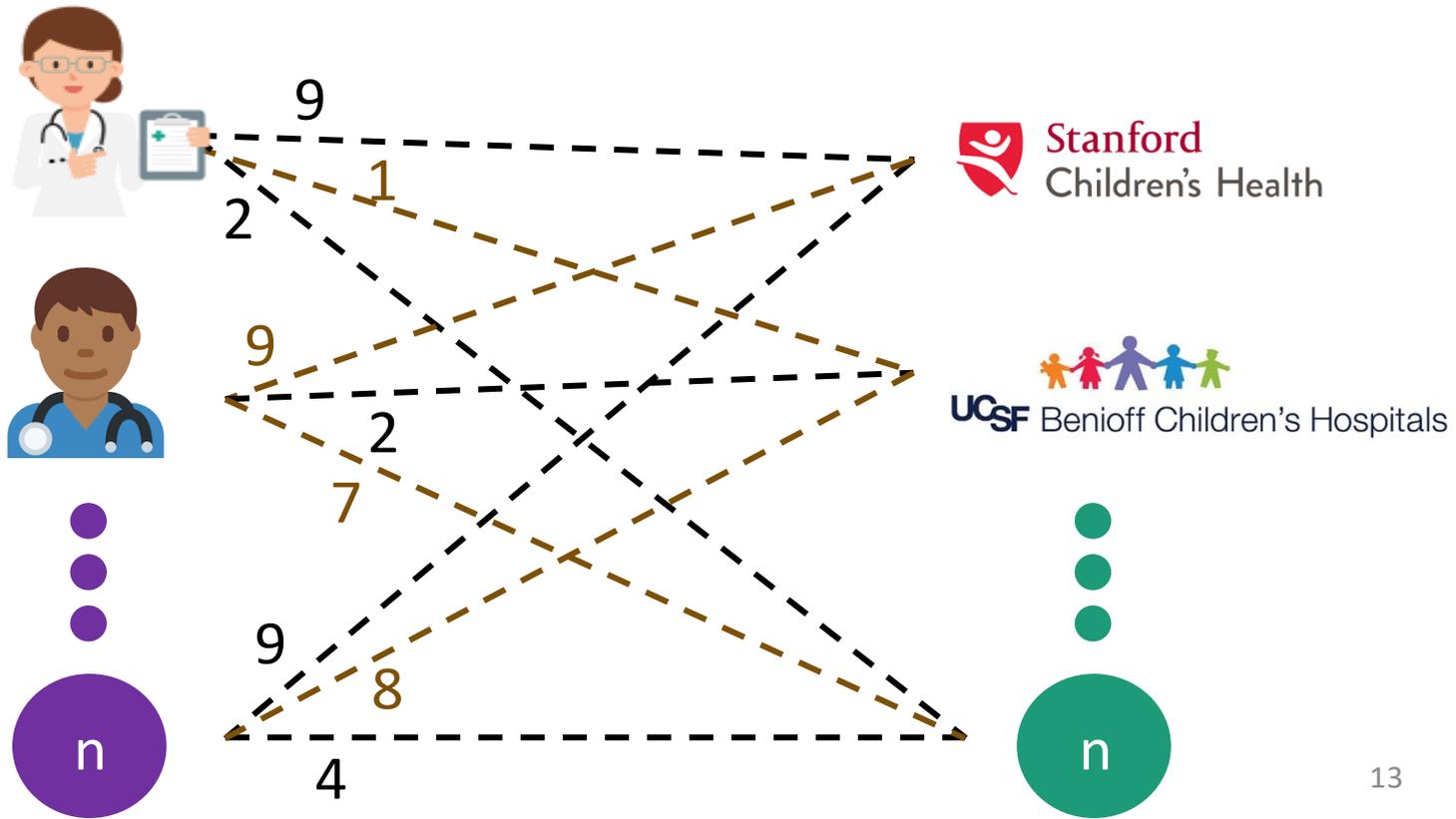
How should you match doctors with hospitals?

Simplifying assumption today:  
Each hospital has 1 slot

# One way to model this problem

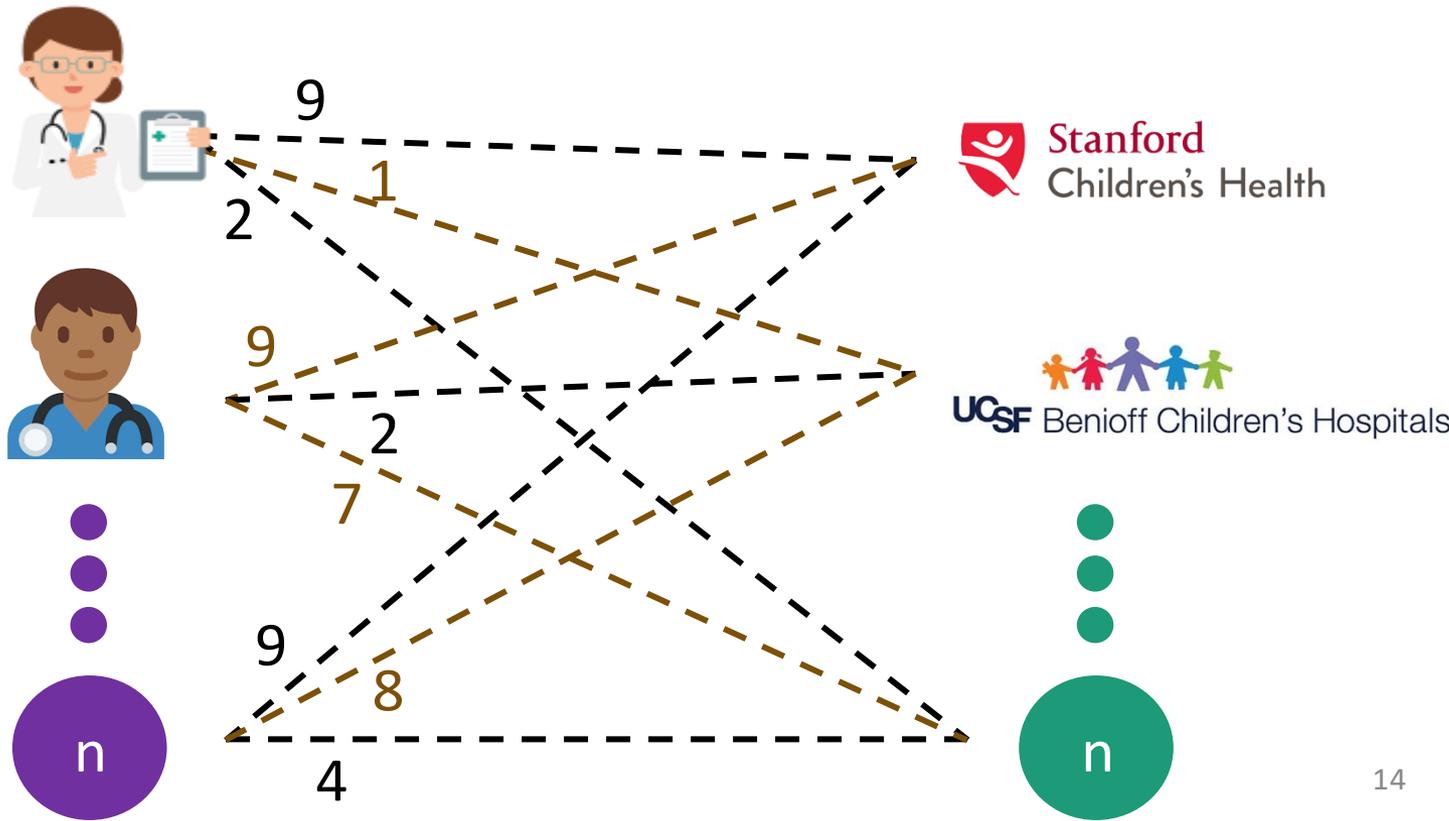
- Each **doctor** has a preference over **hospitals**
- Each **hospital** has a preference over the **doctor**

How should you match doctors with hospitals?



# One way to model this problem

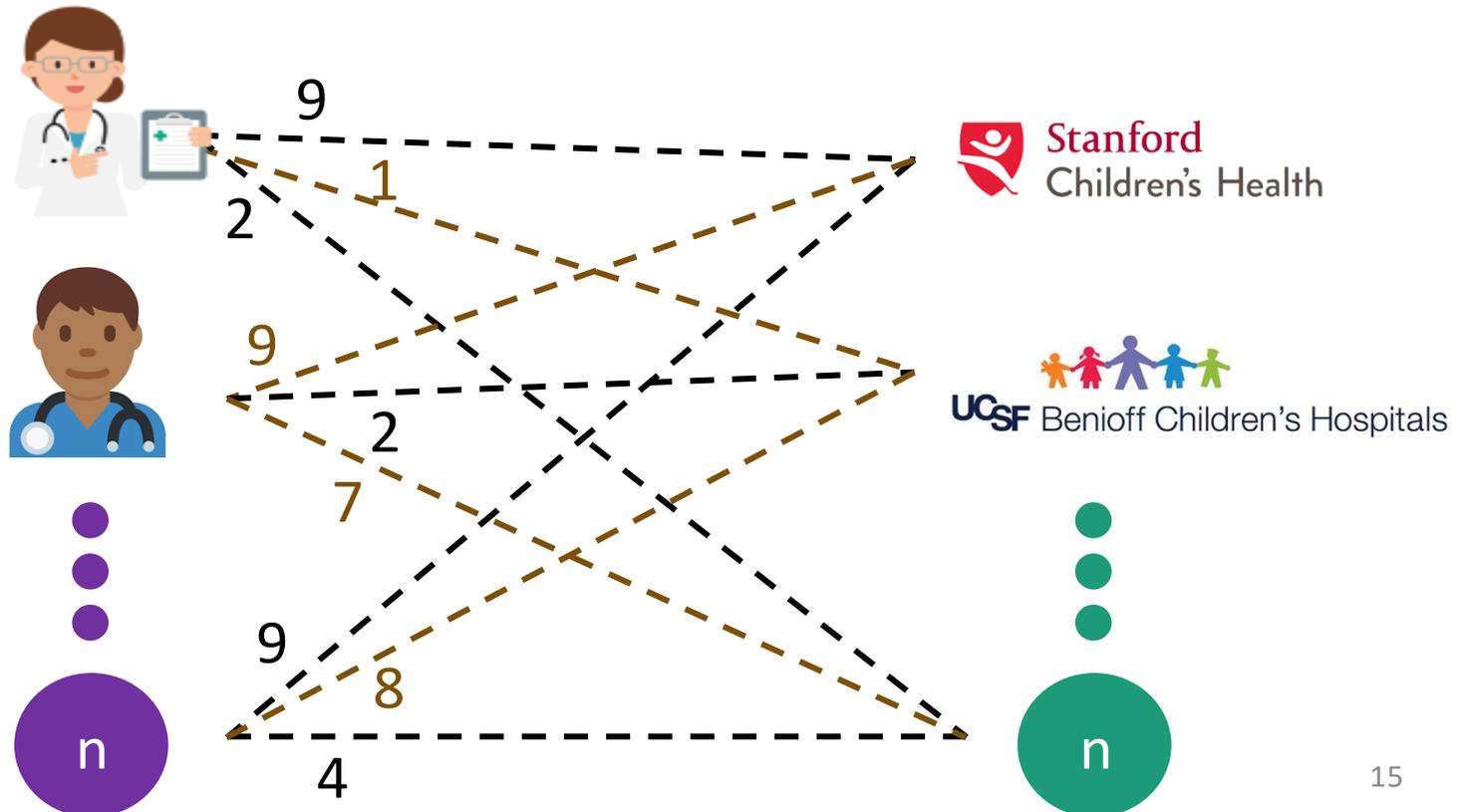
- Bipartite graph between **doctors** and **hospitals**
- Weights on edges = some function of preferences  
(highest weight = most preferred)



# One way to model this problem

- Bipartite graph between **doctors** and **hospitals**
- Weights on edges = some function of preferences

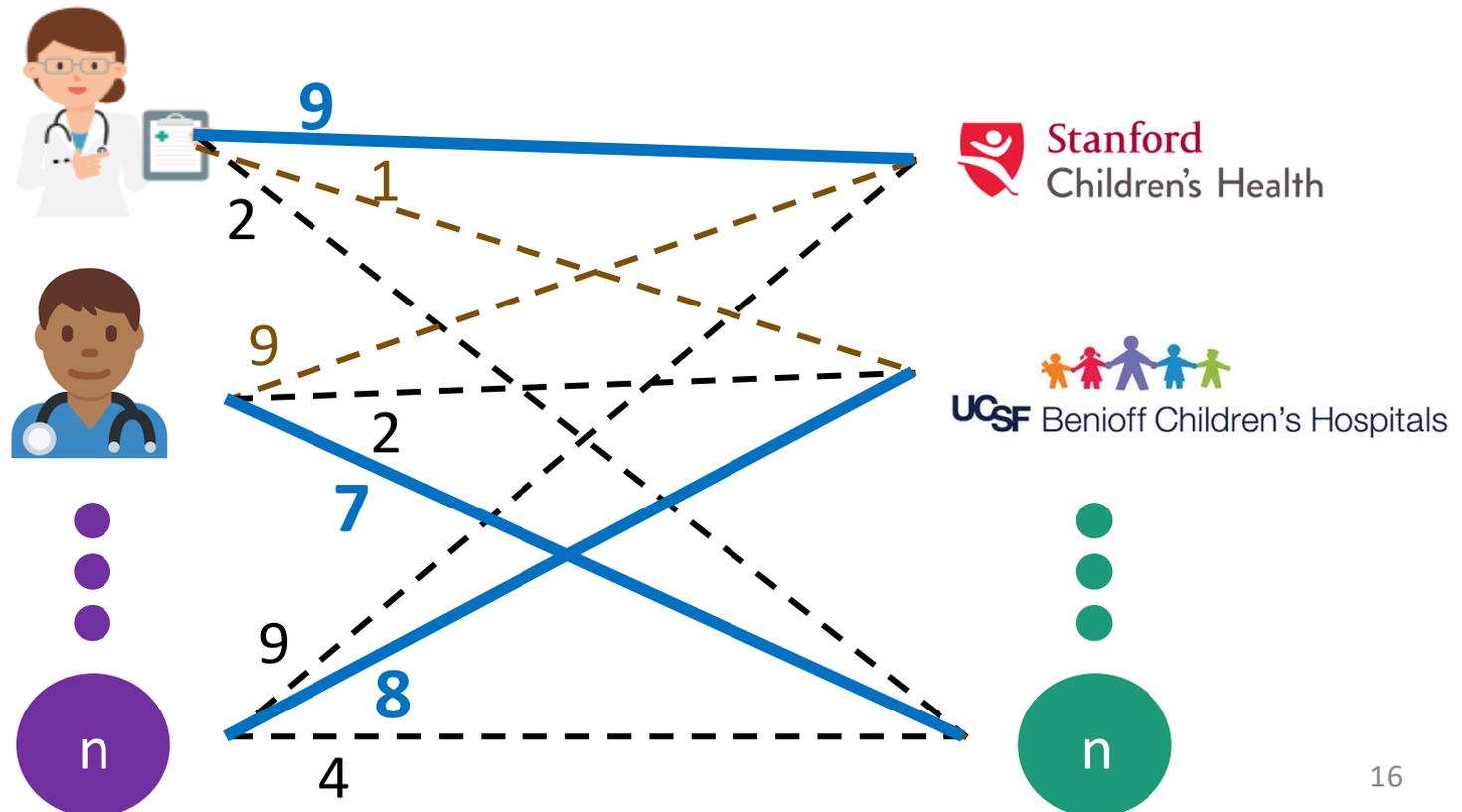
“Hungarian Algorithm” (CS261) finds a max weight matching



# One way to model this problem

- Bipartite graph between **doctors** and **hospitals**
- Weights on edges = some function of preferences

“Hungarian Algorithm” (CS261) finds a max weight matching



“Each hospital/doctor has a list of preferences”

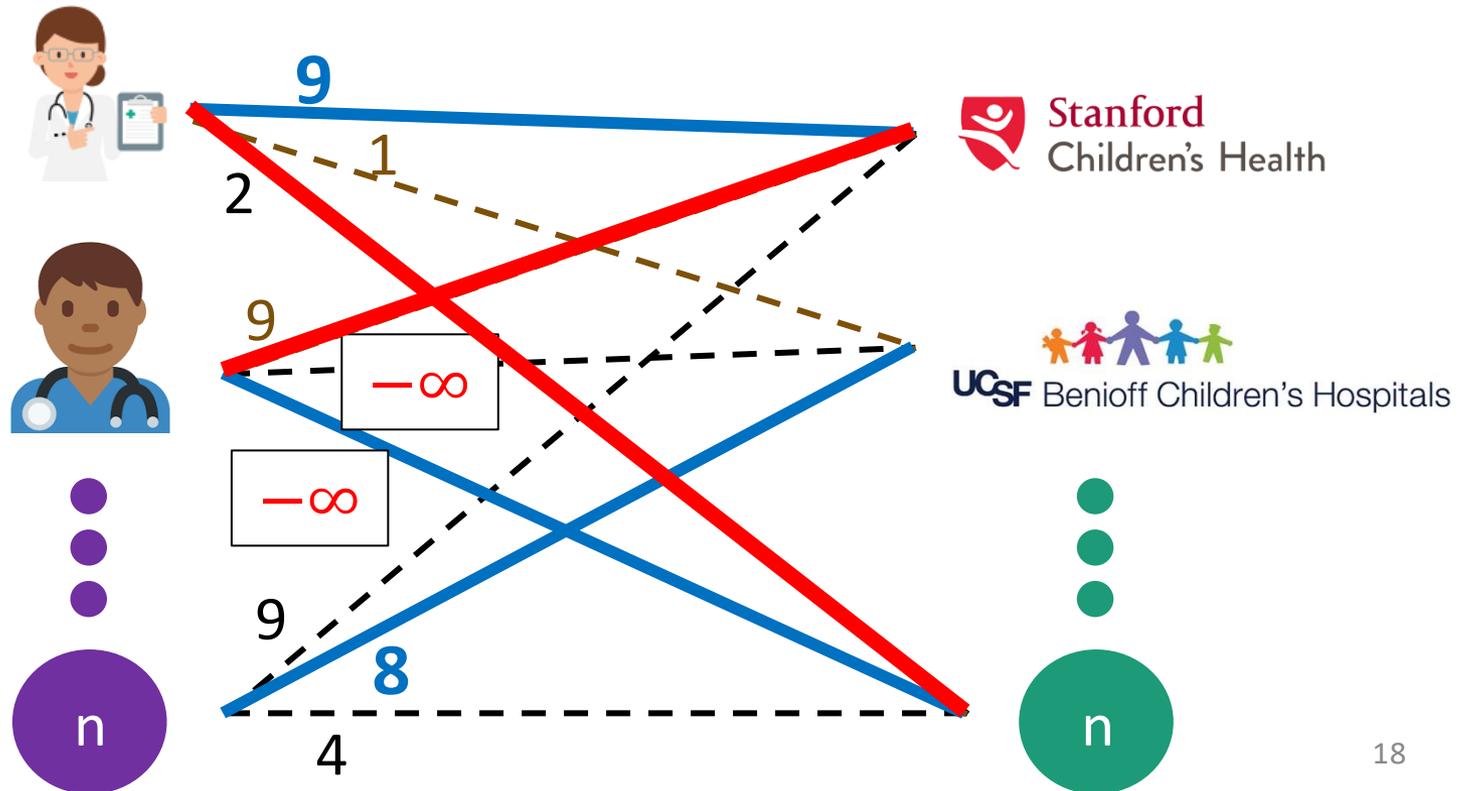
Missing step:

How does the *algorithm* get the preferences?

# Where does your input come from?

... and what can go wrong if we don't think about it carefully:

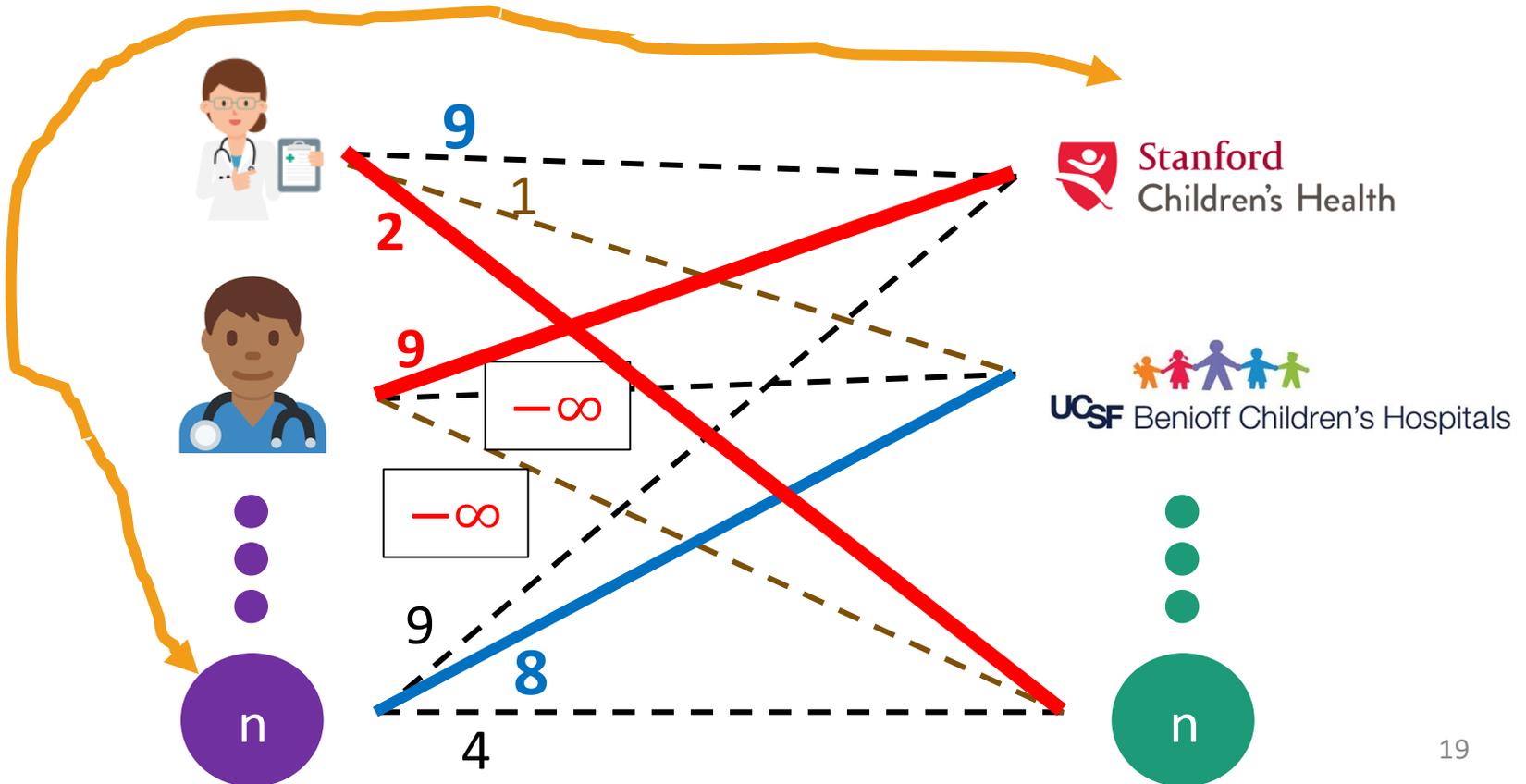
1. Some doctors may misreport their preferences



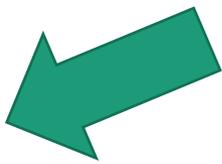
# Where does your input come from?

... and what can go wrong if we don't think about it carefully:

1. Some doctors may misreport their preferences
2. Some doc+hospital may match outside your algorithm

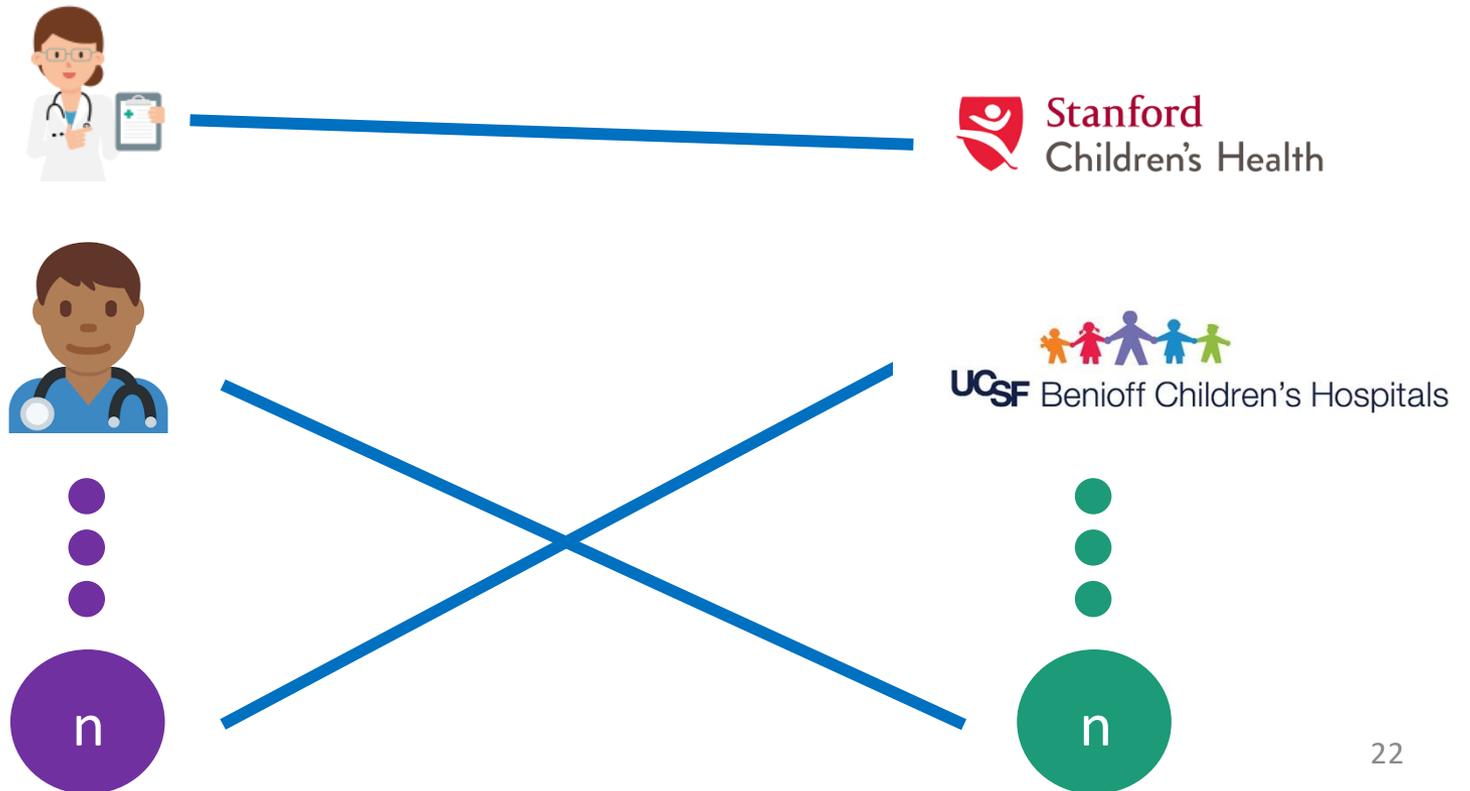


# Today

- Hospitals/residents problem
  - **Stable matchings**
    - Solve the hospitals/residents problem
    - But can we find them?
  - **Deferred Acceptance Algorithm**
    - Find stable matchings!
  - Discussion, applications and non-applications
- 

# *Stable Matching*

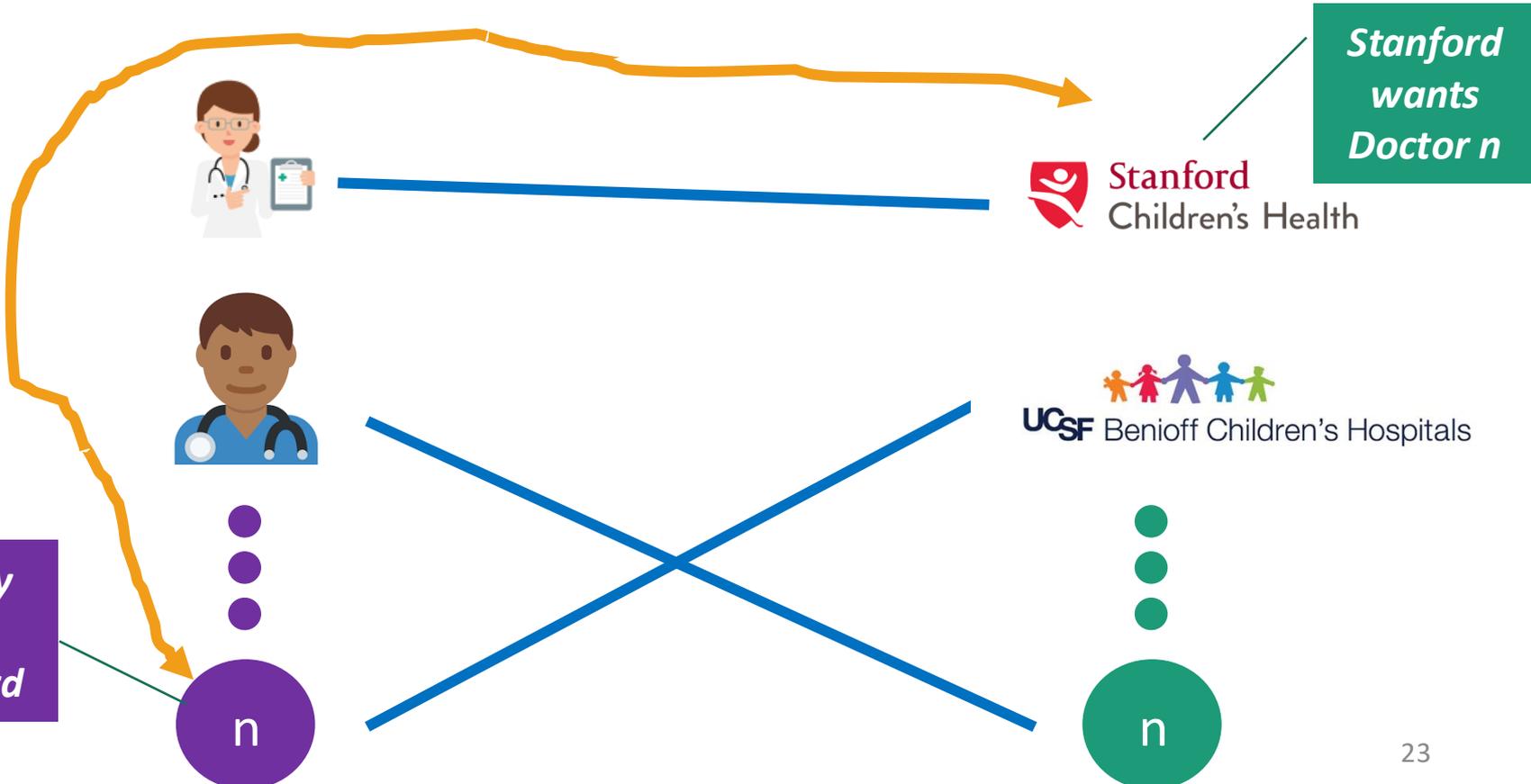
# Stable Matching



# Stable Matching

## Def (blocking pair):

Given Matching  $M$ , (Doctor  $i$ , Hospital  $j$ ) are a **blocking pair** if they prefer each other to their assignment in  $M$



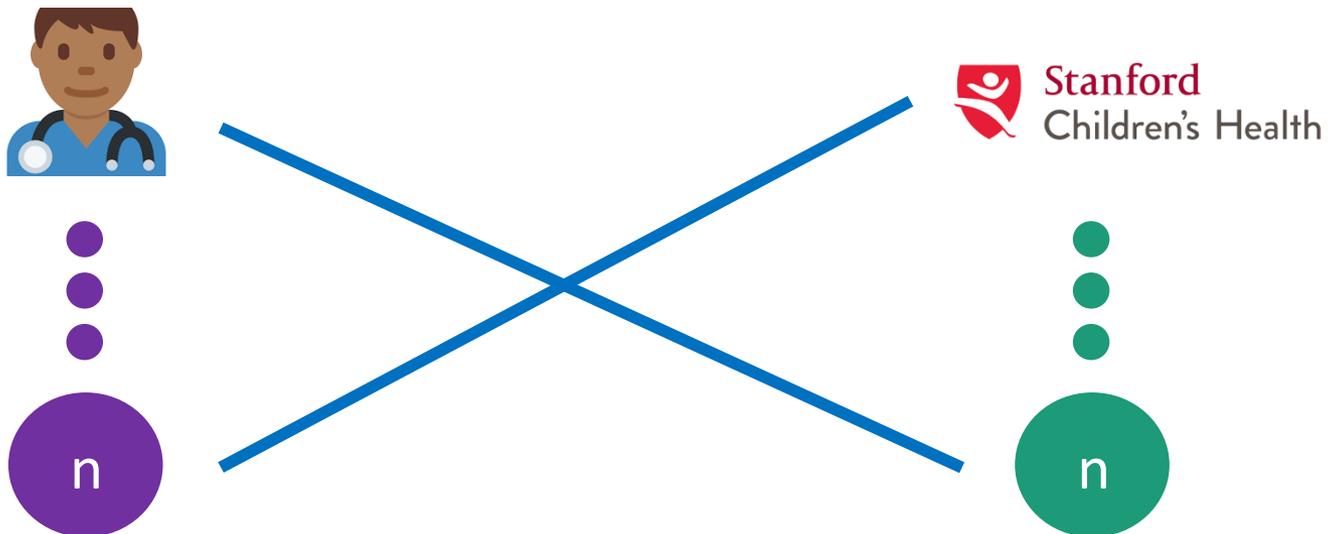
# Stable Matching

## Def (blocking pair):

Given Matching  $M$ , (Doctor  $i$ , Hospital  $j$ ) are a **blocking pair** if they prefer each other to their assignment in  $M$

## Def (stable matching):

$M$  is a **stable matching** if there are no **blocking pairs**.



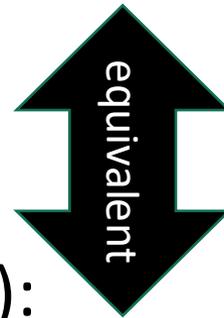
# Stable Matching

## Def (blocking pair):

Given Matching  $M$ , (Doctor  $i$ , Hospital  $j$ ) are a **blocking pair** if they prefer each other to their assignment in  $M$

## Def (stable matching):

$M$  is a **stable matching** if there are no **blocking pairs**.



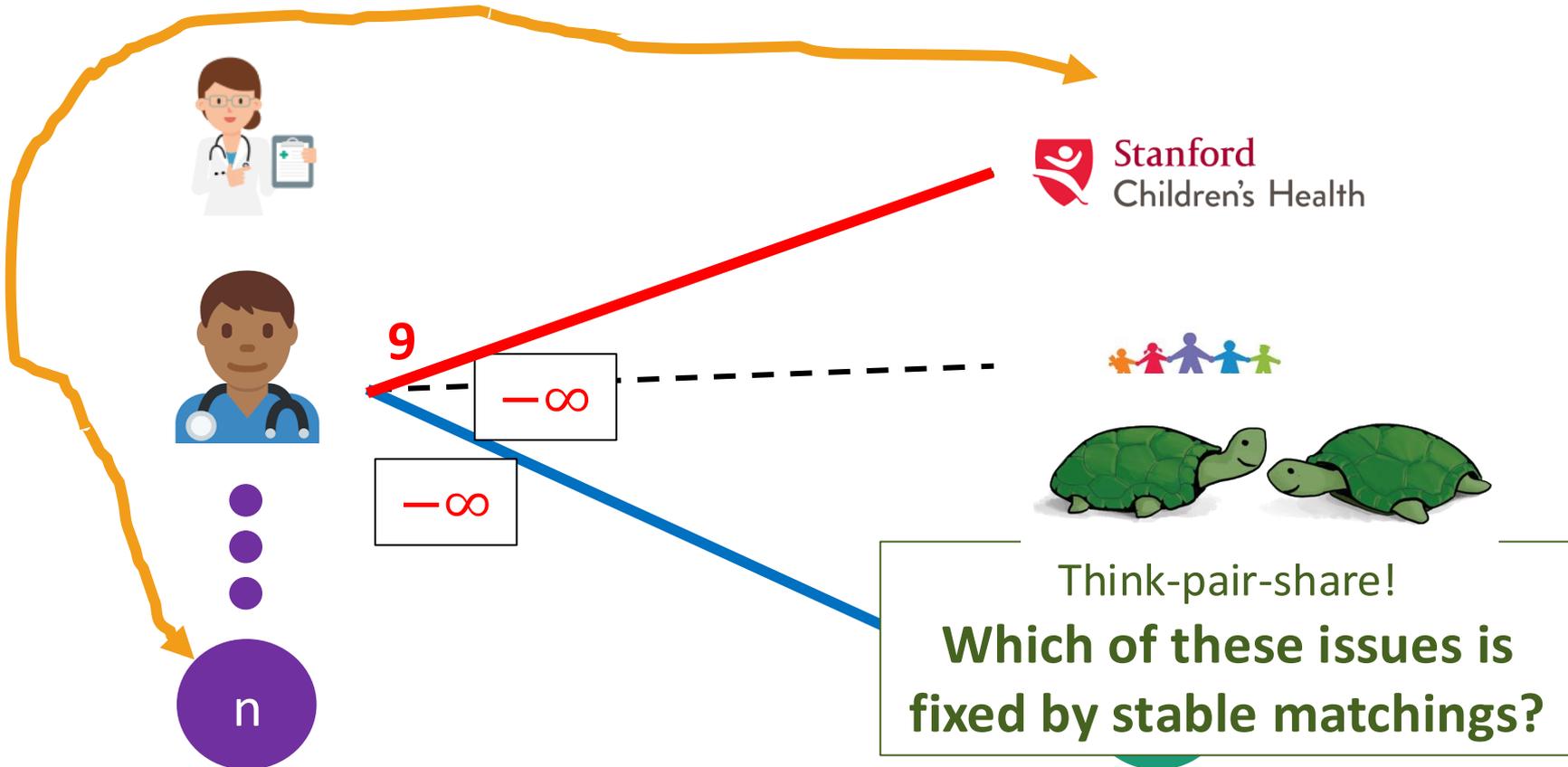
For every unmatched pair  $(i,j)$ :

- Doctor  $i$  prefers Hospital  $M(i)$  over Hospital  $j$ , or;
- Hospital  $j$  prefers Doctor  $M(j)$  over Doctor  $i$

# Unstable Matching and incentives

Problems we identified with unstable matchings:

1. Some doctors may misreport their preferences
2. Some doc+hospital may match outside your algorithm

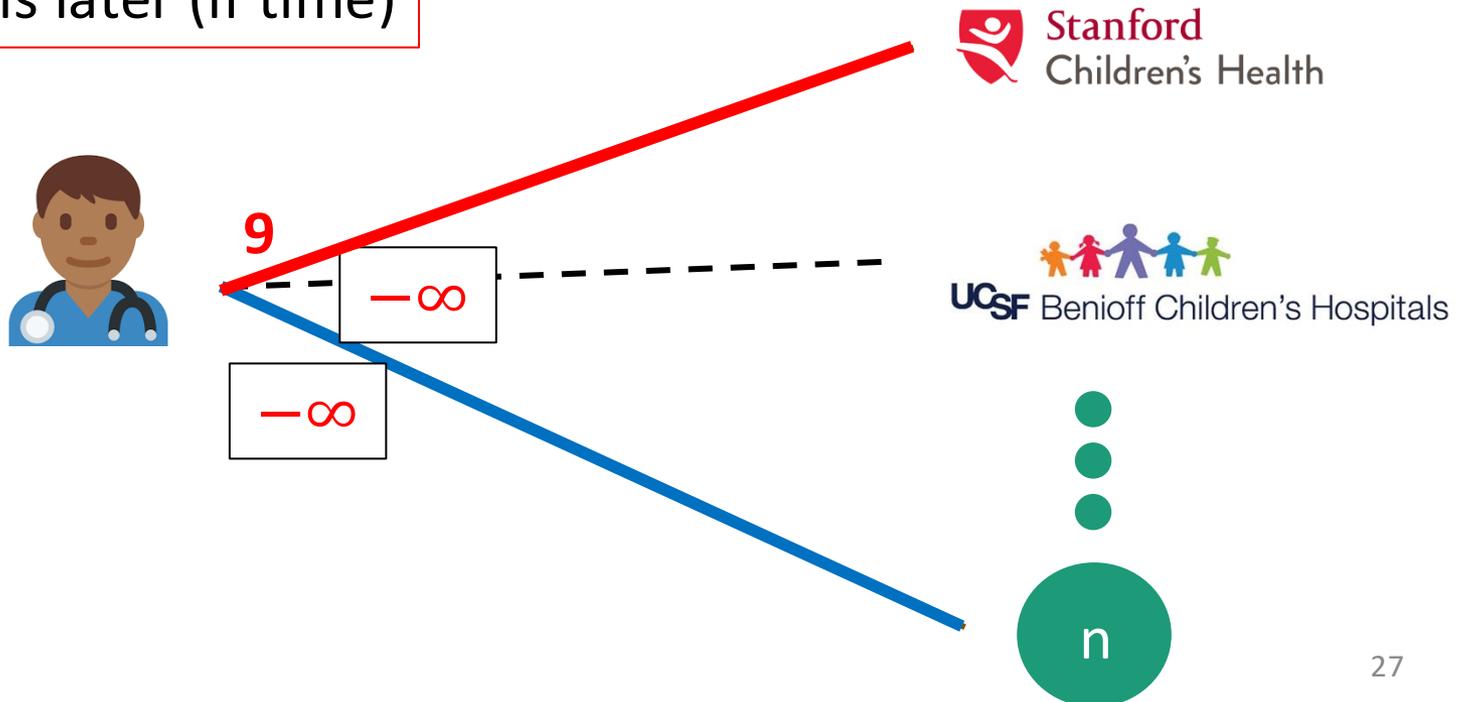


# Stable Matching and incentives

With **stable matching**:

1. Will doctors misreport their preferences?

Not obvious!  
We'll come back  
to this later (if time)



# Stable Matching and incentives

With **stable matching**:

- Doctor+hospital **never** prefer to match outside algorithm!



This is the point of stable matching:  
Only a blocking pair would prefer to match outside.  
**Stable matching = no blocking pairs!**



n

# *Stable Matching Problem*

How to find stable matchings!

# Stable Matching Problem

## Stable Matching Problem

**Input:** each doctor/hospital submits a ranking (permutation) of  $\{1, \dots, n\}$

**Output:** a **stable matching**

Alice's preferences	
1 <sup>st</sup>	Stanford
2 <sup>nd</sup>	...
...	...
n <sup>th</sup>	UCSF

Stanford's preferences	
1 <sup>st</sup>	Alice
2 <sup>nd</sup>	...
...	...
n <sup>th</sup>	Bob

### Def (blocking pair):

Given Matching  $M$ , (Doctor  $i$ , Hospital  $j$ ) are a **blocking pair** if they prefer each other to their assignment in  $M$

### Def (stable matching):

$M$  is a **stable matching** if there are no **blocking pairs**.

# Naïve attempt #1

Greedy algorithm:

Step 1- match all the pairs  $(i,j)$  such that  
 $j$  is  $i$ 's top choice, and  $i$  is  $j$ 's top choice

Step 2- hopefully recurse on the rest somehow...

- Observation: Step 1 never rules out any solution 😊

**Stable Matching Problem**

**Input:**  $2n$  rankings of  $\{1, \dots, n\}$

**Output:** a **stable matching**

# A slightly more ambitious attempt

Greedy attempt #2:

Step 1- try to match every doctor to her favorite hospital

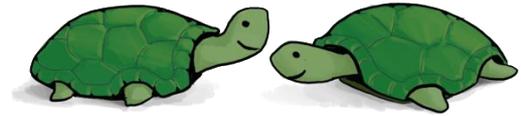
- Break ties by hospital preference

Step 2- hopefully recurse on the rest somehow...

**Stable Matching Problem**

**Input:**  $2n$  rankings of  $\{1, \dots, n\}$

**Output:** a **stable matching**



# A slightly more an... ..

Think-pair-share!

Matching  $(C,y)$  was a bad idea...  
How could we avoid it?

Greedy attempt #2:

Step 1- try to match every doctor to her favorite hospital

- Break ties by hospital preference

We're already wrong!

- Step 1:  $A,B$  want  $x$ ,  $C$  wants  $y$  so we **match**  $(A,x)$  and  $(C,y)$
- But now  $(B,y)$  is **blocking**!

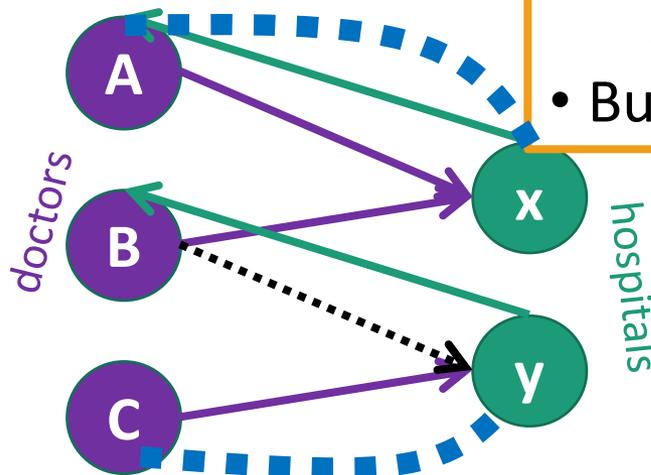
Doctor's #1 choice



Doctor's #2 choice



Hospital's #1 choice



**Stable Matching Problem**

**Input:**  $2n$  rankings of  $\{1, \dots, n\}$

**Output:** a **stable matching**

# Today

- Hospitals/residents problem
- **Stable matchings**
  - Solve the hospitals/residents problem
  - But can we find them?
- **Deferred Acceptance Algorithm** 
  - Find stable matchings!
- Discussion, applications and non-applications

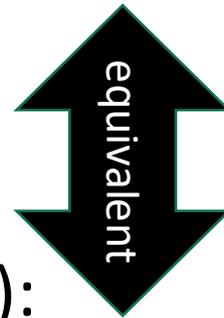
# Questions?

## Def (blocking pair):

Given Matching  $M$ , (Doctor  $i$ , Hospital  $j$ ) are a **blocking pair** if they prefer each other to their assignment in  $M$

## Def (stable matching):

$M$  is a **stable matching** if there are no **blocking pairs**.



For every unmatched pair  $(i,j)$ :

- Doctor  $i$  prefers Hospital  $M(i)$  over Hospital  $j$ , or;
- Hospital  $j$  prefers Doctor  $M(j)$  over Doctor  $i$

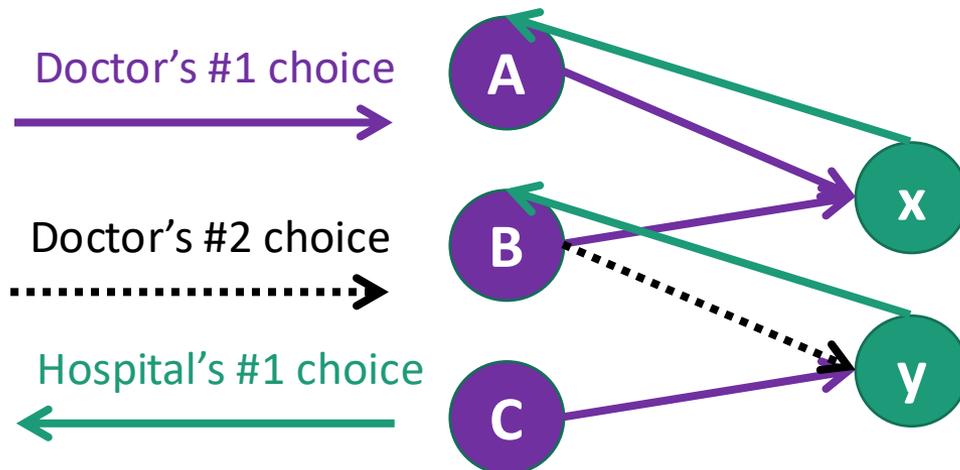
# Deferred Acceptance Algorithm

[Gale Shapley '62] -> 2012 Nobel Prize\* in Econ!

\*- Joint w/ Al Roth from Stanford

# Deferred Acceptance Algorithm

Main idea: **try** to match each doctor to top choice;  
if you discover a **blocking pair**, just switch the matching!



The issue was:

A, B want x, C want y

we tried to match (A, x) and (C, y)  
but then (B, y) was **blocking!**

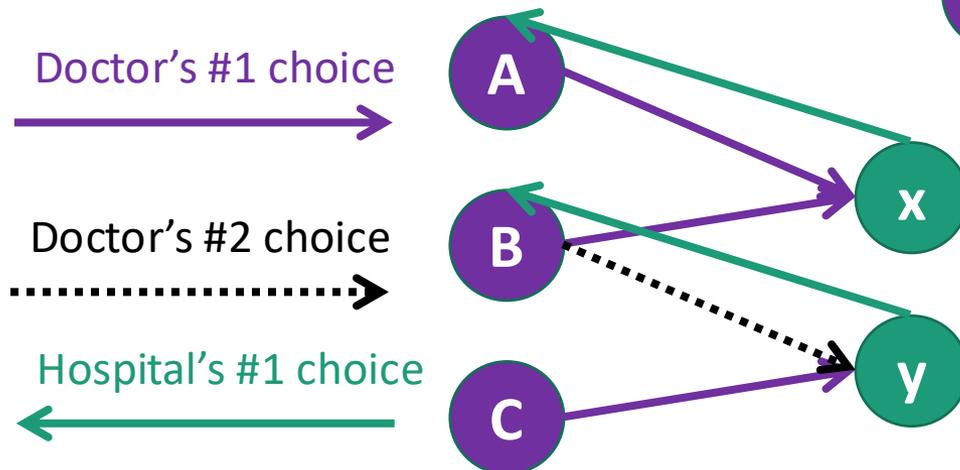
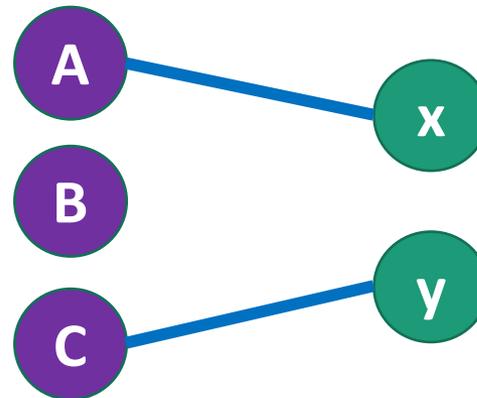
# Deferred Acceptance Algorithm

Main idea: **try** to match each doctor to top choice;  
if you discover a **blocking pair**, just switch the matching!

Algorithm iteration 1:

A, B want x; C wants y

So we match (A,x) and (C,y)



The issue was:

A, B want x, C want y

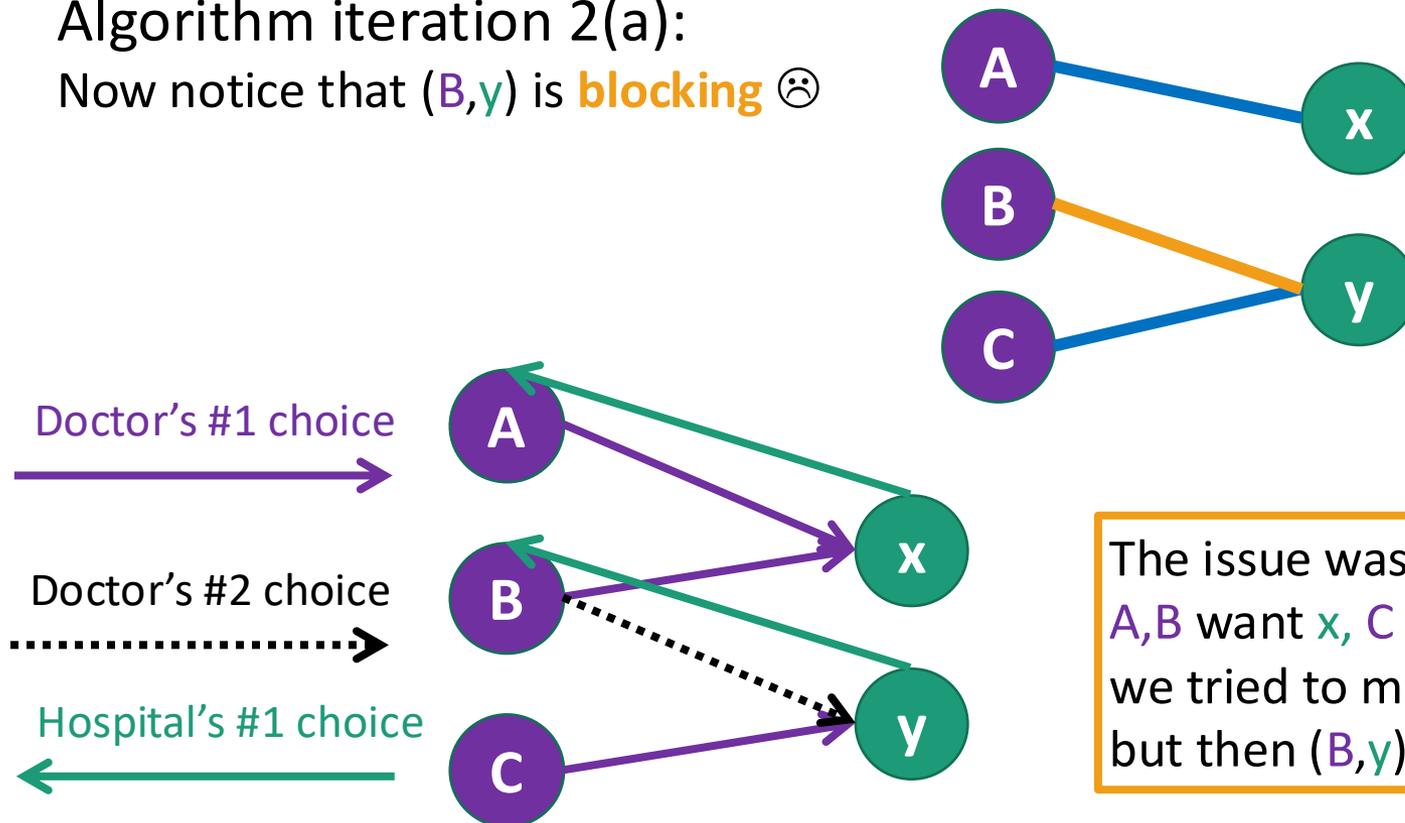
we tried to match (A,x) and (C,y)  
but then (B,y) was **blocking**!

# Deferred Acceptance Algorithm

Main idea: *try* to match each doctor to top choice;  
if you discover a **blocking pair**, just switch the matching!

Algorithm iteration 2(a):

Now notice that  $(B,y)$  is **blocking** ☹️



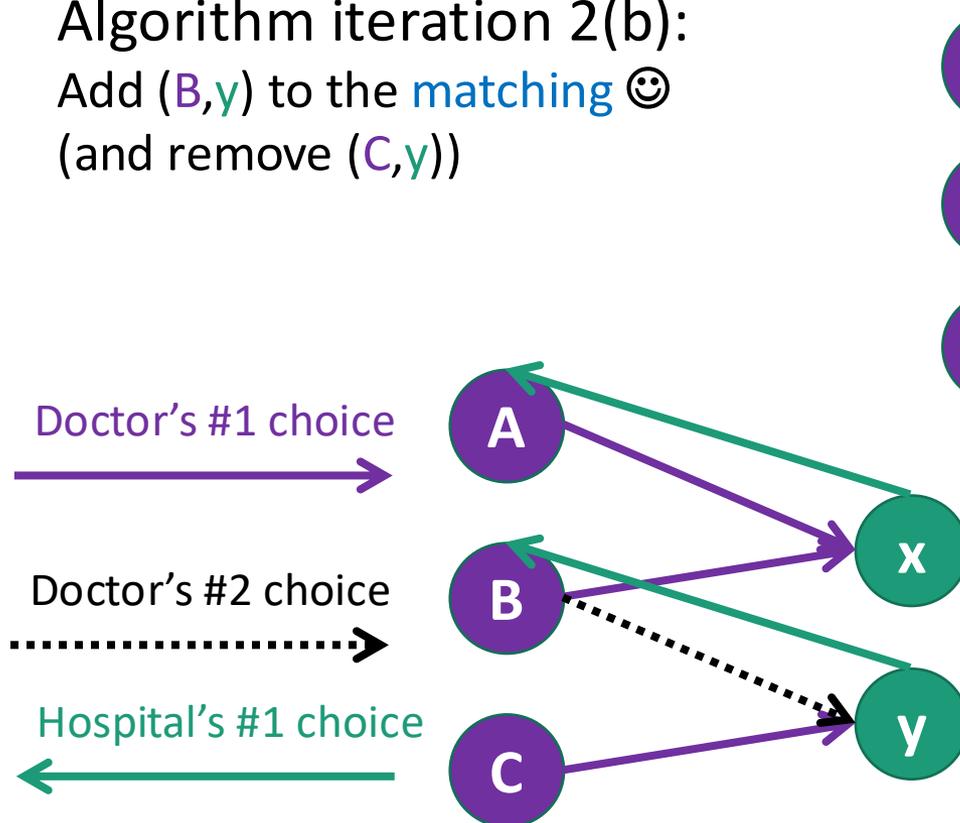
The issue was:

A,B want x, C want y  
we tried to match  $(A,x)$  and  $(C,y)$   
but then  $(B,y)$  was **blocking**!

# Deferred Acceptance Algorithm

Main idea: *try* to match each doctor to top choice;  
if you discover a **blocking pair**, just switch the matching!

Algorithm iteration 2(b):  
Add (B,y) to the matching 😊  
(and remove (C,y))



The issue was:  
A,B want x, C want y  
we tried to match (A,x) and (C,y)  
but then (B,y) was **blocking**!

# Deferred Acceptance Algorithm

Main idea: **try** to match each doctor to top choice;  
if you discover a **blocking pair**, just switch the matching!

Almost-pseudo-code:

**While** there is an unmatched doctor **i**:

    Try to match **i** to next-favorite **hospital** in her list;

**If** this **hospital** doesn't have a doctor yet:

    Both Doctor **i** and **hospital** are happy with this new match 😊

**Else-if** this **hospital** prefers its current match **i'** over **i**:

    Doctor **i** remains unmatched

**Else-if** this **hospital** prefers **i** over **i'**:

    Unmatch **i'**; Match (**i**, **hospital**)

# Example run-through

# DA Example Run 1



Alice

X, Y, Z



Bob

Y, X, Z



Charlie

Y, Z, X

B, A, C



X

A, B, C



Y

B, C, A



Z

# DA Example Run 1



Alice



X, Y, Z



Bob

Y, X, Z



Charlie

Y, Z, X



B, A, C



X

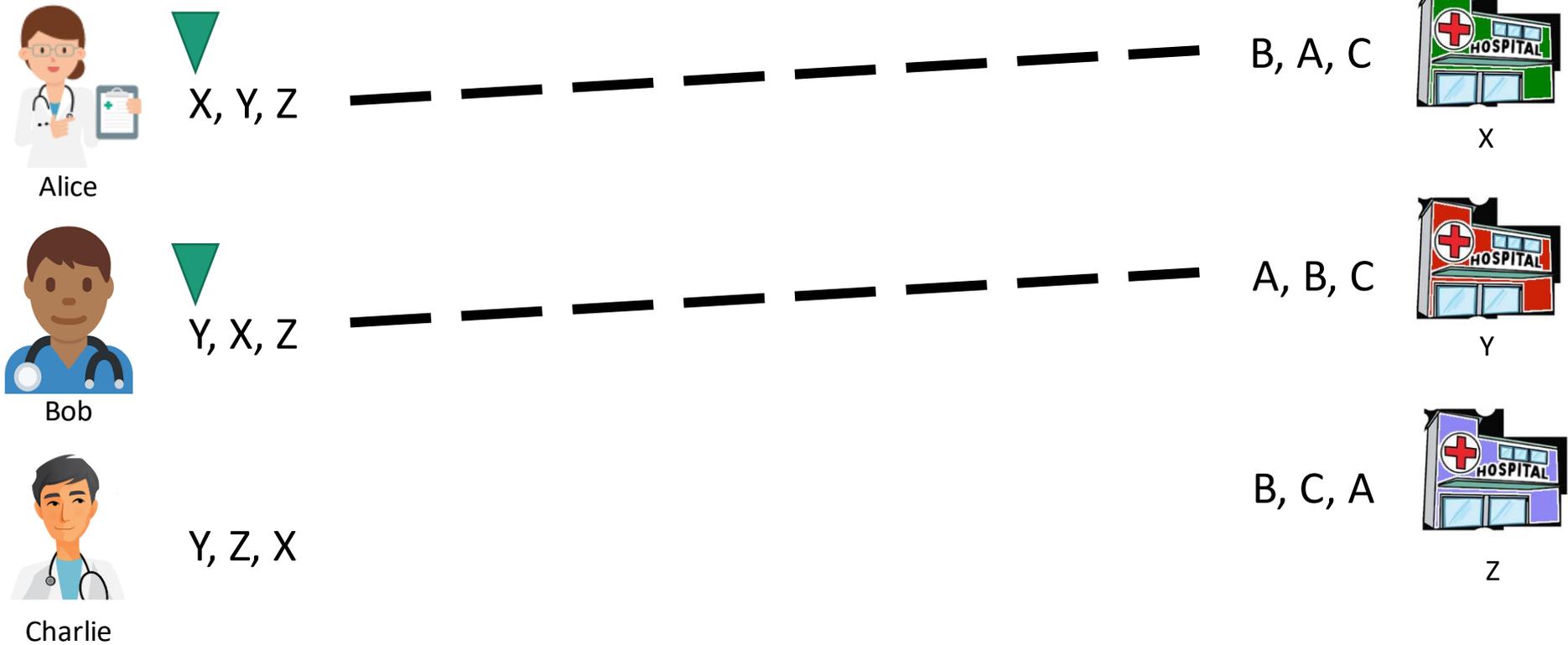


Y

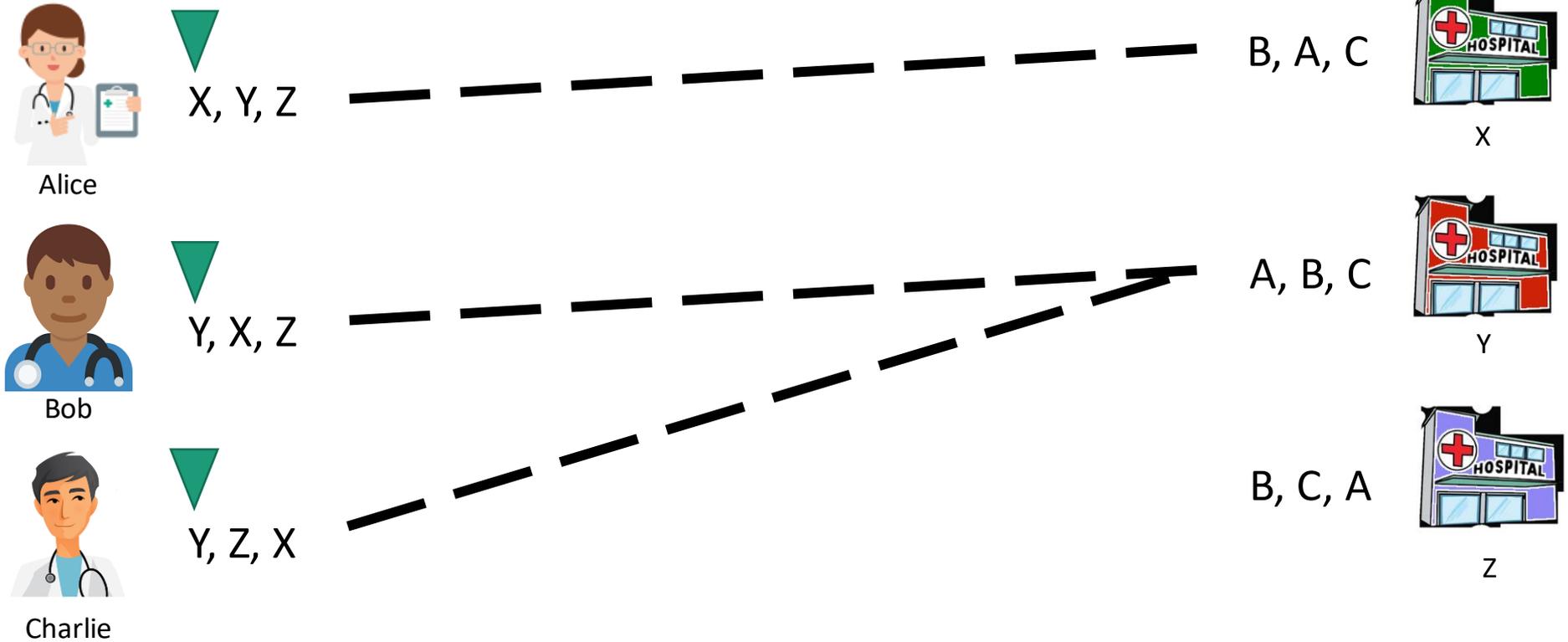


Z

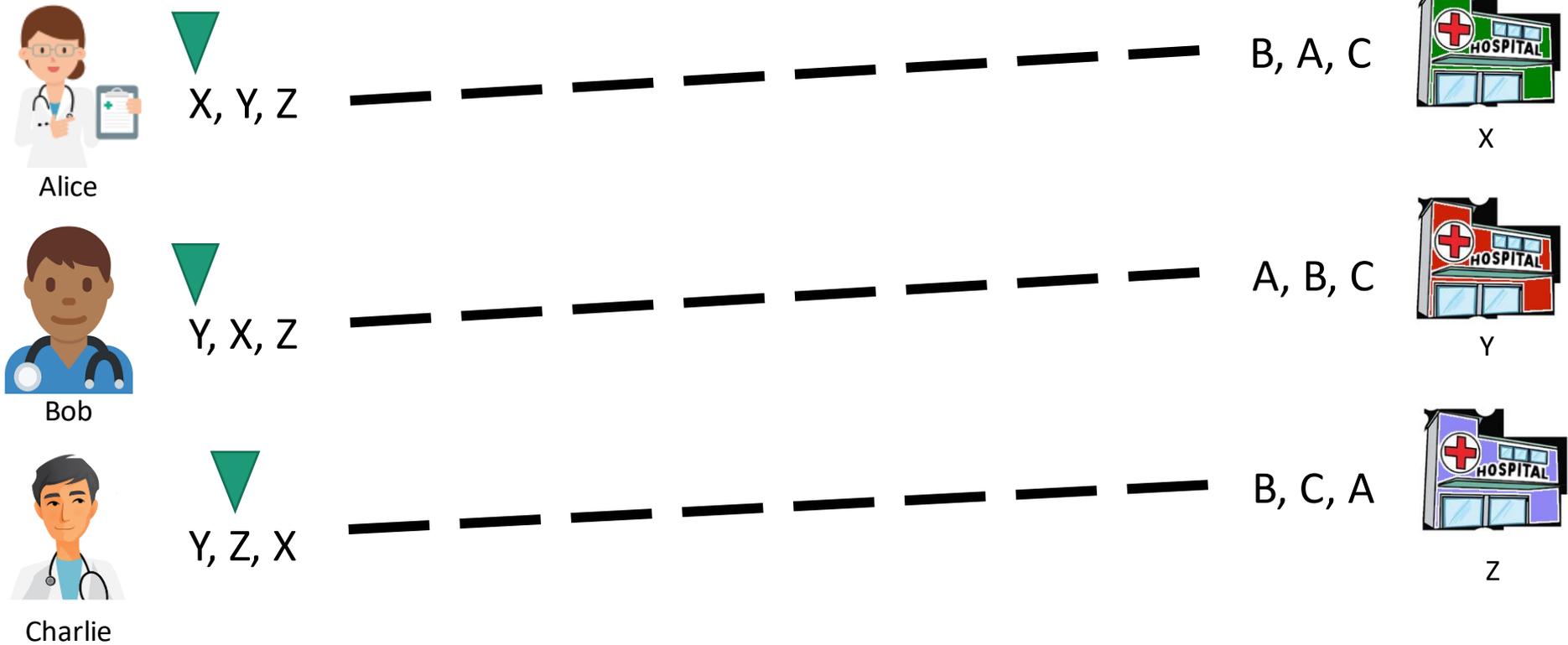
# DA Example Run 1



# DA Example Run 1



# DA Example Run 1



# DA Example Run 1



Alice



X, Y, Z



Bob



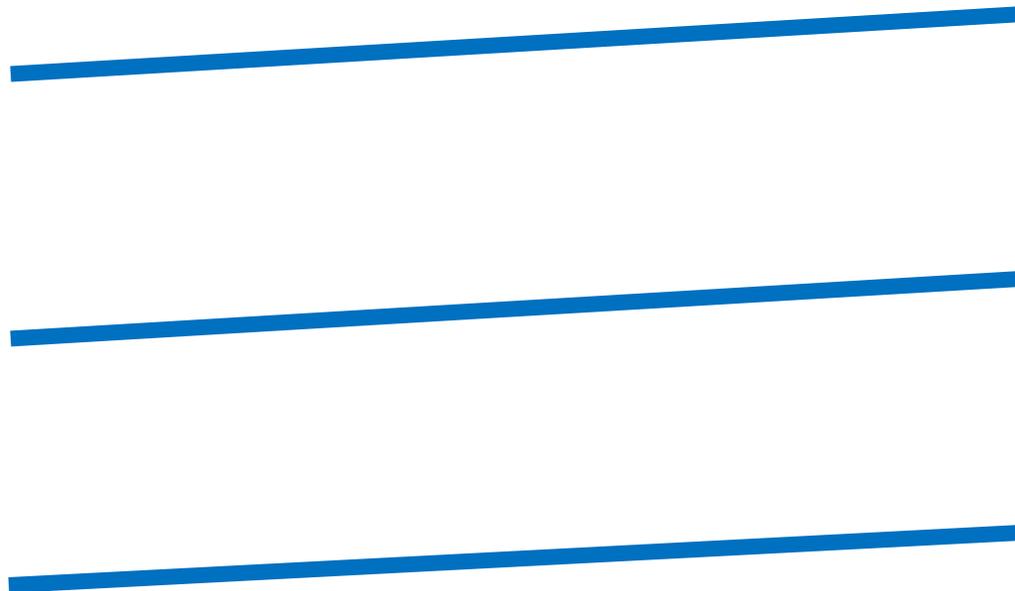
Y, X, Z



Charlie



Y, Z, X



B, A, C



X

A, B, C



Y

B, C, A



Z

Another example

# DA Example Run 2



Alice

X, Y, Z



Bob

Y, X, Z



Charlie

Y, Z, X

B, A, C



X

A, C, B



Y

B, C, A



Z

# DA Example Run 2



Alice



X, Y, Z



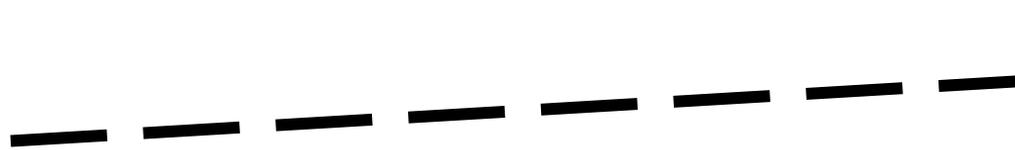
Bob

Y, X, Z



Charlie

Y, Z, X



B, A, C



X

A, C, B



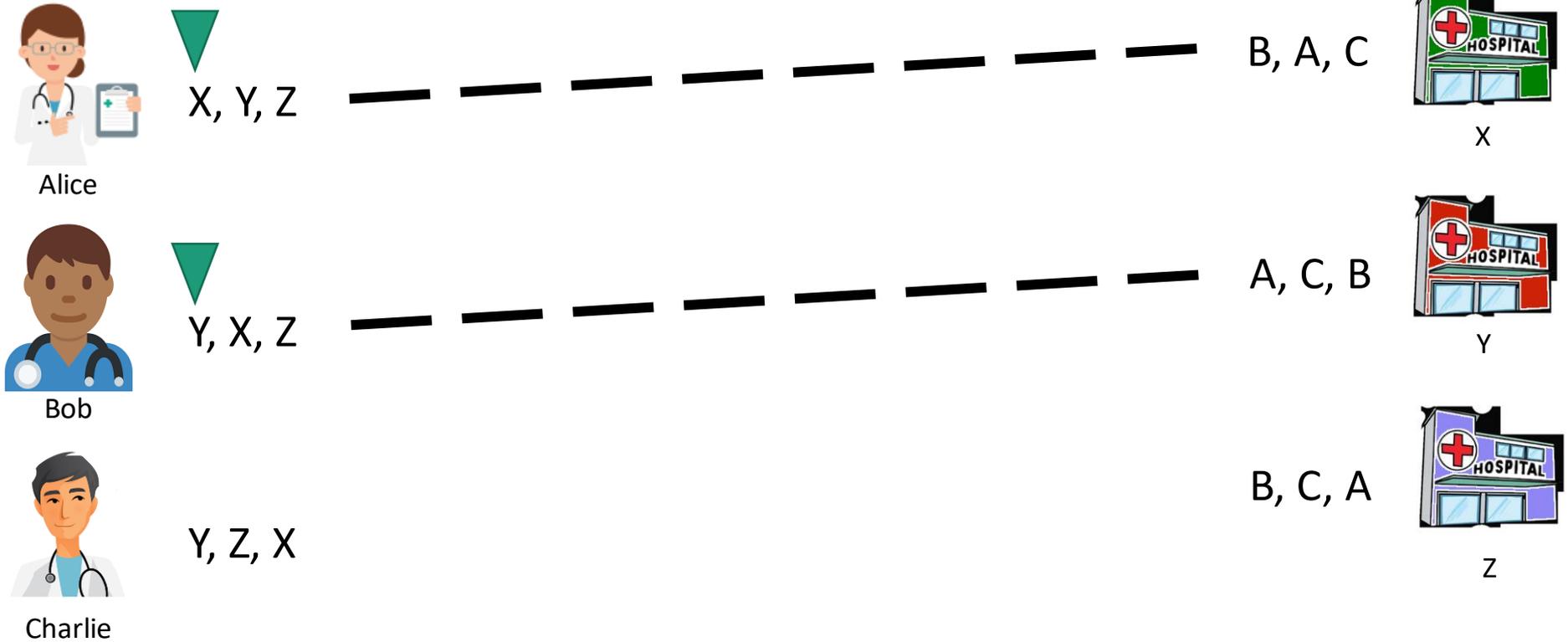
Y

B, C, A

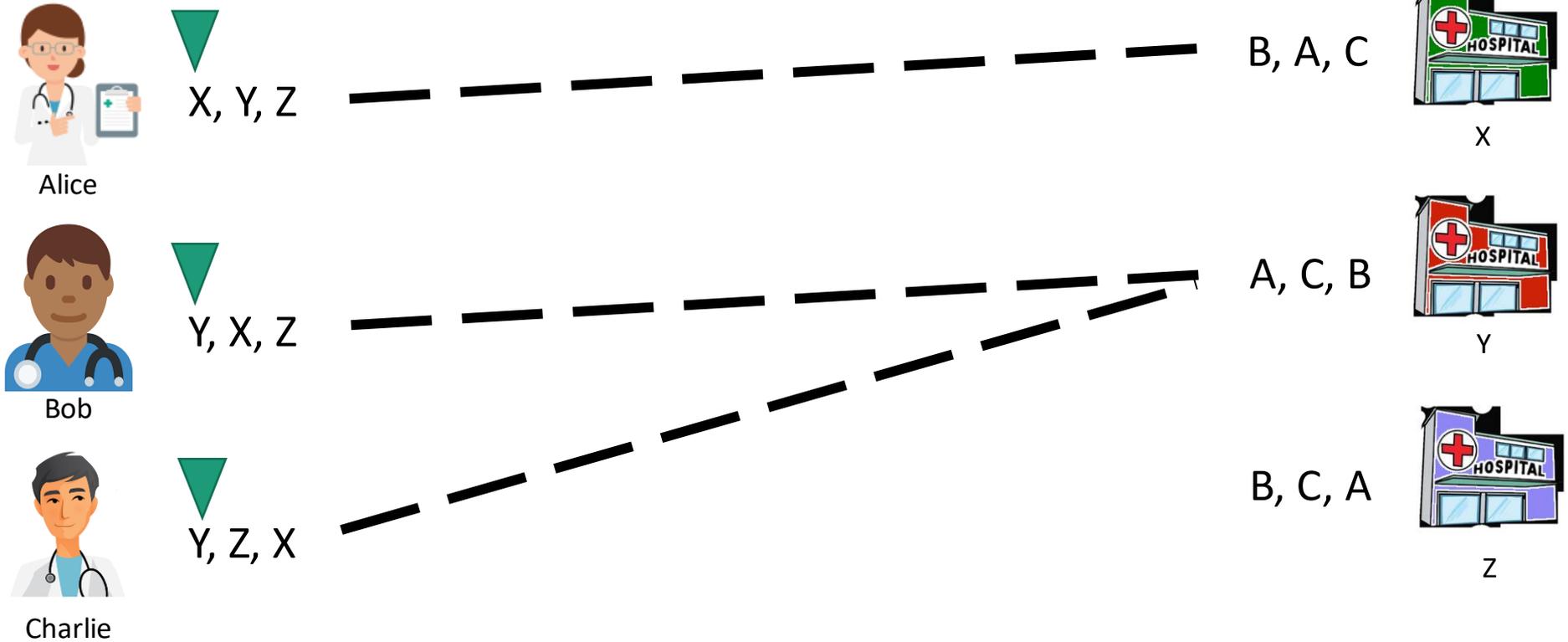


Z

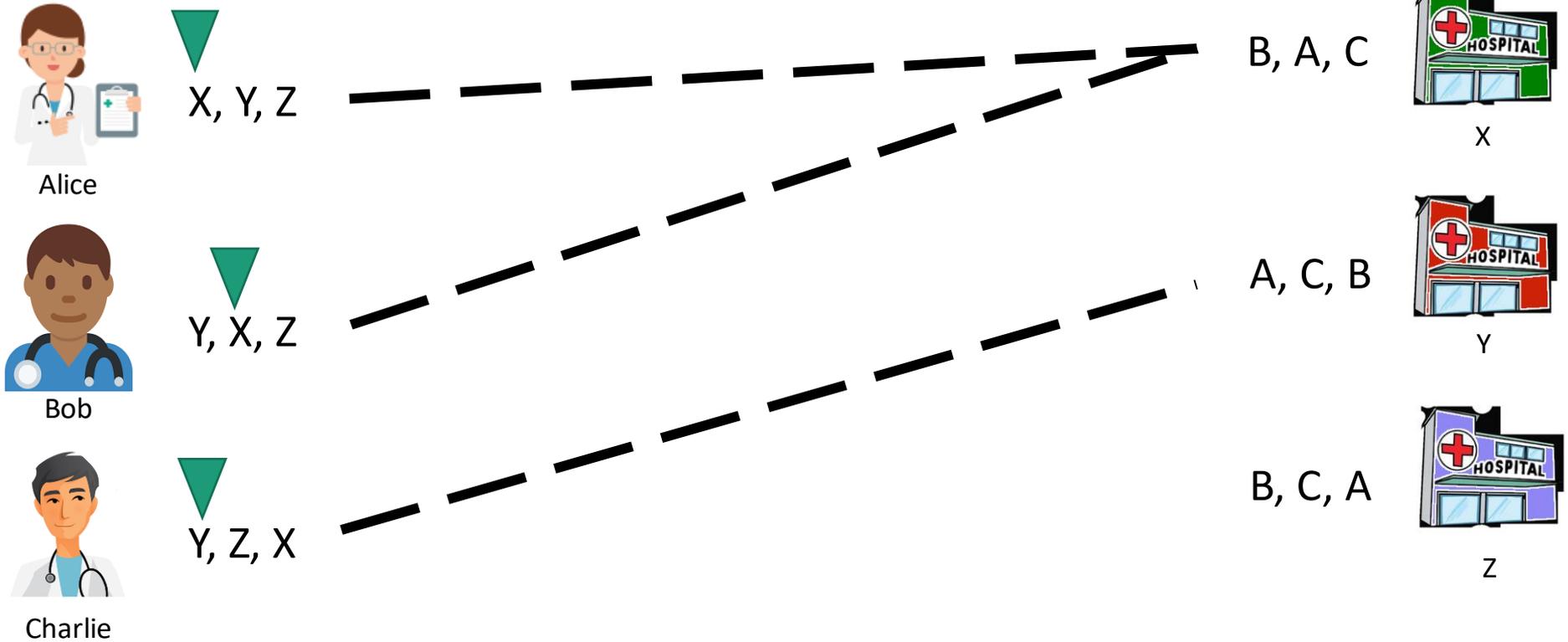
# DA Example Run 2



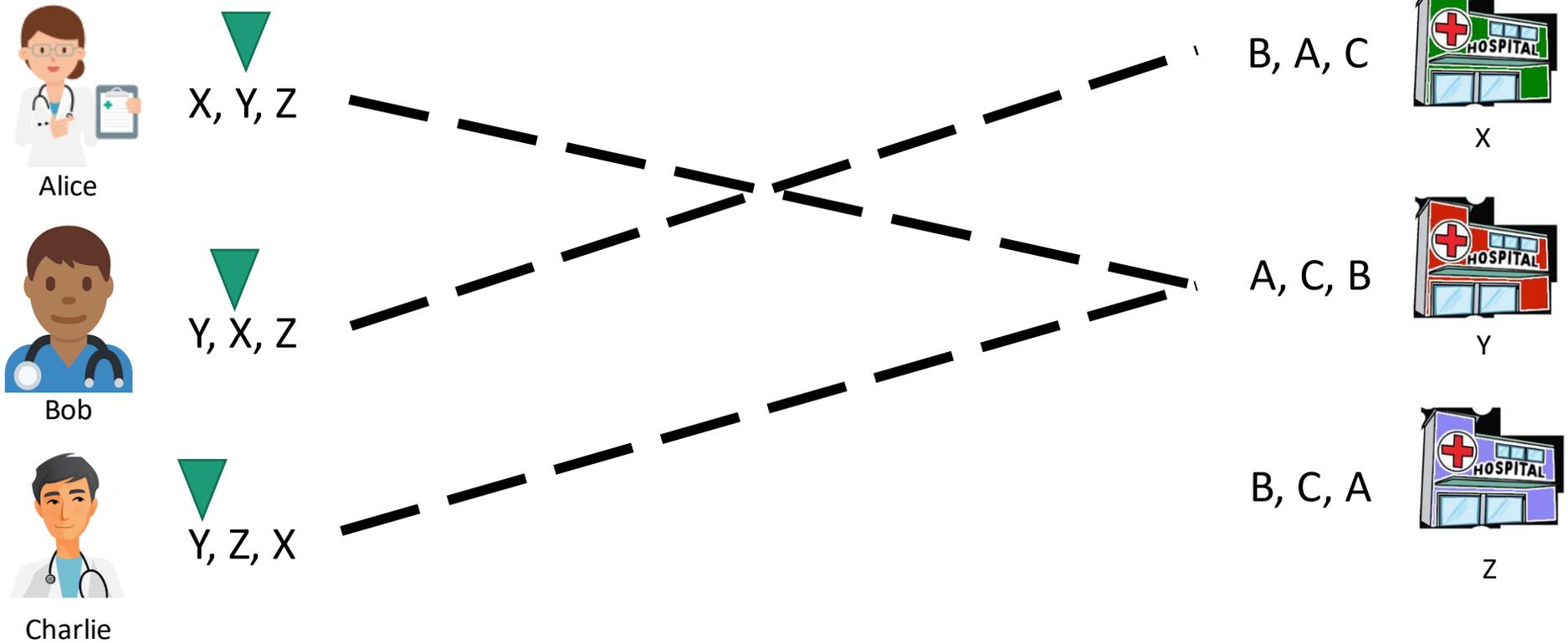
# DA Example Run 2



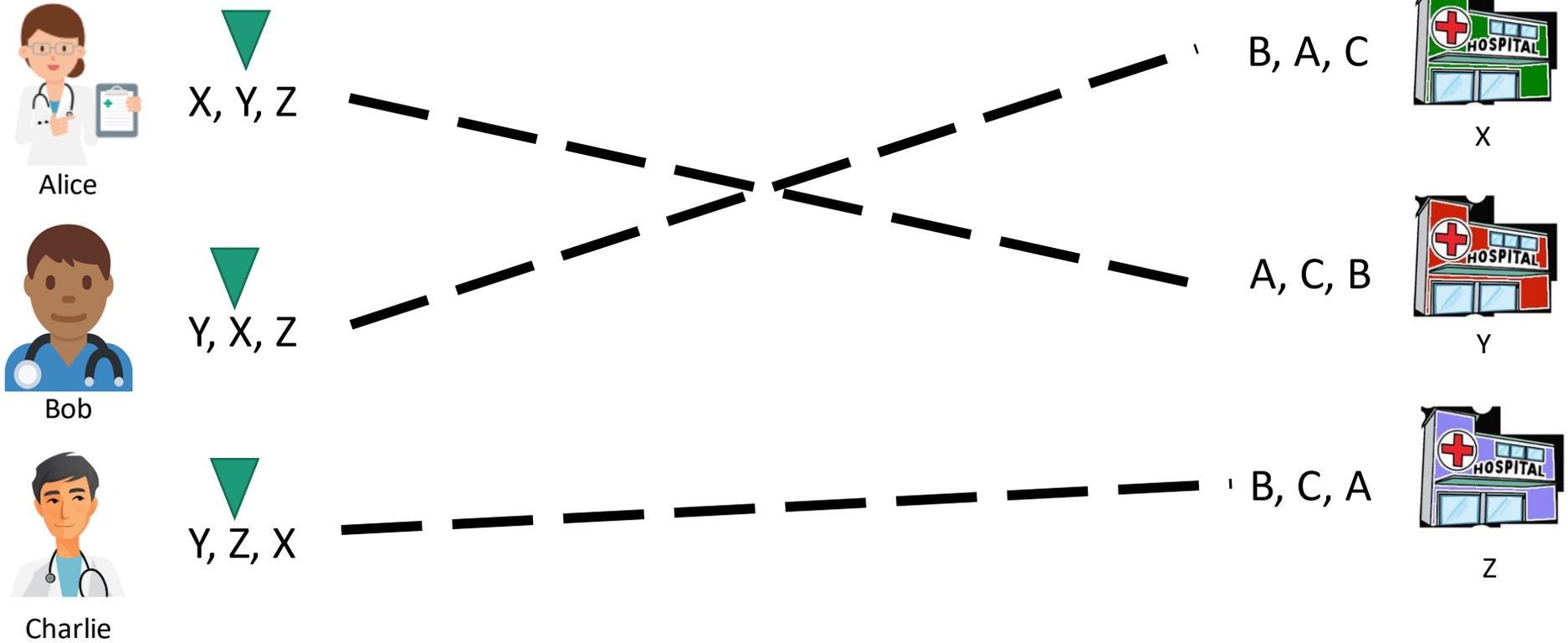
# DA Example Run 2



# DA Example Run 2



# DA Example Run 2



# DA Example Run 2



Alice

X, Y, Z



Bob

Y, X, Z



Charlie

Y, Z, X

B, A, C



X

A, C, B



Y

B, C, A



Z

# Deferred Acceptance Algorithm

Deferred-Acceptance(Doctors,Hospitals):

```
// initialize
```

```
freeDoctors ← Doctors
```

```
for all d in Doctors:
```

```
    d.current ← 0
```

```
for all h in Hospitals:
```

```
    h.D ← NIL
```

```
// main loop
```

```
while (exists d in freeDoctors)
```

```
    h ← d.rank(d.current++) // h is d's  
                           next favorite
```

```
    if (h.free = true)
```

```
        h.D ← d; h.free ← false
```

```
        remove d from freeDoctors
```

```
    else-if (h.rank(d) < h.rank(h.D))
```

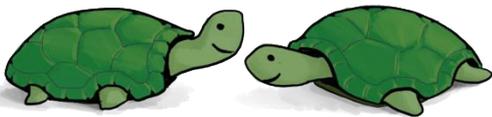
```
        add h.D to freeDoctors
```

```
        h.D ← d
```

```
        remove d from freeDoctors
```

```
return (h,h.D) for all h in Hospitals
```

```
// h prefers d to  
previous match
```



Think-pair-share!  
**Running time?**

# Deferred Acceptance Algorithm

## Running time:

Each iteration of  
while loop =  $O(1)$

Each iteration:  
We +1 **d.current**  
for some **doctor**

We always have:

**d.current**  $\leq n$   
for every **doctor** (There are  
 $n$  hospitals...)

Therefore, total  
run-time =  $O(n^2)$

```
// main loop
while (exists d in freeDoctors)
    h ← d.rank(d.current++) // h is d's
                           next favorite
    if (h.free = true)
        h.D ← d; h.free ← false
        remove d from freeDoctors
    else-if (h.rank(d) < h.rank(h.D))
        add h.D to freeDoctors
        h.D ← d
        remove d from freeDoctors
return (h,h.D) for all h in Hospitals
```

# DA algorithm

- Does it work?

- Yes!



- Is it fast?

- $O(n^2)$  - this is linear in the input size!

At worst exhaust through every doctor's  
preference list

# Deferred Acceptance works!

**Theorem:** Given  $n$  doctors and  $n$  hospitals,  
DA algorithm outputs a complete stable matching.

**Corollary:** A stable matching exists.  
(This is not obvious!)

# Proof of Theorem

**Theorem:** Given  $n$  doctors and  $n$  hospitals,  
DA algorithm outputs a complete stable matching.

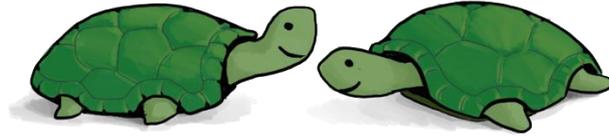
**Proof:** Follows from Claims 1+3 below...

**Claim 1:** At every iteration, current match is stable  
w.r.t. non-free doctors and hospitals.

**Claim 2:** Once a hospital is matched, it remains matched  
(possibly to a different doctor) until end of algorithm.

**Claim 3:** At the end of algorithm, every doctor/hospital is matched.

# Proof of claims



Think-pair-share:  
**Prove these!**

**Claim 1:** At every iteration, current match is stable w.r.t. non-free doctors and hospitals.

**Proof by contradiction:** Suppose  $(d, h)$  blocking pair.

- $d$  is currently matched to worse hospital than  $h$ .
- $d$  already tried to match to  $h$ .
- $h$  either refused  $d$  or left  $d$  later. Why?
- $h$  must be matched to better doctor than  $d$  – contradiction!

**Claim 2:** Once a hospital is matched, it remains matched (possibly to a different doctor) until end of algorithm.

**“Proof”:** obvious from algorithm

**Claim 3:** At the end of algorithm, every doctor/hospital is matched.

**Proof by contradiction:** Suppose  $(d, h)$  still free.

End of algorithm →  $d$  already tried to match to  $h$ .

→ after that step,  $h$  wasn't free → by Claim 2, contradiction!

# Deferred Acceptance works!

**Theorem:** Given  $n$  doctors and  $n$  hospitals,  
DA algorithm outputs a complete stable matching.

**Corollary:** A stable matching exists.

**Claim 1:** At every iteration, current match is stable  
w.r.t. non-free doctors and hospitals.

**Claim 2:** Once a hospital is matched, it remains matched  
(possibly to a different doctor) until end of algorithm.

**Claim 3:** At the end of algorithm, every doctor/hospital is matched.

# What have we learned?

**Blocking Pair:** A doctor and hospital that prefer each other over their respective matches.

**Stable Matching:** A matching without blocking pairs!

## **Deferred Acceptance Algorithm**

*“Tentatively match each free doctor to best interested hospital.  
Allow the hospital to leave match when a better doctor arrives.”*

Runs in time  $O(n^2)$  = linear in input size 😊

# Today

- Hospitals/residents problem
- **Stable matchings**
  - Solve the hospitals/residents problem
  - But can we find them?
- **Deferred Acceptance Algorithm**
  - Find stable matchings!
- Discussion, applications and non-applications



# The optimal stable matching?

DA algorithm found *a* stable matching...

- Is it *optimal*?
- What does optimality mean?



Theorem: The matching returned by DA is **doctor-optimal**, i.e. every doctor is matched to favorite hospital possible in any stable matching.

Corollary: Doctors cannot gain from misreporting their preferences.

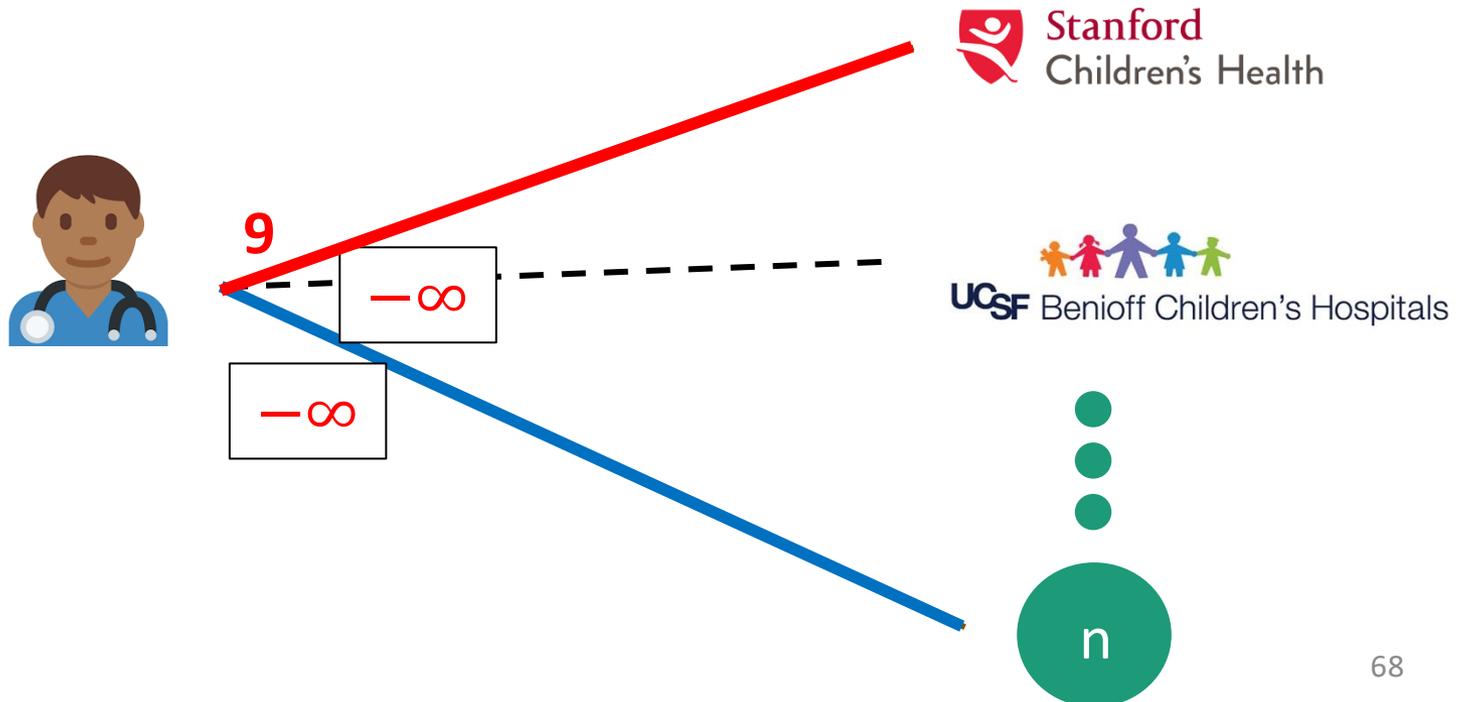


Prove this theorem!

# Stable Matching and incentives

- Doctor 2 may tell you he only wants to go to Stanford, but...

Corollary: This won't help him if we find Stable Matching with DA!

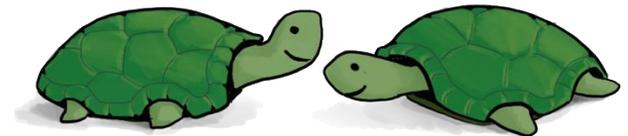




# The optimal stable matching?

Theorem: The matching returned by DA is **hospital-worst**, i.e. every hospital is matched to *least*-favorite doctor possible in any stable matching.

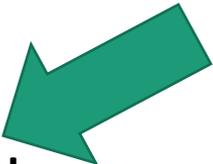
Caution: Hospitals *can* gain from misreporting their preferences.



Think-pair-share:

How would you find a hospital-optimal stable matching?  
Should actual matching be doctor- or hospital-optimal?

# Today

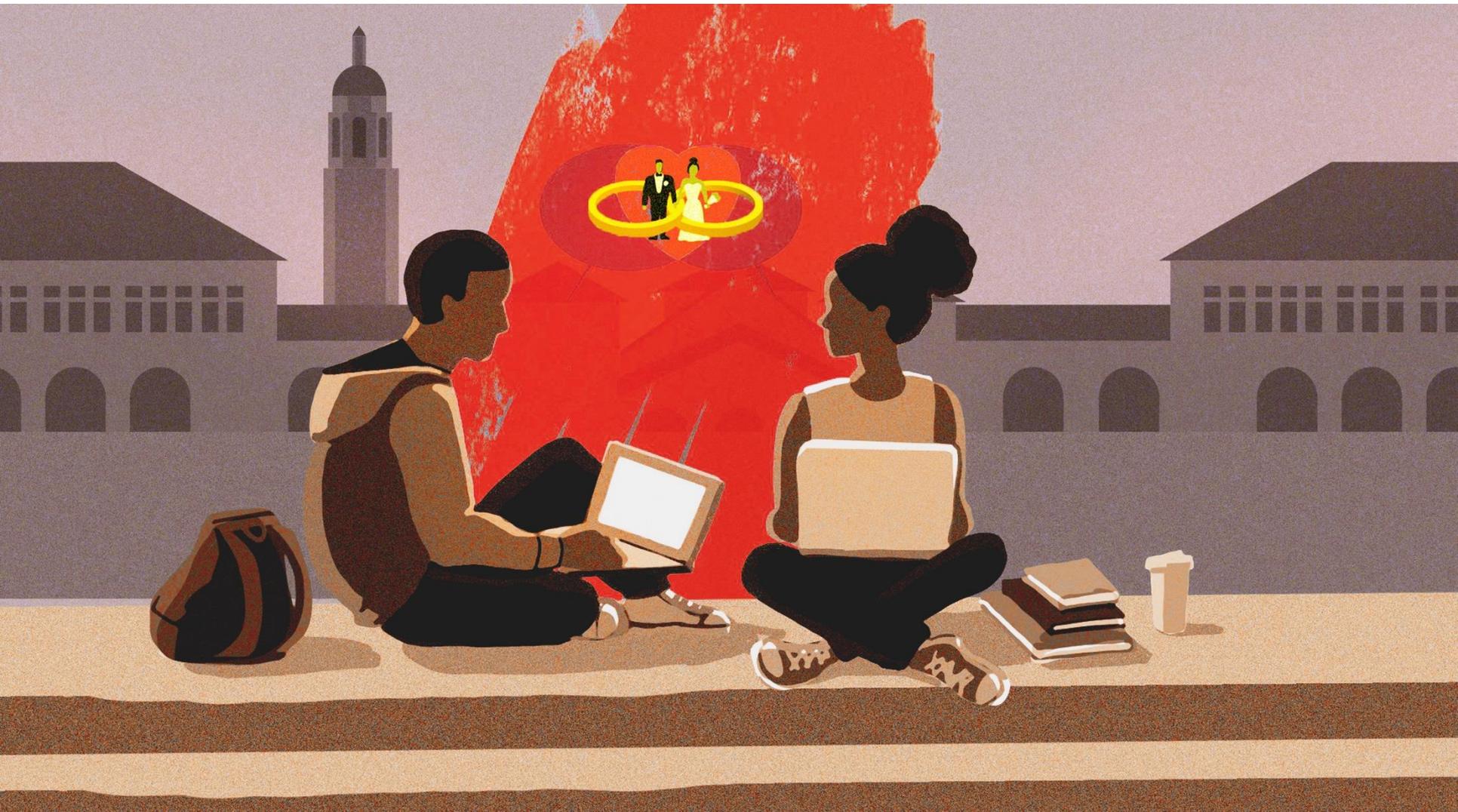
- Hospitals/residents problem
  - **Stable matchings**
    - Solve the hospitals/residents problem
    - But can we find them?
  - **Deferred Acceptance Algorithm**
    - Find stable matchings!
  - Discussion, applications and non-applications
- 

# More applications

- [NYC Public Schools matching](#)
- [US Army talent-based branching](#)
- [Chile's centralized university admissions system](#)



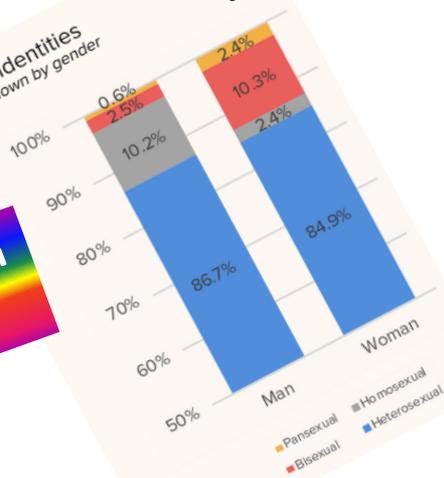
# Stanford Marriage Pact



# Stanford Marriage Pact

- Matches between Stanford students who want to make a *pact*:  
“If we don’t get married by time X, we’ll marry each other.”
- Historically, Gale-Shapley’s original paper talked about *Stable Marriage*
  - Men = doctors; women = hospitals.
- Original Marriage Pact used variant of Deferred Acceptance
  - It doesn’t any more...

Sexual Identities  
broken down by gender



Preference graph  
is not bipartite!

**Marriage Pact doesn't need stability:**  
It is meant to be a back-up match-  
Couples are encouraged to find outside matches!

# Today

- Hospitals/residents problem
- **Stable matchings**
  - Solve the hospitals/residents problem
  - But can we find them?
- **Deferred Acceptance Algorithm**
  - Find stable matchings!
- Discussion, applications and non-applications

# Recap

## Blocking Pair:

A doctor and hospital that prefer each other over their respective matches.

Stable Matching: A matching without blocking pairs!

## Deferred Acceptance Algorithm:

*“Tentatively match each free doctor to best interested hospital.  
Allow the hospital to leave match when a better doctor arrives.”*

Runs in time  $O(n^2)$  = linear in input size 😊

Doctor optimality: Stable matching found with DA is optimal for doctors

It's important to **think** about how **our algorithms affect people.**