CSE 331: Algorithms & Complexity

“Gale-Shapley”

Prof. Charlie Anne Carlson (She/Her)

Lecture 4

Monday September 5th, 2025

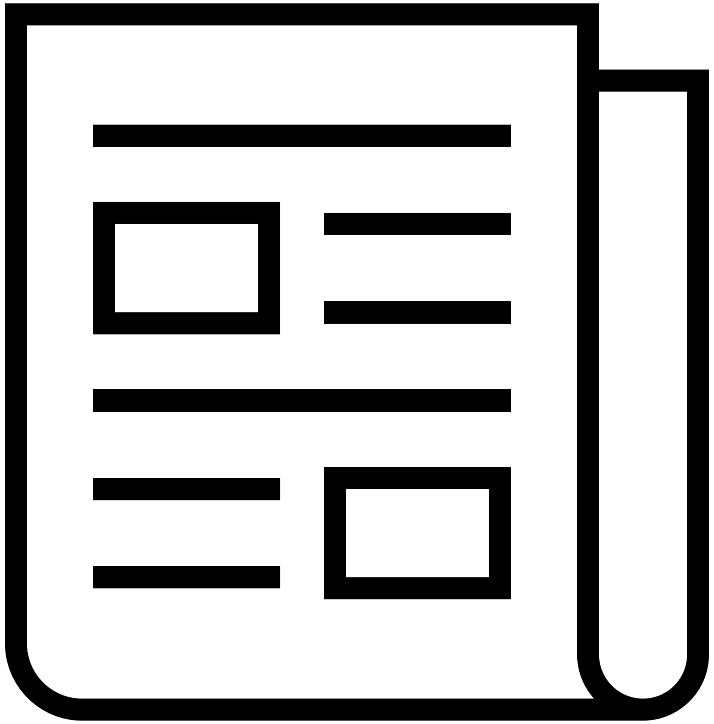
University at Buffalo

Schedule

1. Course Updates
2. Group Project
3. Recap
4. Gale-Shapely

Course Updates

- Office Hours Posted
- Complete Syllabus Quiz
- HW 2 Posted Tuesday



Group Project

CSE 331 Syllabus Piazza Schedule Homeworks Autolab Project Support Pages

CSE 331 Project

Fall 2025

Details and motivations for the project.

Motivation

CSE 331 is primarily concerned with the technical aspects of algorithms: how to design them and then how to analyze their correctness and runtime. However, algorithms are pervasive in our world and are commonplace in many aspects of society. The main aim of the project is to have you explore some of the social implications of algorithms.

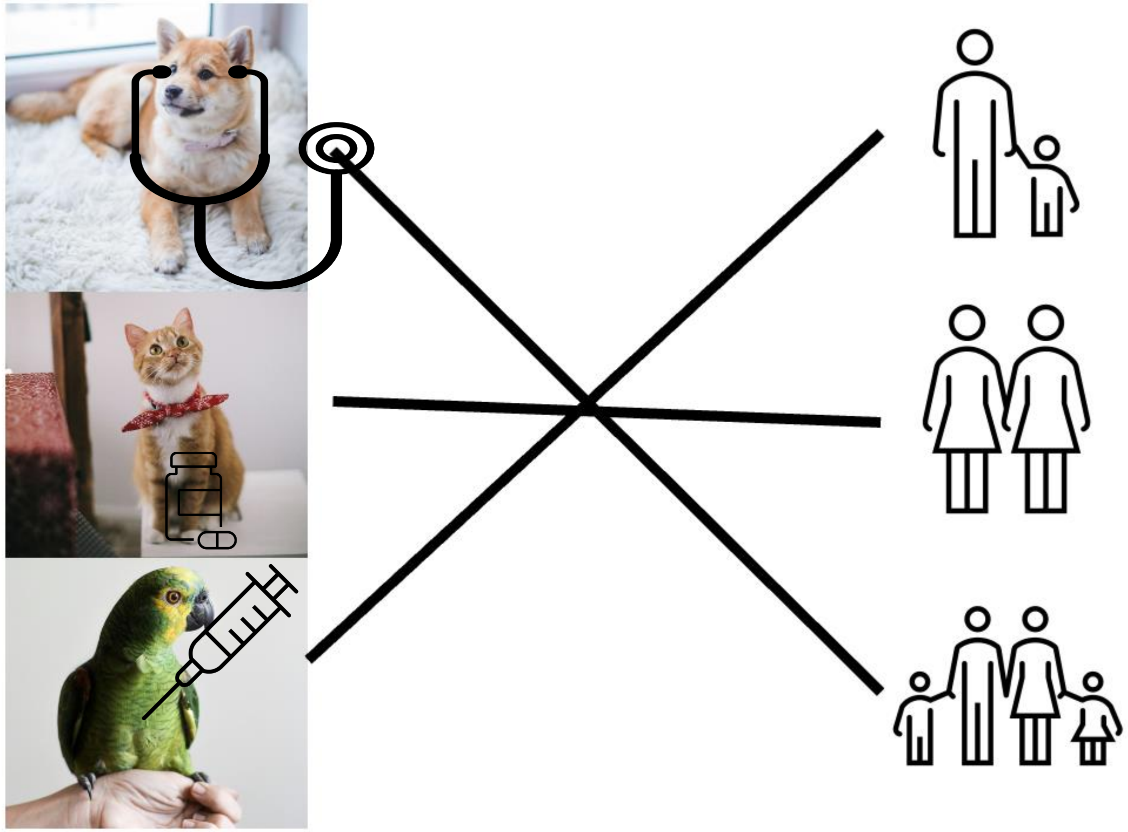
Just to give some examples for such implications:

- Big data is hot these days and there is a (not-uncommon) belief that by running (mainly machine learning) algorithms on big data, we can detect patterns and use those to potentially make policy decisions. Here is a cautionary talk:

Hospital Assignments

We say that an assignment M is **stable** if there exist no resident p and hospital f such that

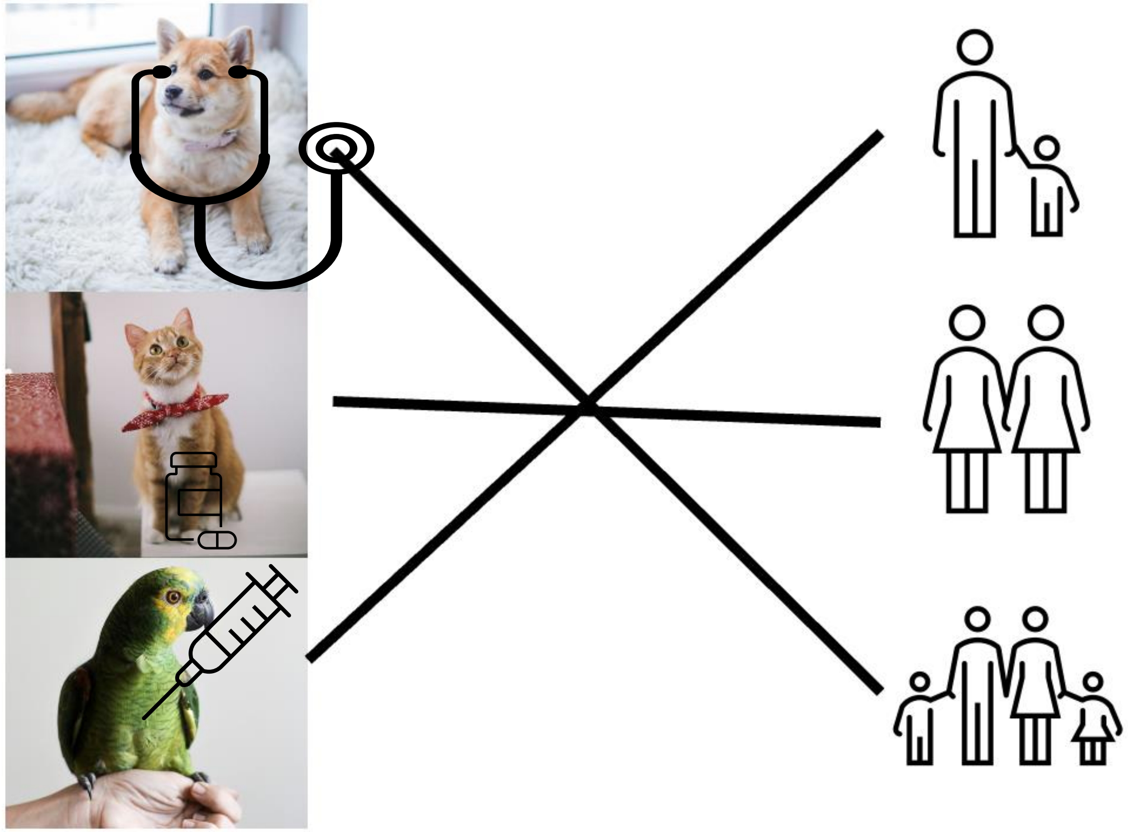
- ?
- ?



Hospital Assignments

We say that a matching M is **stable** if there exist no resident p and hospital f such that

- p prefers f to their current hospital and
- f prefers p to their current resident.

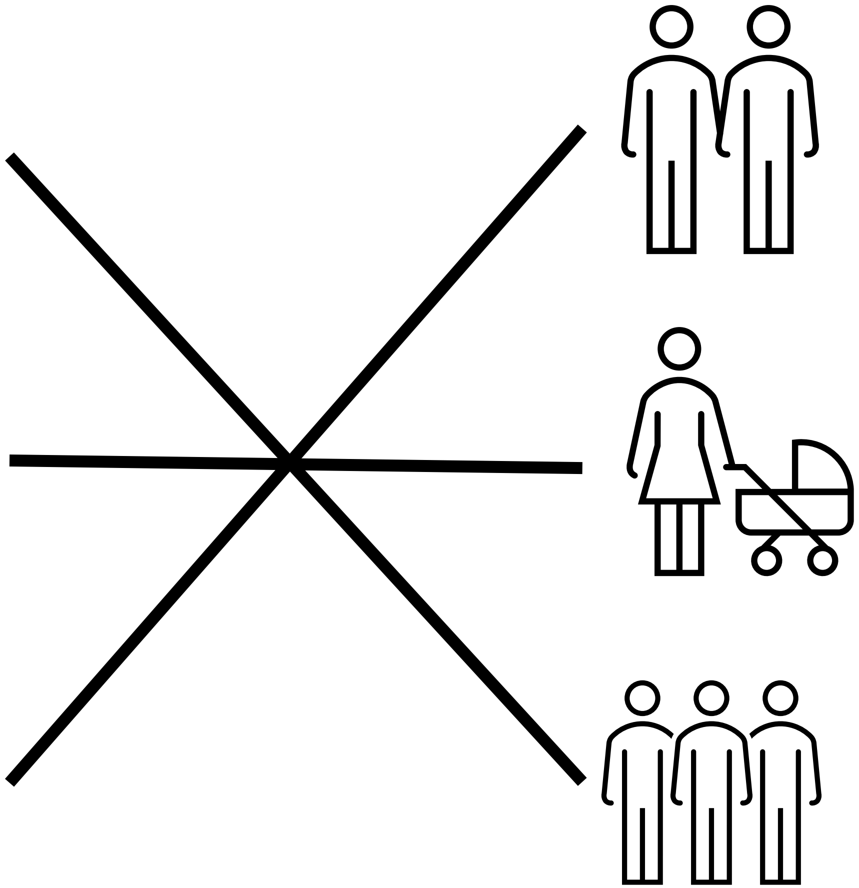


Who is Unstable?

Clover
D > E > F

Dr. Fluffy
D > F > E

Ronin
E > D > F

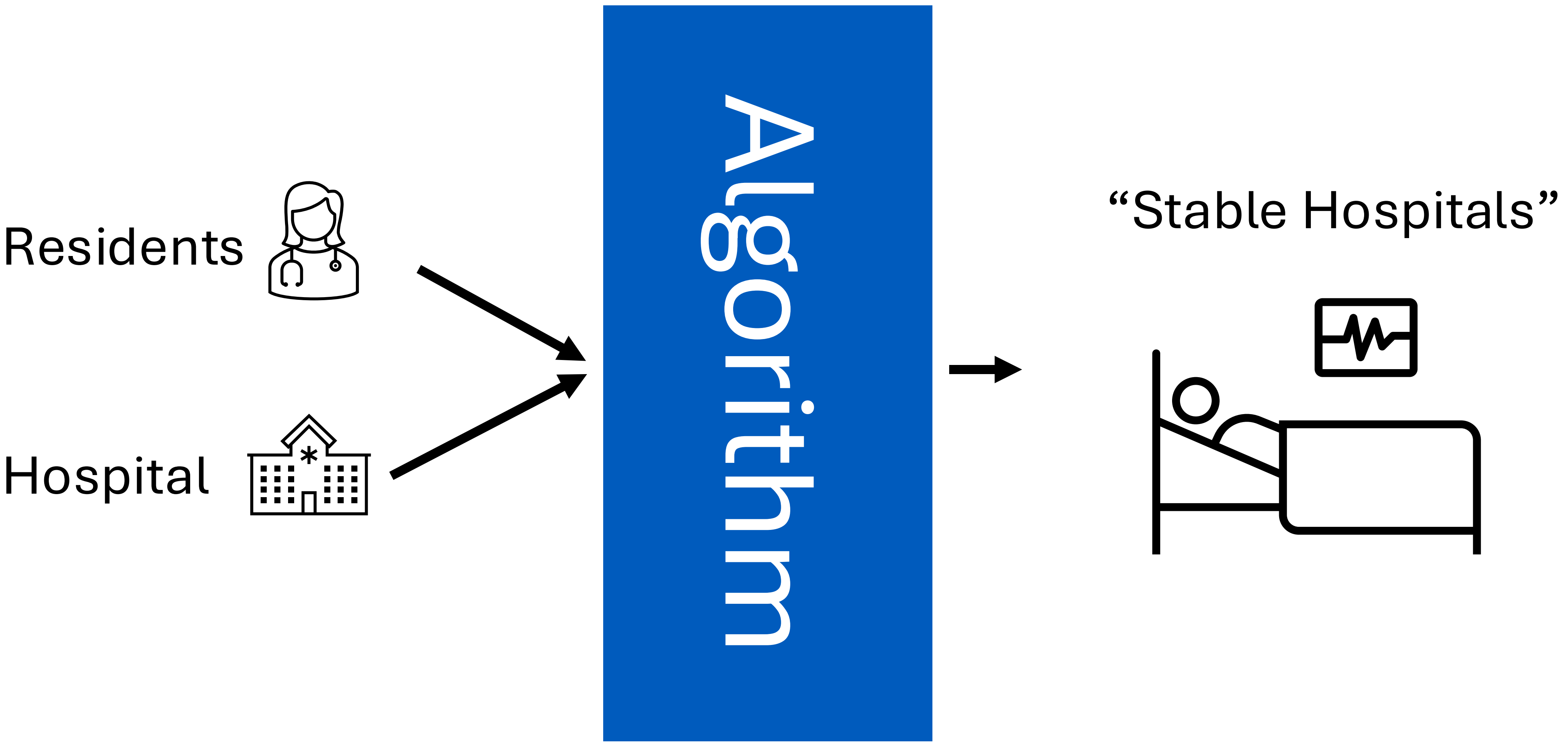


Family D
Snake > Bunny > Spider

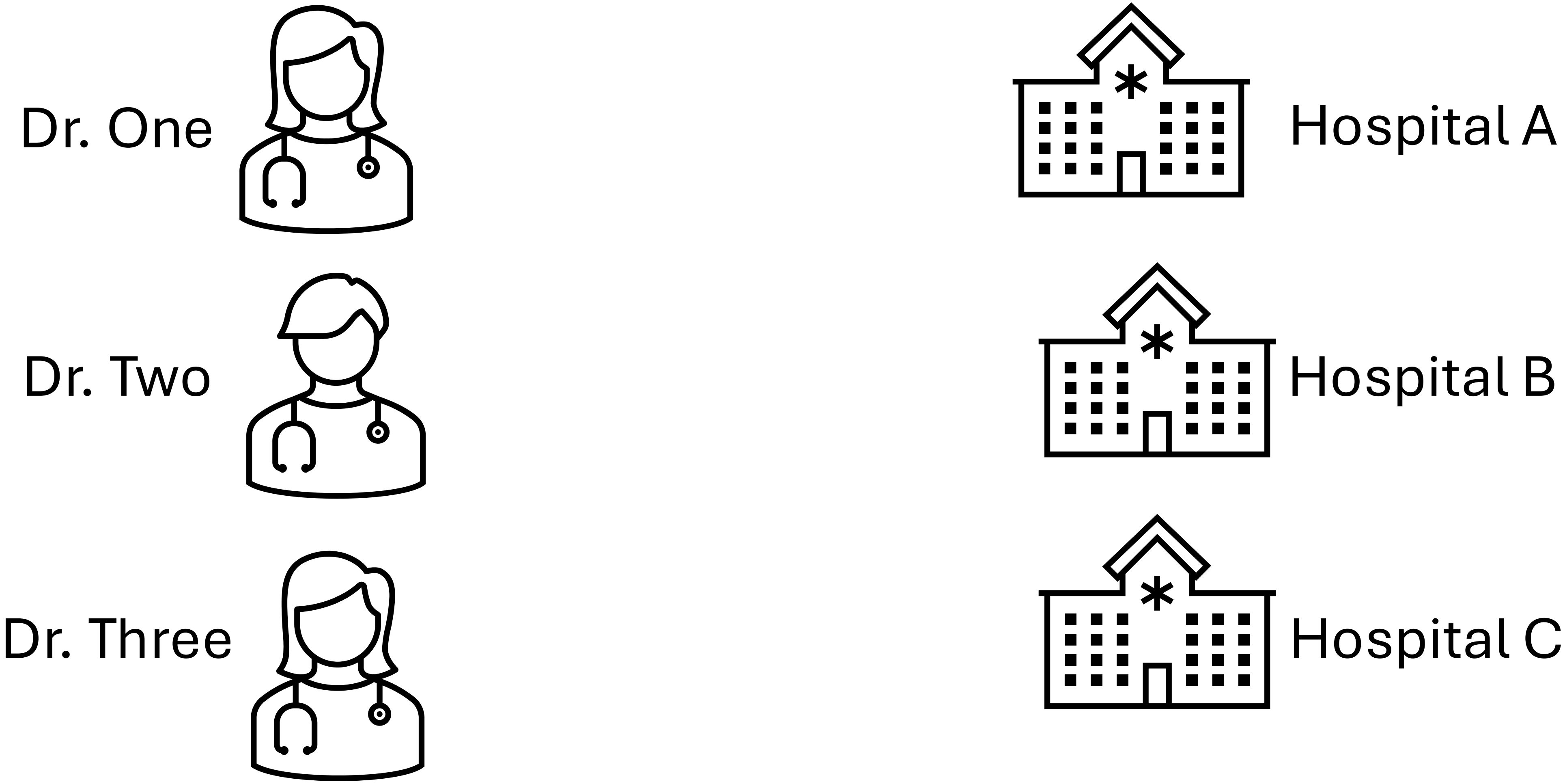
Family E
Bunny > Snake > Spider

Family F
Spider > Bunny > Snake

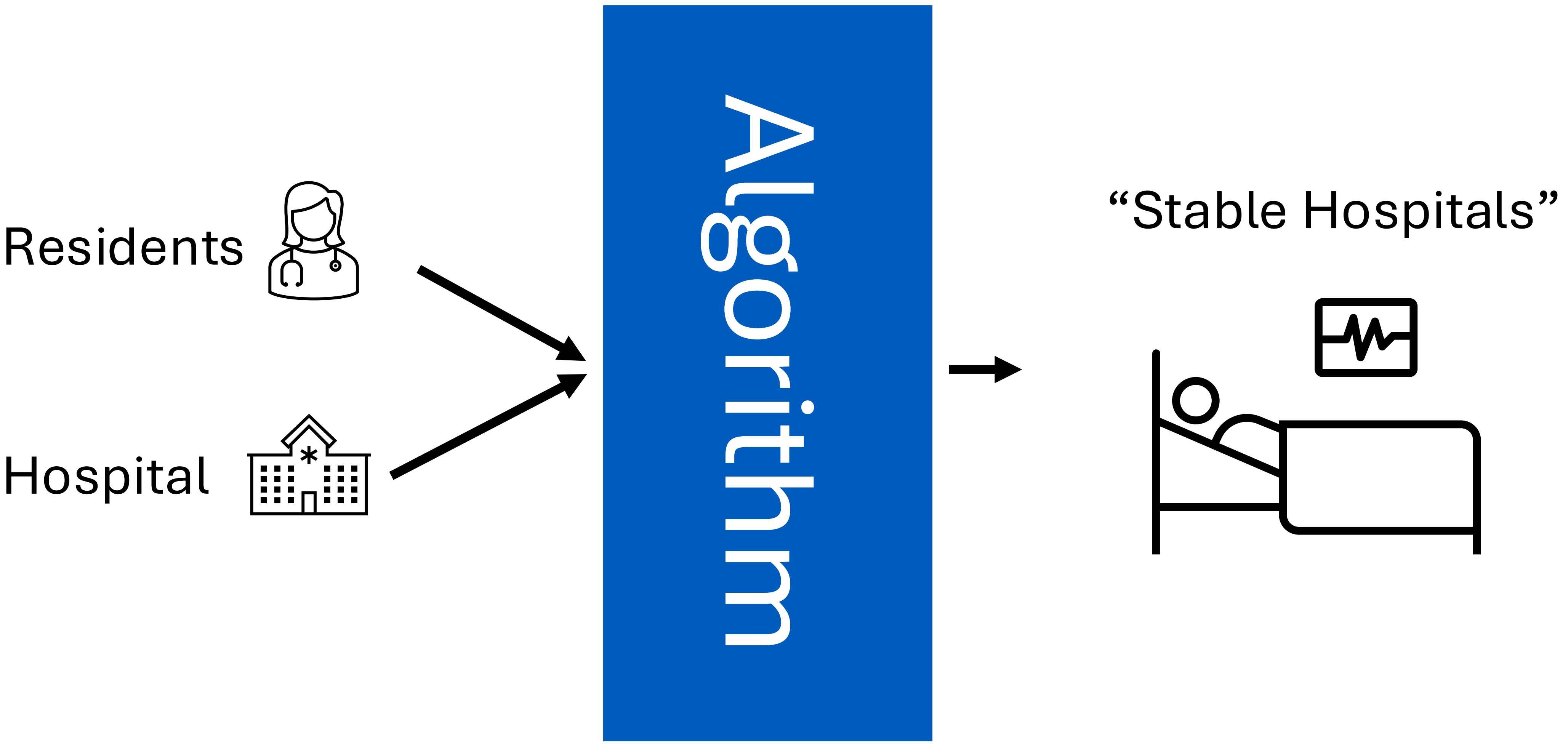
Hospital Resident Assignments Problem



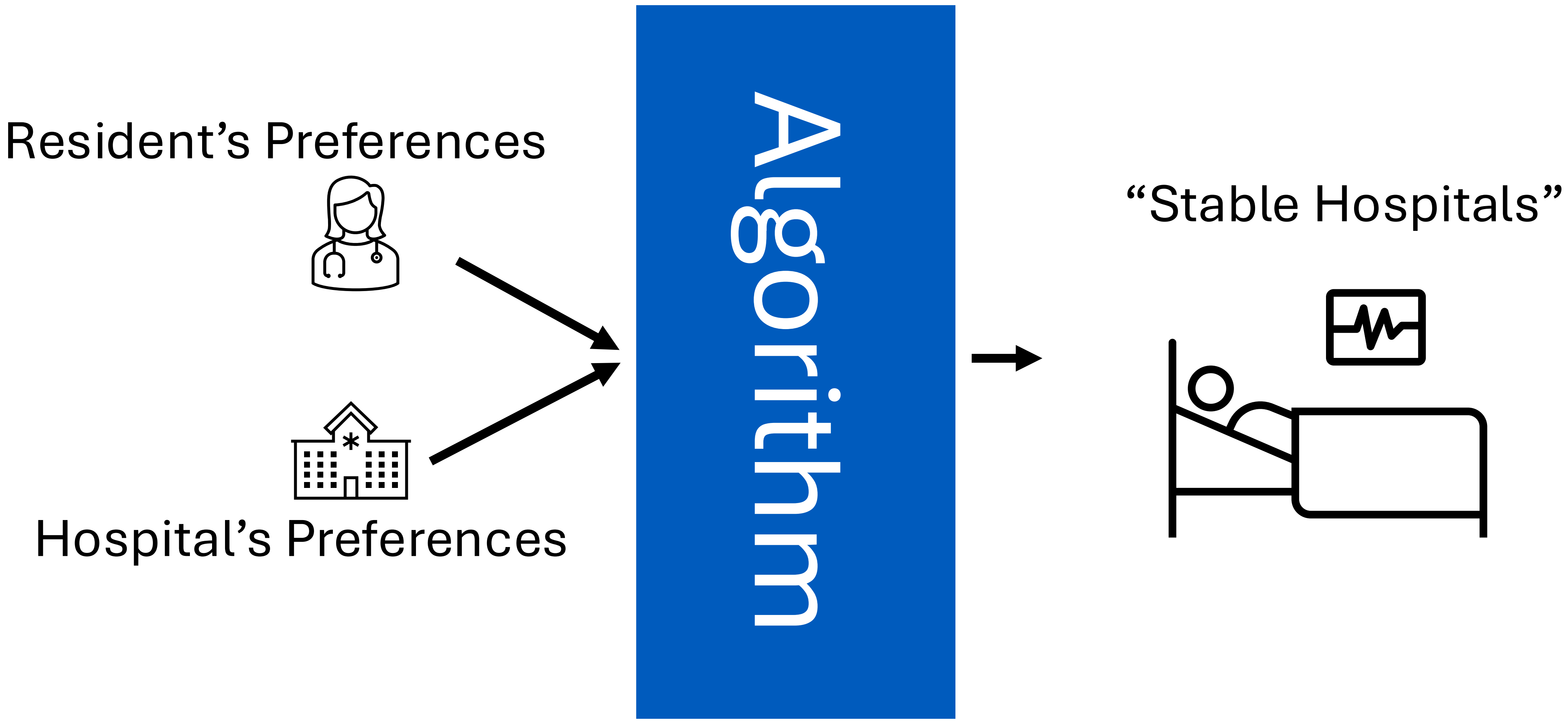
Resident Assignments



Q: What is missing?



Q: What is missing?



Resident Assignments

Dr. One
A > B > C

Dr. Two
A > C > B

Dr. Three
C > A > B

Hospital A
2 > 1 > 3

Hospital B
1 > 2 > 3

Hospital C
3 > 2 > 1

Gale-Shapley Algorithm

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

  $s \leftarrow$ first student on h's list to whom h has not yet proposed.

  IF (s is unmatched)

    Add h–s to matching M.

  ELSE IF (s prefers h to current partner h')

    Replace h'–s with h–s in matching M.

  ELSE

    s rejects h.

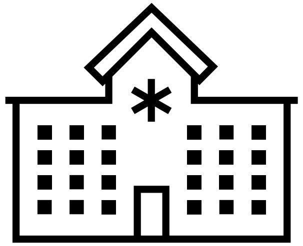
RETURN stable matching M.

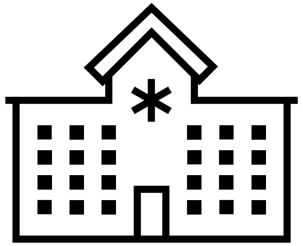
Resident Assignments

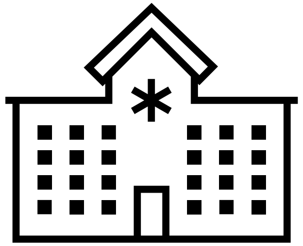
Dr. One
A > B > C


Dr. Two
A > C > B


Dr. Three
C > A > B



Hospital A
2 > 1 > 3


Hospital B
1 > 2 > 3


Hospital C
3 > 2 > 1

Gale-Shapley Algorithm

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

Replace h'–s with h–s in matching M.

ELSE

s rejects h.

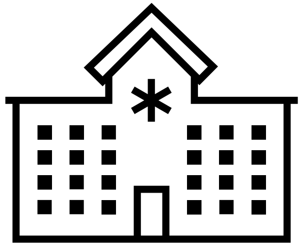
RETURN stable matching M.

Resident Assignments

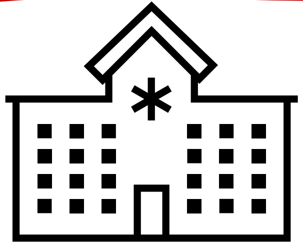
Dr. One
A > B > C

Dr. Two
A > C > B

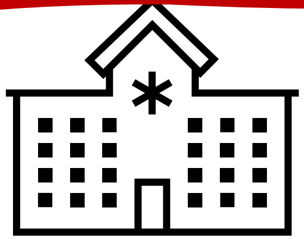
Dr. Three
C > A > B



Hospital A
2 > 1 > 3

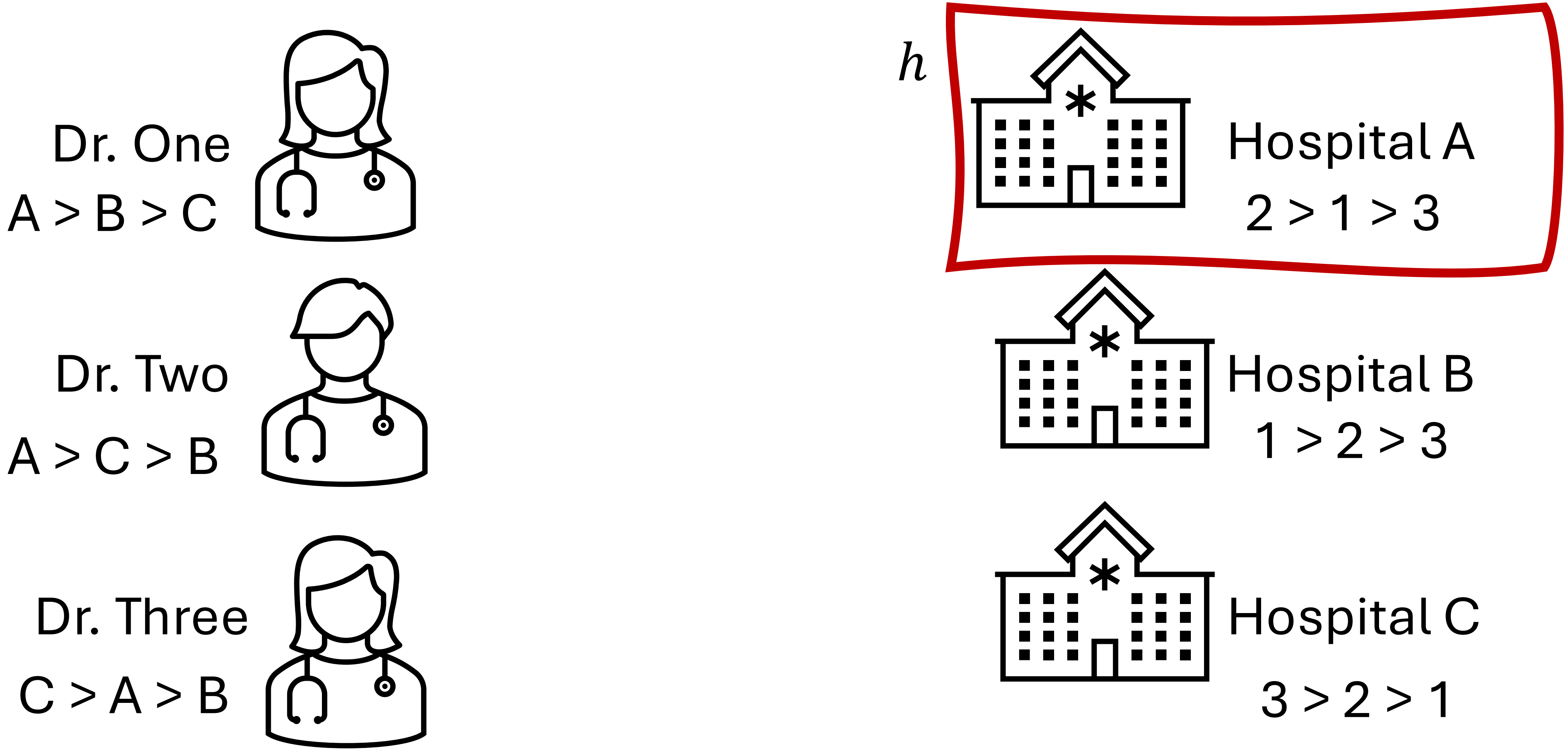


Hospital B
1 > 2 > 3



Hospital C
3 > 2 > 1

Resident Assignments



Gale-Shapley Algorithm

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

→ $s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

Replace h'–s with h–s in matching M.

ELSE

s rejects h.

RETURN stable matching M.

Resident Assignments

Dr. One
A > B > C

Dr. Two
A > C > B

Dr. Three
C > A > B

h

Hospital A
2 > 1 > 3

Hospital B
1 > 2 > 3

Hospital C
3 > 2 > 1

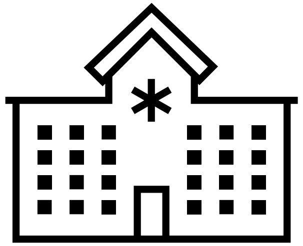
Resident Assignments

Dr. One
A > B > C

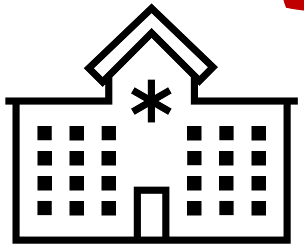
Dr. Two
A > C > B

Dr. Three
C > A > B

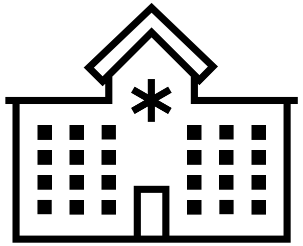
h



Hospital A
2 > 1 > 3



Hospital B
1 > 2 > 3



Hospital C
3 > 2 > 1

Gale-Shapley Algorithm

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

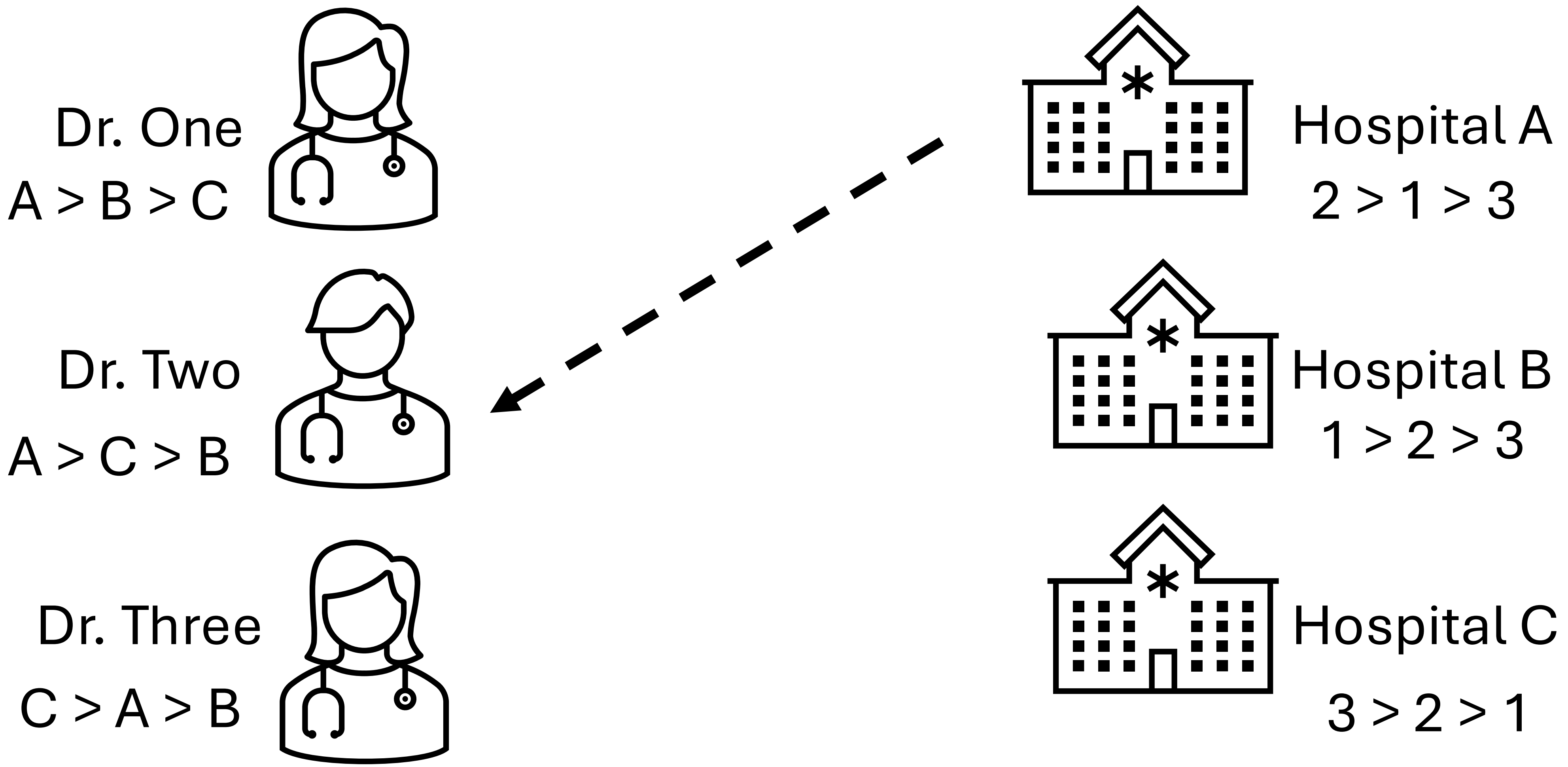
Replace h'–s with h–s in matching M.

ELSE

s rejects h.

RETURN stable matching M.

Resident Assignments



Gale-Shapley Algorithm

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

→ Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

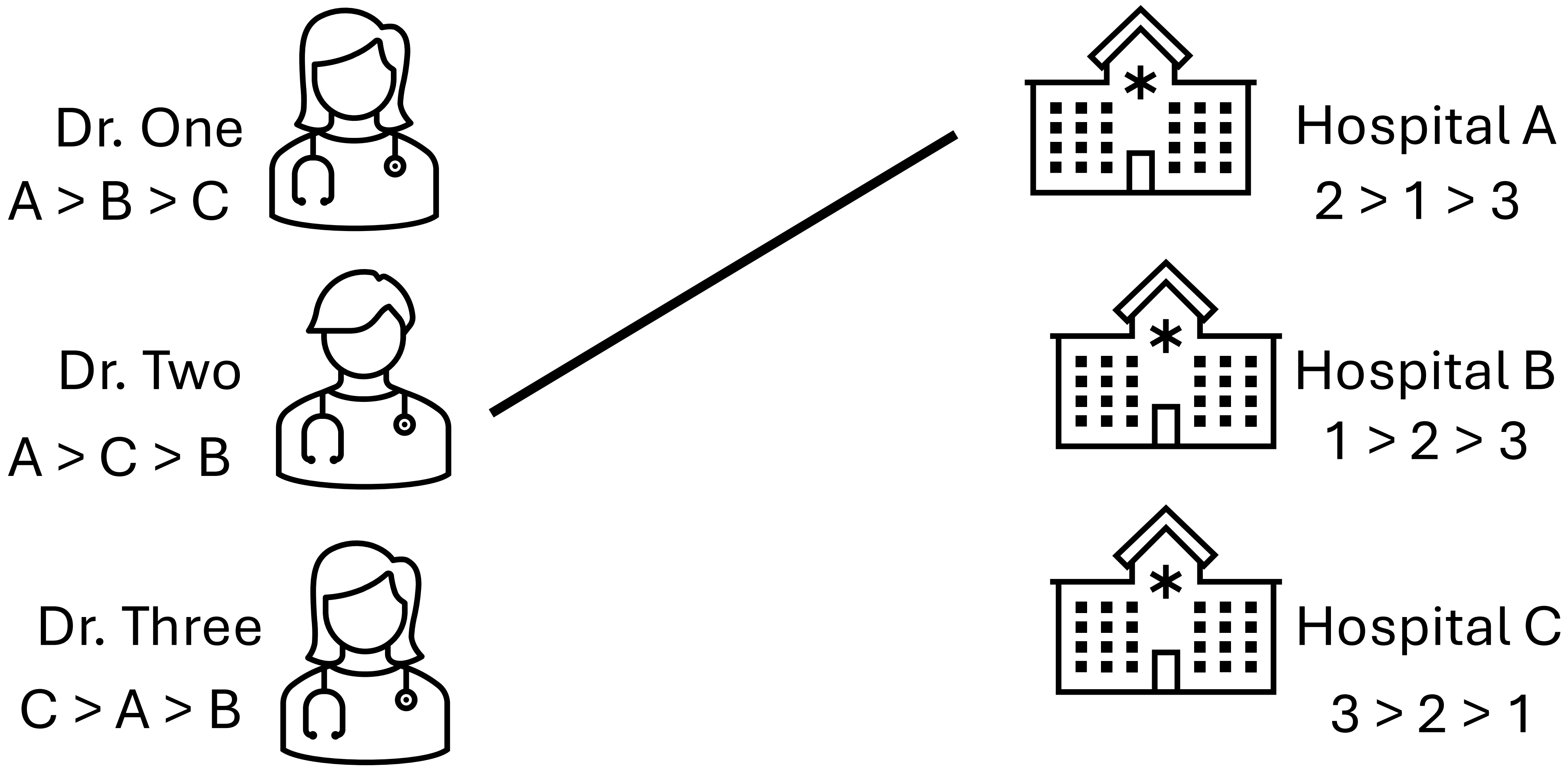
Replace h'–s with h–s in matching M.

ELSE

s rejects h.

RETURN stable matching M.

Resident Assignments



Gale-Shapley Algorithm

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

  $s \leftarrow$ first student on h's list to whom h has not yet proposed.

  IF (s is unmatched)

    Add h–s to matching M.

  ELSE IF (s prefers h to current partner h')

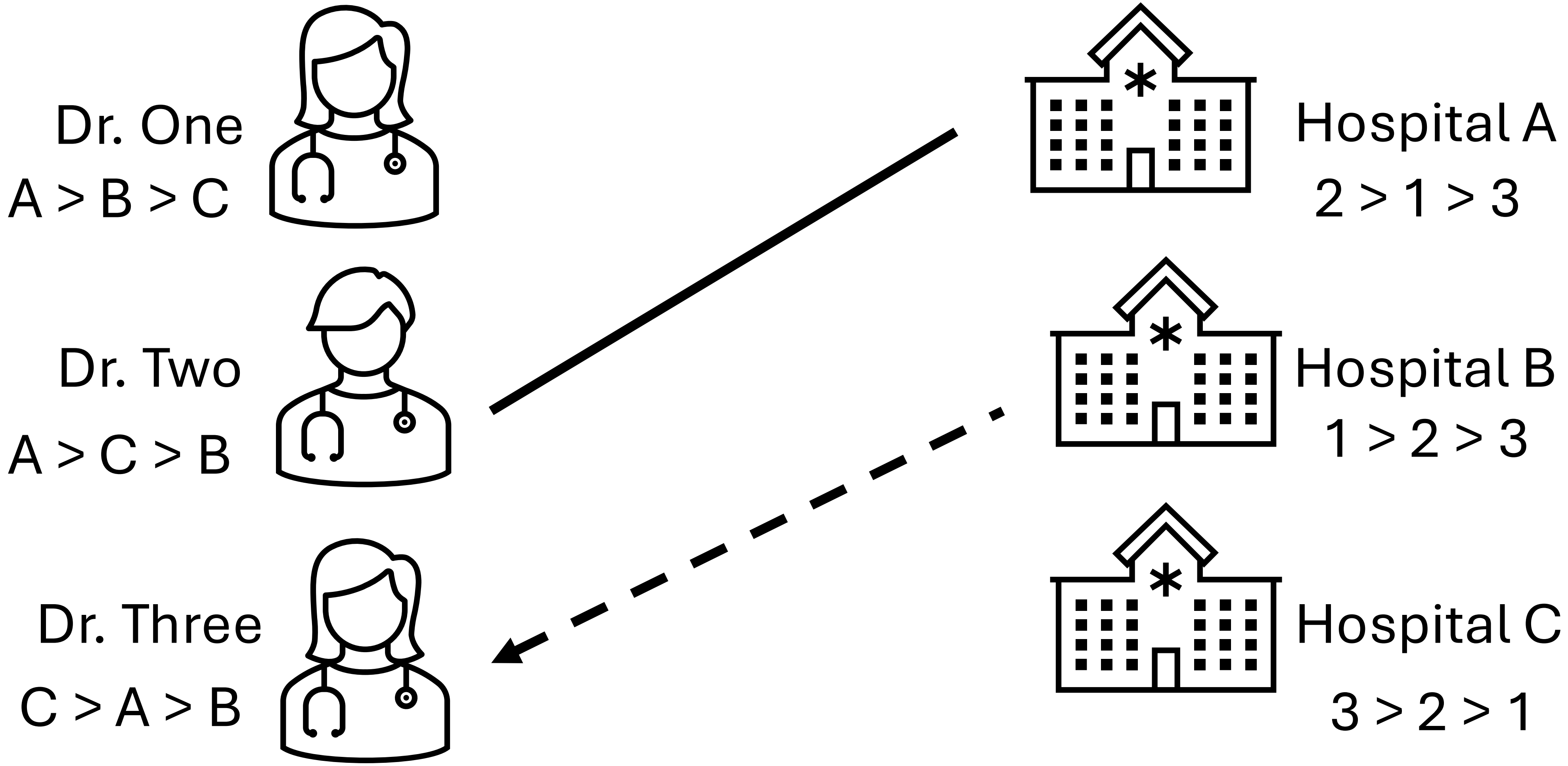
    Replace h'–s with h–s in matching M.

  ELSE

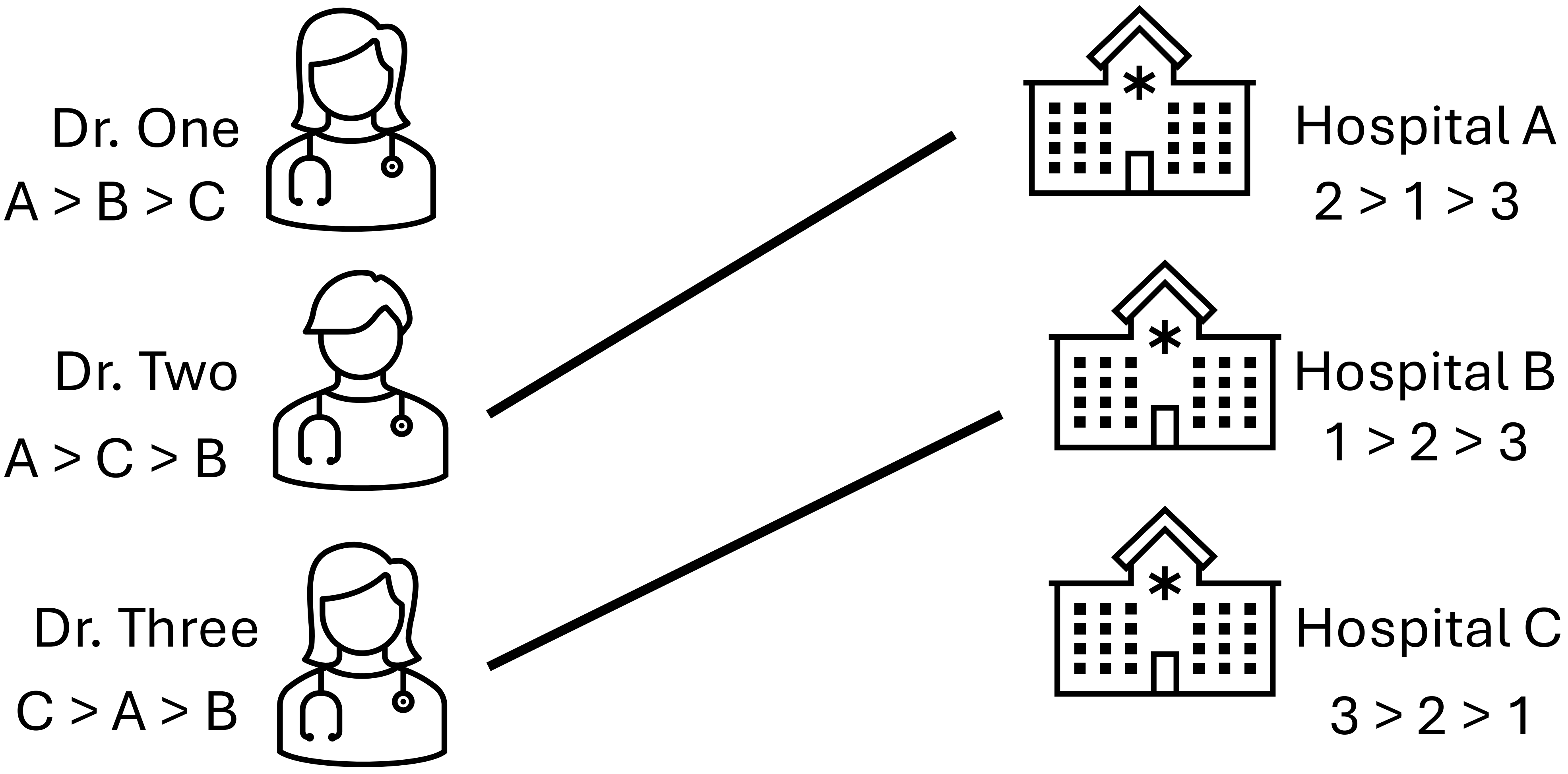
    s rejects h.

RETURN stable matching M.

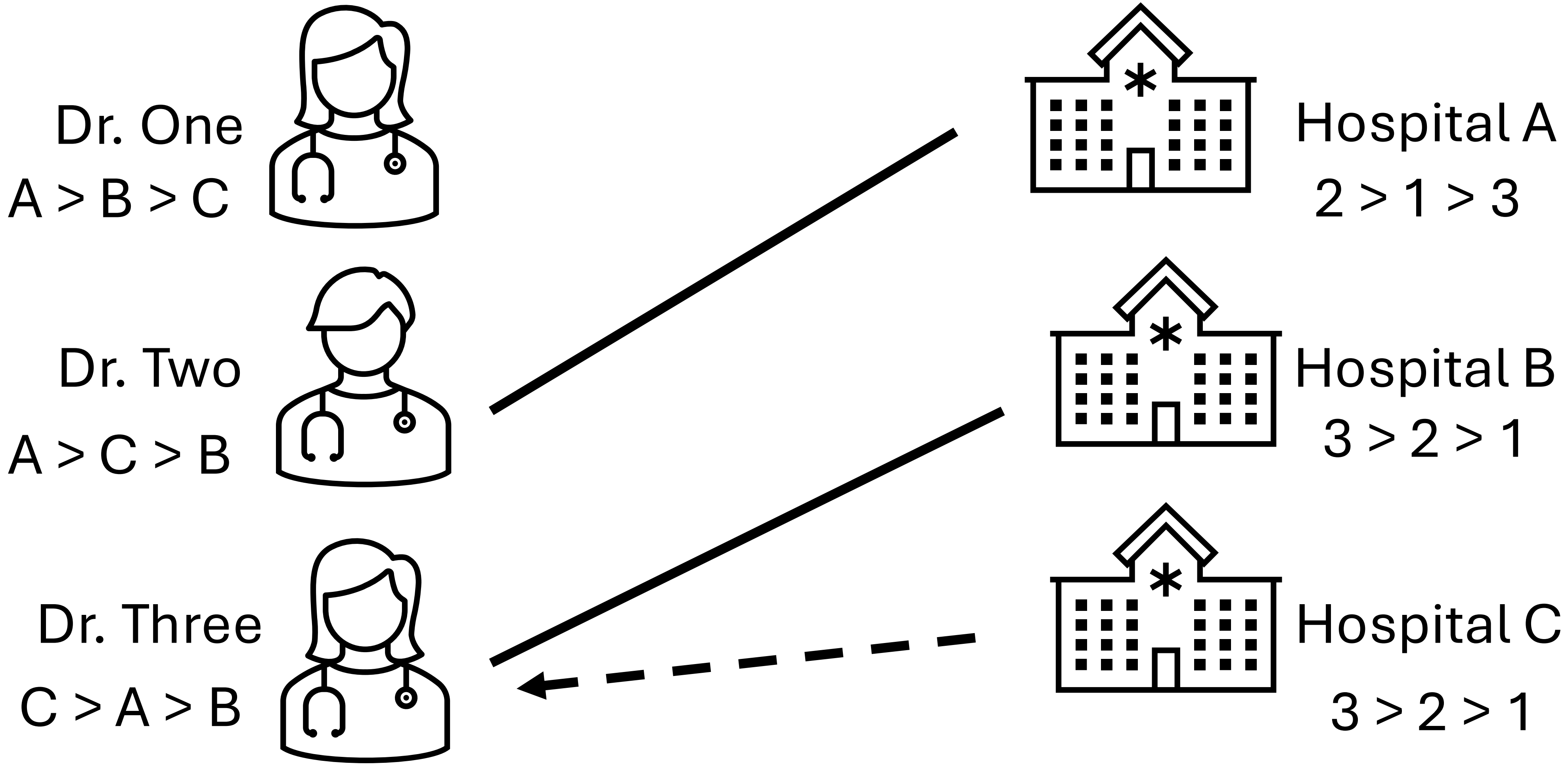
Resident Assignments



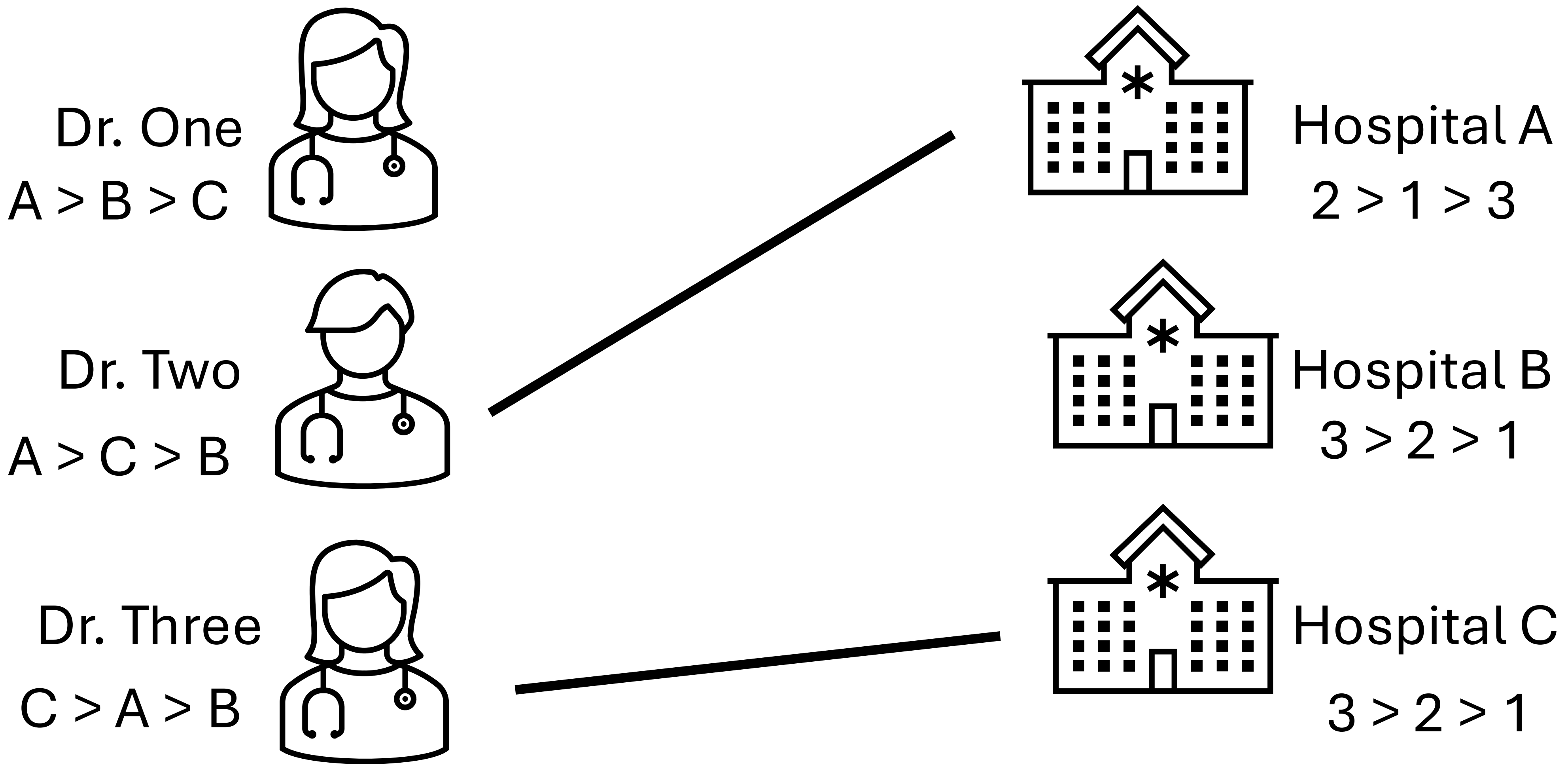
Resident Assignments



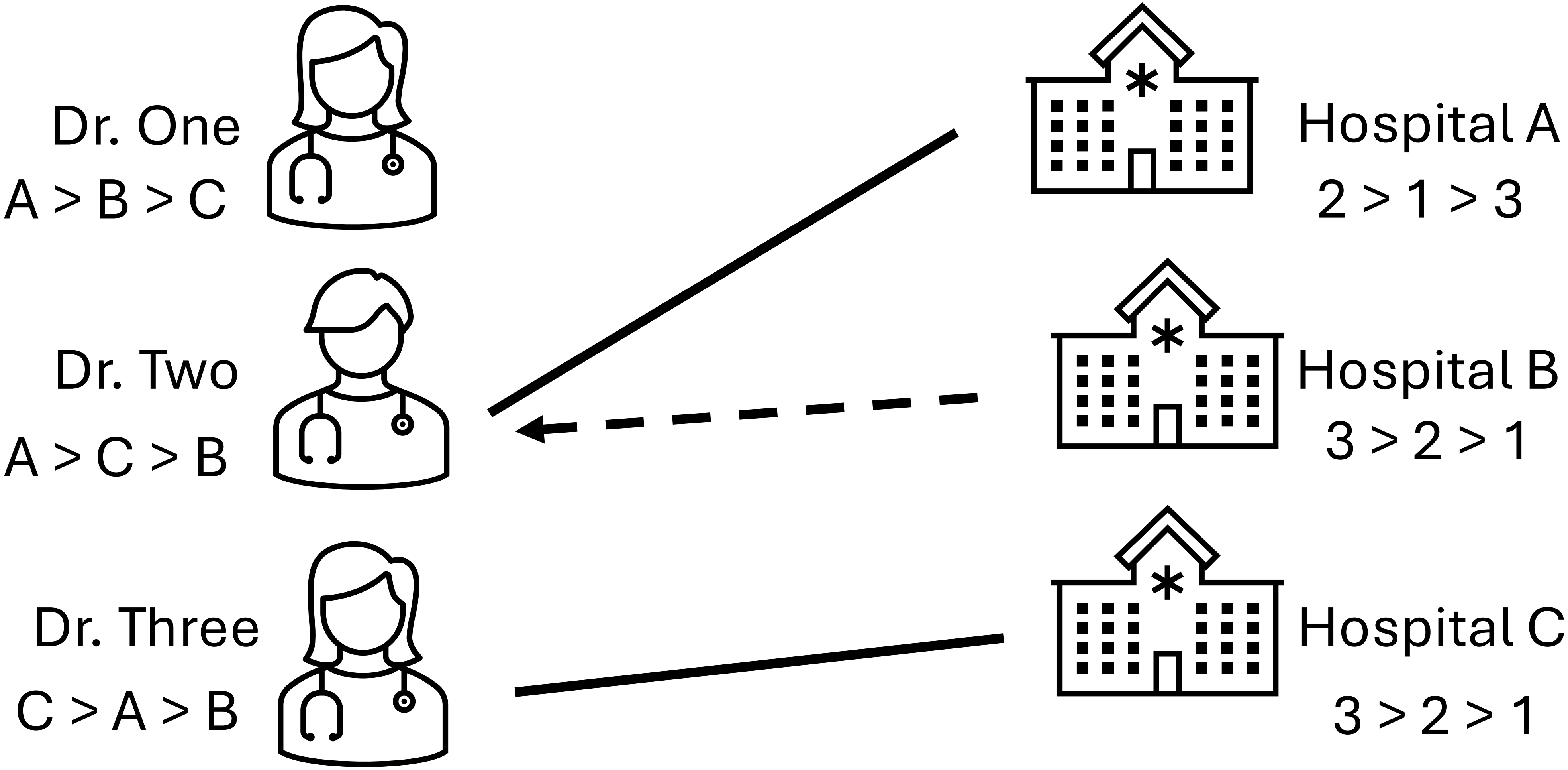
Resident Assignments



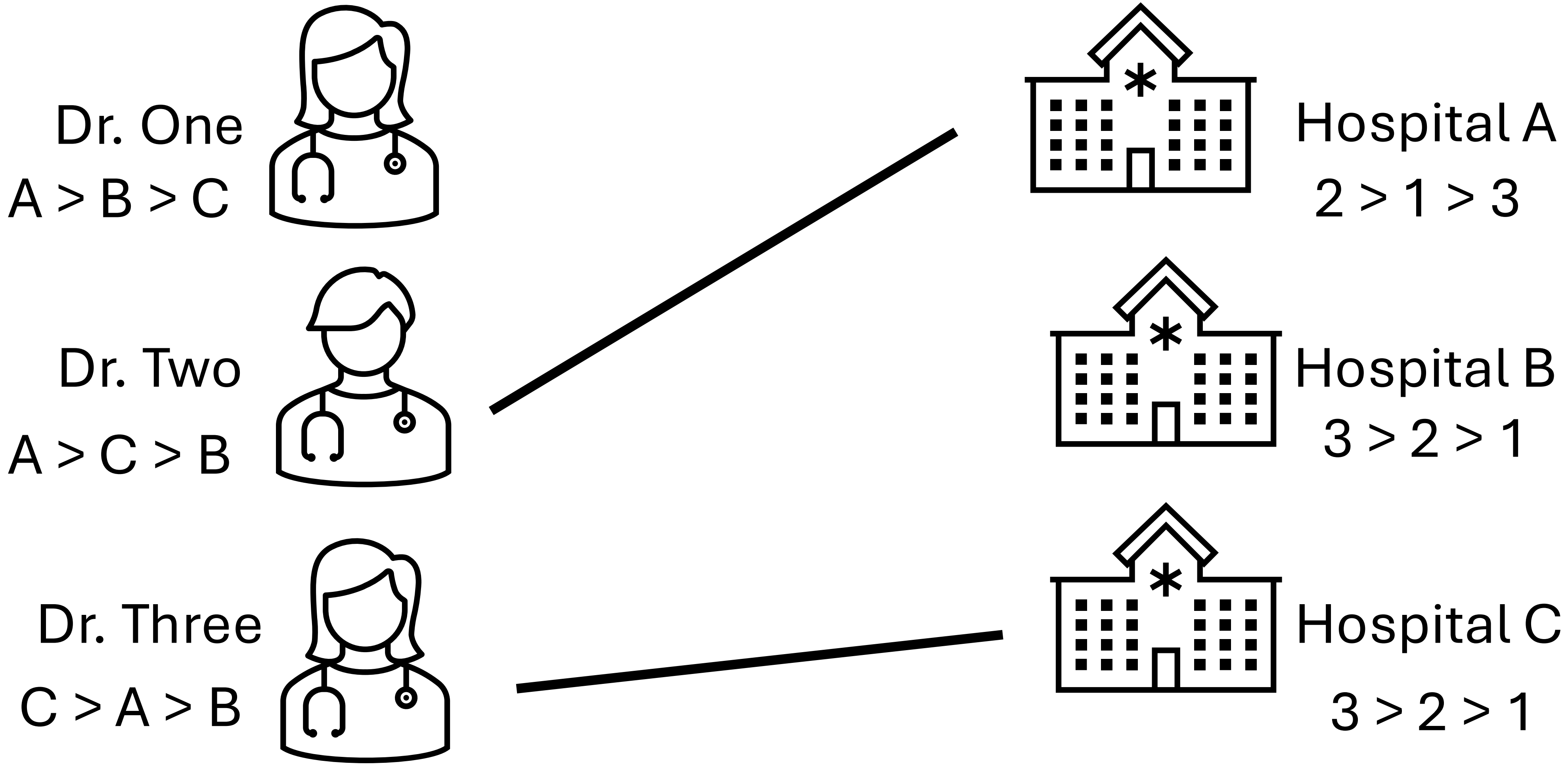
Resident Assignments



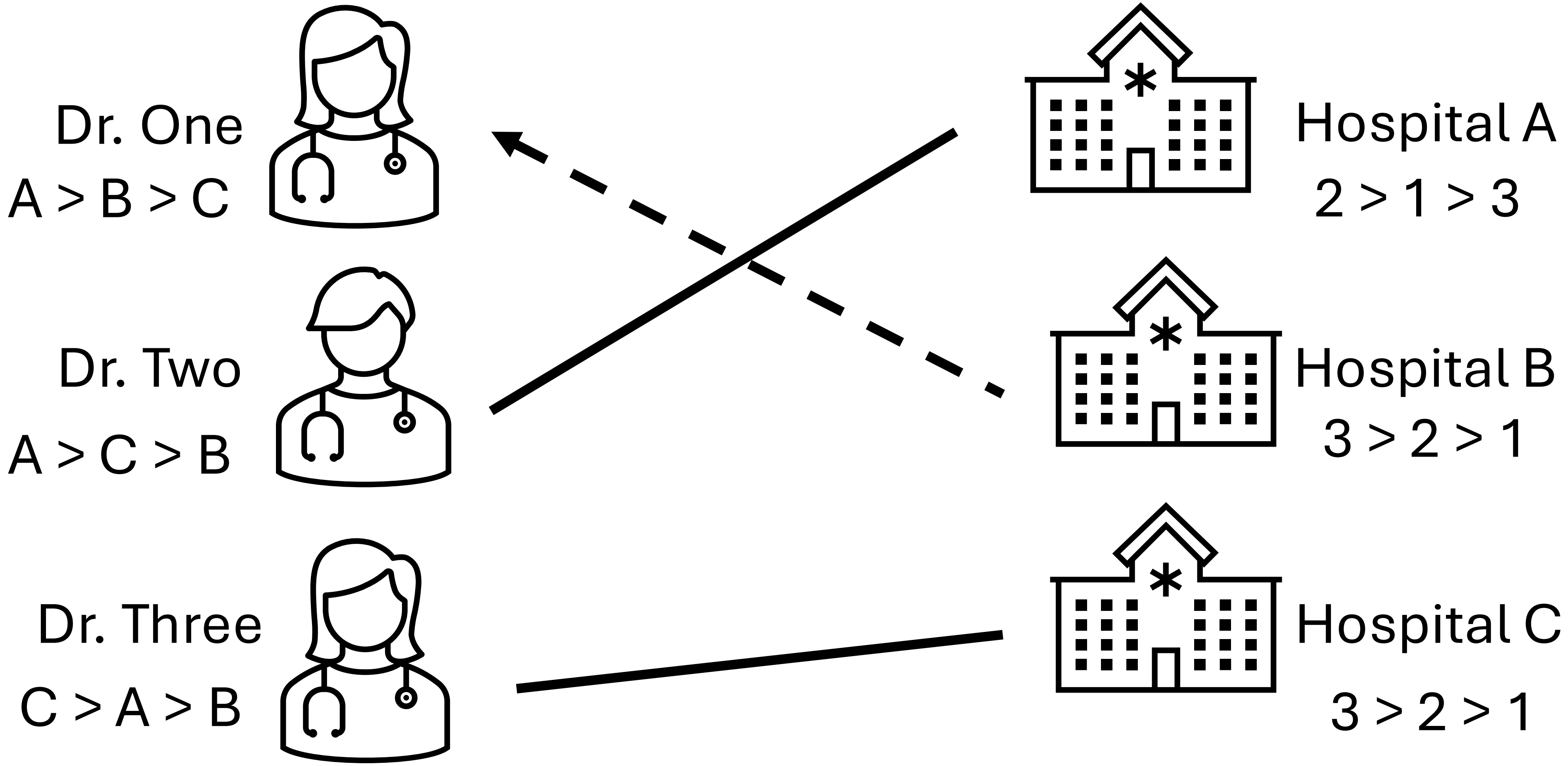
Resident Assignments



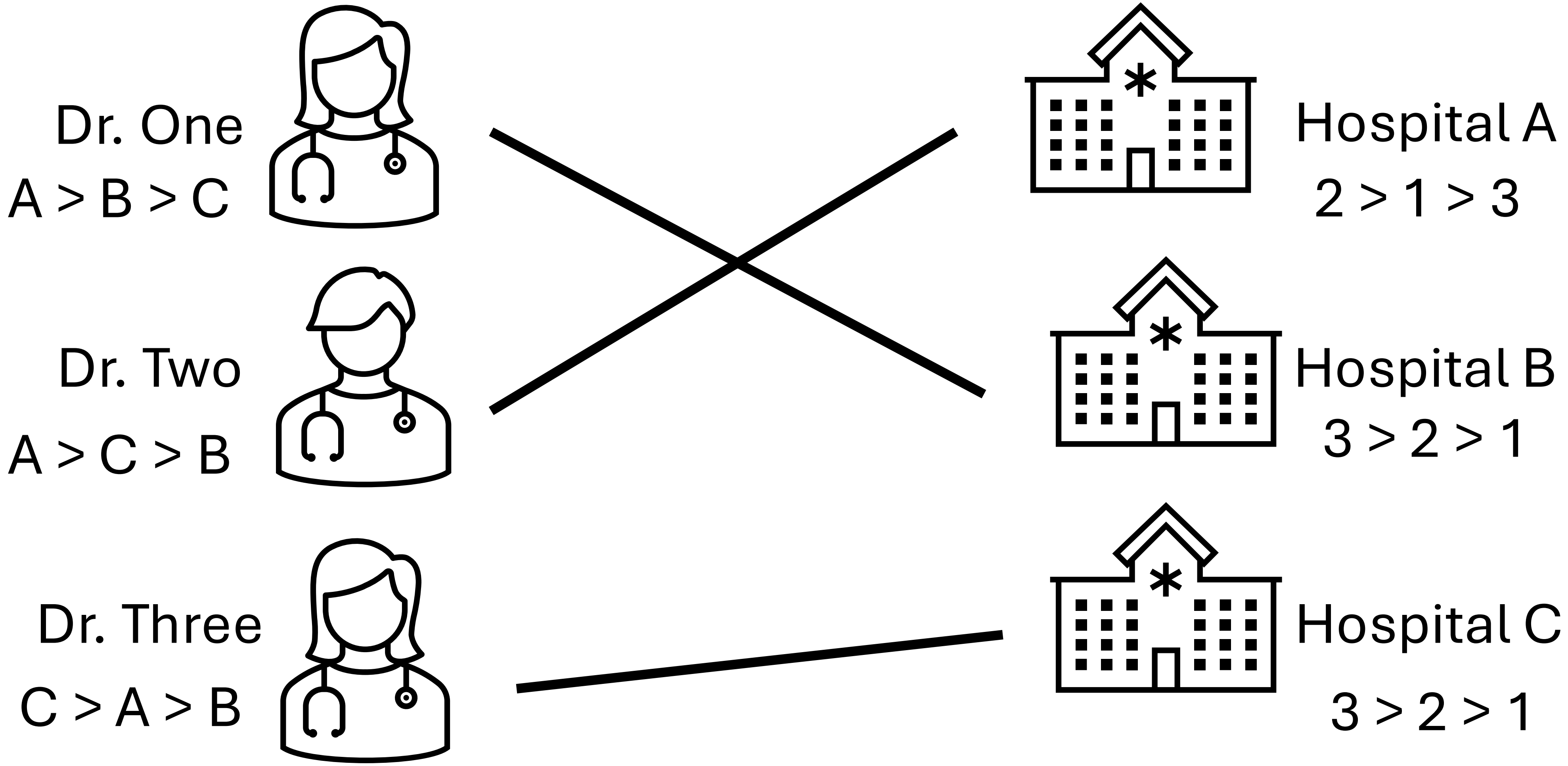
Resident Assignments



Resident Assignments



Resident Assignments



Q: Does a Stable Matching Always Exist?

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

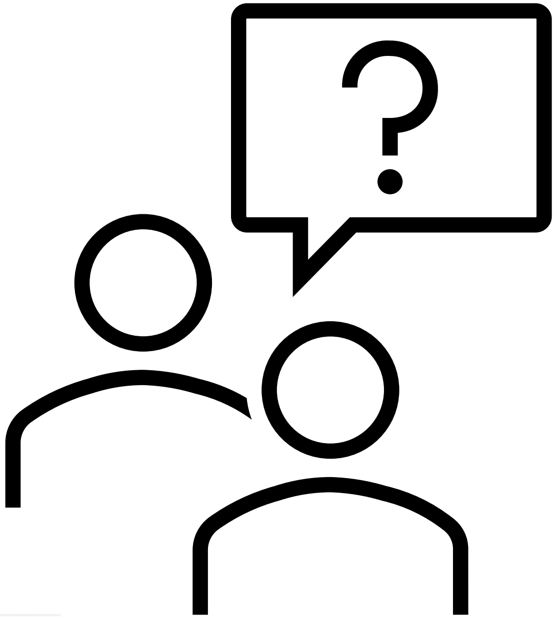
ELSE IF (s prefers h to current partner h')

Replace h'–s with h–s in matching M.

ELSE

s rejects h.

RETURN stable matching M.



Q: Does a Stable Matching Always Exist?

Observation 1: [Hint: What can we say about the order of proposals made by hospitals?]

Observation 2: [Hint: What can we say about a student's status after they've been matched?]

GALE–SHAPLEY (*preference lists for hospitals and students*)

```
INITIALIZE M to empty matching.
WHILE (some hospital h is unmatched and hasn't proposed to every student)
   s ← first student on h's list to whom h has not yet proposed.
   IF (s is unmatched)
      Add h–s to matching M.
   ELSE IF (s prefers h to current partner h′)
      Replace h′–s with h–s in matching M.
   ELSE
      s rejects h.

RETURN stable matching M.
```

Q: Does a Stable Matching Always Exist?

Observation 1: Hospitals propose in decreasing order of preference.

Observation 2: Once a student is matched, the student never becomes less happy.

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

  $s \leftarrow$ first student on h's list to whom h has not yet proposed.

  IF (s is unmatched)

    Add h–s to matching M.

  ELSE IF (s prefers h to current partner h')

    Replace h'–s with h–s in matching M.

  ELSE

    s rejects h.

RETURN stable matching M.

Claim Time!

Claim: The algorithm terminates after at most __ iterations of the `WHILE` loop.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

Replace h'–s with h–s in matching M.

ELSE

s rejects h.

Claim Time!

Claim: The algorithm terminates after at most n^2 iterations of the `WHILE` loop.

```
WHILE (some hospital h is unmatched and hasn't proposed to every student)
  s ← first student on h's list to whom h has not yet proposed.
  IF (s is unmatched)
    Add h–s to matching M.
  ELSE IF (s prefers h to current partner h′)
    Replace h′–s with h–s in matching M.
  ELSE
    s rejects h.
```

Proof Idea Time!

Claim: The algorithm terminates after at most n^2 iterations of the `WHILE` loop.

Proof (Idea): In each iteration, a hospital will propose to a different student. There are only n^2 possible pairings.

Proof Time!

Claim: The algorithm terminates after at most n^2 iterations of the `WHILE` loop.

Proof (Ideas): In each iteration, a hospital will propose to a different student. There are only n^2 possible pairings.

Proof: In each iteration, a hospital proposes to a new student. Thus, there are at most n^2 possible proposals since that is the number of possible hospital and student pairings.

Q: Are we done?

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M ← empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

s ← first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

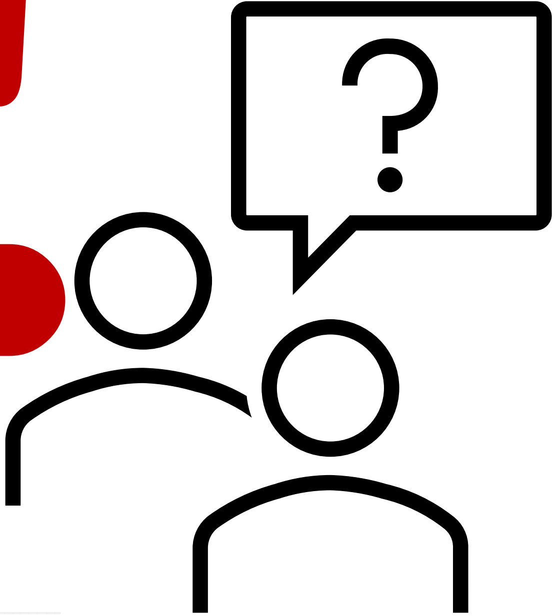
Replace h'–s with h–s in matching M.

ELSE

s rejects h.

RETURN stable matching M.

NO!



Claim Time!

Claim: The algorithm always outputs a matching.

```
WHILE (some hospital h is unmatched and hasn't proposed to every student)
  s ← first student on h's list to whom h has not yet proposed.
  IF (s is unmatched)
     Add h–s to matching M.
  ELSE IF (s prefers h to current partner h′)
     Replace h′–s with h–s in matching M.
  ELSE
     s rejects h.
```

Proof Idea Time!

Claim: The algorithm always outputs a matching.

Proof (Idea): Hospitals only ask to match when they are unmatched. Students only agree if they are unmatched or if they like the hospital better and drop the old hospital.

Proof Time!

Claim: The algorithm always outputs a matching.

Proof (Idea): Hospitals only ask to match when they are unmatched. Students only agree if they are unmatched or if they like the hospital better and drop the old hospital.

Proof: A hospital will only propose if it is unmatched. Hence, the degree of each hospital is at most one. A student will only stay matched with their most preferred hospital. Hence, the degree of each student is at most one.

Q: Are we done?

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIALIZE M to empty matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

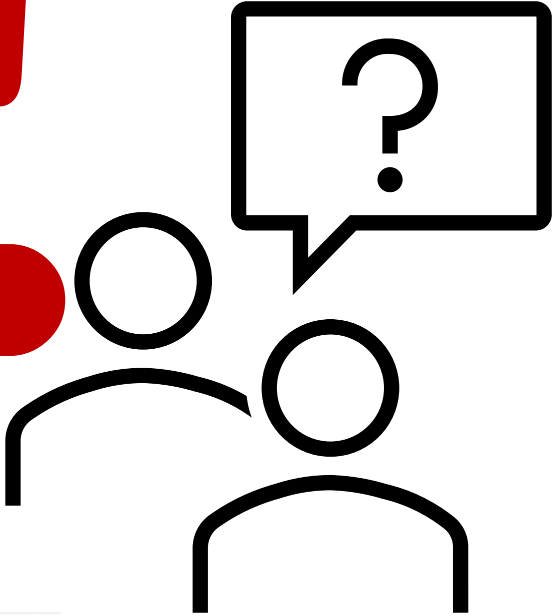
Replace h'–s with h–s in matching M.

ELSE

s rejects h.

RETURN stable matching M.

NO!



Claim Time!

Claim: The algorithm always outputs a perfect matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

$s \leftarrow$ first student on h's list to whom h has not yet proposed.

IF (s is unmatched)

Add h–s to matching M.

ELSE IF (s prefers h to current partner h')

Replace h'–s with h–s in matching M.

ELSE

s rejects h.

Proof Idea Time!

Claim: The algorithm always outputs a perfect matching.

Proof (Idea): We can show that every hospital is matched with a student. We know that each student is matched with at most one hospital. Since there are the same number, we could then conclude that the matching is perfect.

Subclaim Time!

Subclaim: The algorithm always outputs a matching such that each hospital is matched to a student.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

  $s \leftarrow$ first student on h's list to whom h has not yet proposed.

  IF (s is unmatched)

    Add h–s to matching M.

  ELSE IF (s prefers h to current partner h')

    Replace h'–s with h–s in matching M.

  ELSE

    s rejects h.

Subclaim Time!

Subclaim: The algorithm always outputs a matching such that each hospital is matched to a student.

Proof: We proceed with a proof by contradiction.

- Suppose to the contrary that….

Subclaim Time!

Subclaim: The algorithm always outputs a matching such that each hospital is matched to a student.

Proof: We proceed with a proof by contradiction.

- Suppose to the contrary that the algorithm terminates and there is a hospital h that is unmatched.

Subclaim Time!

Subclaim: The algorithm always outputs a matching such that each hospital is matched to a student.

Proof: We proceed with a proof by contradiction.

- Suppose to the contrary that the algorithm terminates and there is a hospital h that is unmatched.
- Then since the algorithm always returns a matching, there must exist a student s who is also not matched.

Subclaim Time!

Proof: We proceed with a proof by contradiction.

- Suppose to the contrary that the algorithm terminates and there is a hospital h that is unmatched.
- Then since the algorithm always returns a matching, there must exist a student s who is also not matched.
- If h proposed to s then we know from an earlier observation that s would still be matched.

Subclaim Time!

Proof: We proceed with a proof by contradiction.

- Suppose to the contrary that the algorithm terminates and there is a hospital h that is unmatched.
- Then since the algorithm always returns a matching, there must exist a student s who is also not matched.
- If h proposed to s then we know from an earlier observation that s would still be matched.
- If h did not propose to s then the while loop would not have exited.

Subclaim Time!

Proof: We proceed with a proof by contradiction.

- Suppose to the contrary that the algorithm terminates and there is a hospital h that is unmatched.
- Then since the algorithm always returns a matching, there must exist a student s who is also not matched.
- If h proposed to s then we know from an earlier observation that s would still be matched.
- If h did not propose to s then the while loop would not have exited.
- In all cases we have a contradiction and thus, the original assumption must be wrong.

Subclaim Time!

Subclaim: The algorithm always outputs a matching such that each student is matched to a hospital.

Proof:

Subclaim Time!

Subclaim: The algorithm always outputs a matching such that each hospital is matched to a student.

Proof: By the previous subclaim, we know that all n hospitals get matched to students. We also know by a previous claim that no hospital is matched to more than one student. Hence, each student must be matched to a hospital.

Big Proof Time!

Claim: The algorithm always outputs a Prefect matching.

Proof (Idea): We can show that every hospital is matched with a student. We know that each student is matched with at most one hospital. Since there are the same number, we could then conclude that the matching is perfect.

Proof: By both subclaims we know that the matching output by the algorithm matches every student and every hospital. saturates

Q: Are we done?

GALE–SHAPLEY (*preference lists for hospitals and students*)

INITIAL mpty matchin

WHILE (some ospital h is matched and has t proposed to every st nt)

s ← first s dent on h' ist to whom h has no et proposed.

(s is un tched)

dd h– o matchi M.

ELS F (s efers h to urrent partner h')

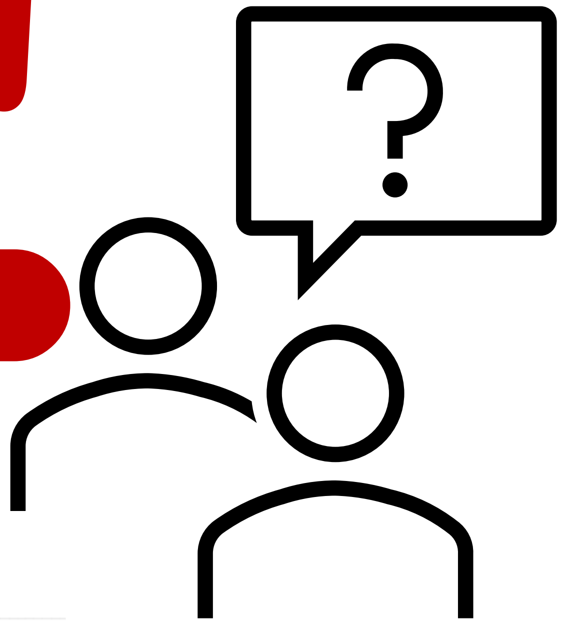
Rep ce '–s with h–s matching M.

ELSE

s rejects h.

RETURN stable matching M.

NO!



Claim Time!

Claim: The algorithm always outputs a stable matching.

WHILE (some hospital h is unmatched and hasn't proposed to every student)

  $s \leftarrow$ first student on h's list to whom h has not yet proposed.

  IF (s is unmatched)

    Add h–s to matching M.

  ELSE IF (s prefers h to current partner h')

    Replace h'–s with h–s in matching M.

  ELSE

    s rejects h.

Proof Idea Time!

Claim: The algorithm always outputs a stable matching.

Proof (Ideas):

Proof Idea Time!

Claim: The algorithm always outputs a stable matching.

Proof (Ideas):

1. Show that every unmatched pair is not unstable.
2. **Show that the existence of an unstable pair would be a contradiction.**

Proof Time!

Proof: We proceed with a proof by contradiction.

- **Suppose to the contrary that…**

Proof Time!

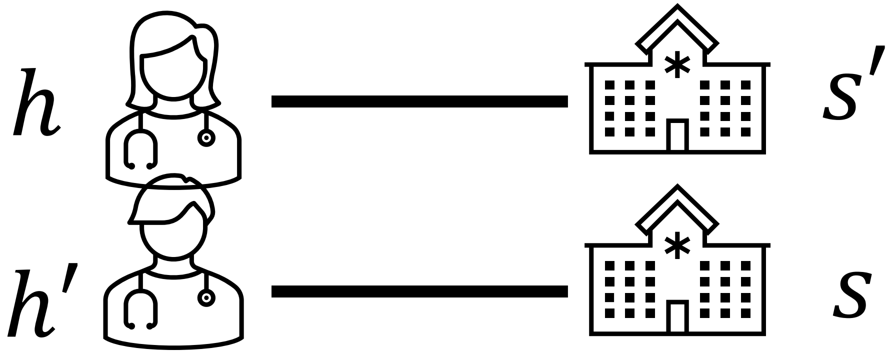
Proof: We proceed with a proof by contradiction.

- Suppose to the contrary that the matching M returned was unstable. Then there must exist an unstable pair (h, s).
  - Since we know that matching returned is perfect, we know there exists hospital h' and student s' such that $(h, s') \in M$ and $(h', s\ ) \in M$.

Proof Time!

Proof: We proceed with a proof by contradiction.

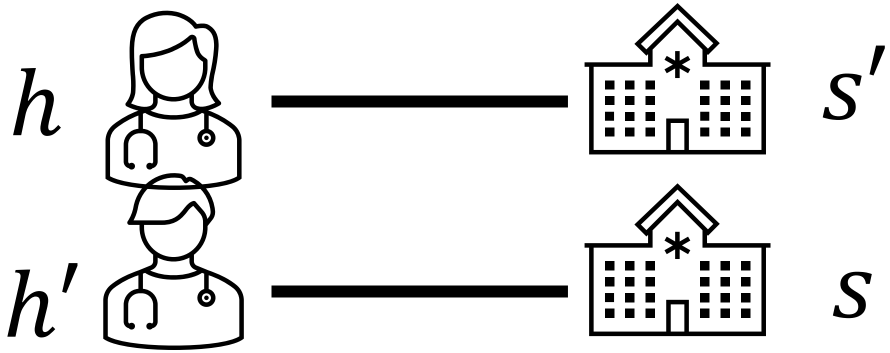
- Suppose to the contrary that the matching M returned was unstable. Then there must exist an unstable pair (h, s).
  - Since we know that matching returned is perfect, we know there exists hospital h' and student s' such that $(h, s') \in M$ and $(h', s) \in M$.



Proof Time!

Proof: …

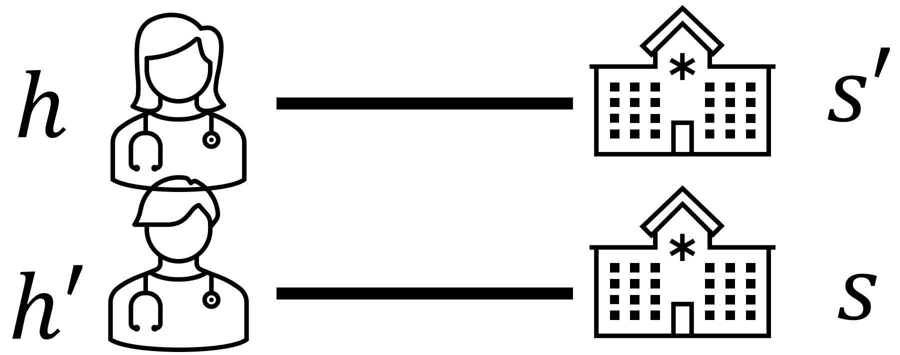
- Suppose h did propose to s at some point in the algorithm.



Proof Time!

Proof: …

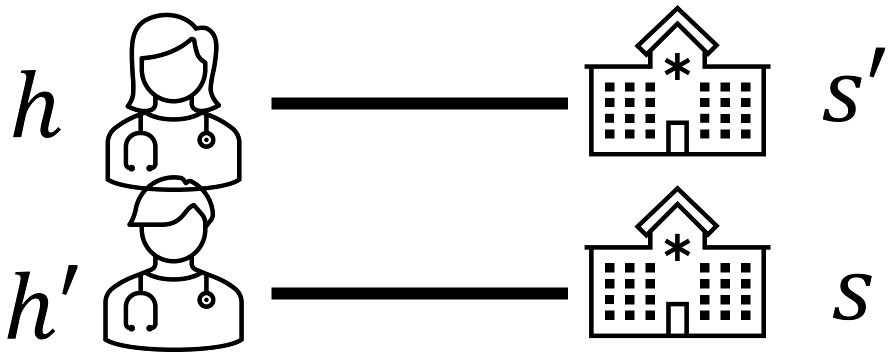
- Suppose h did propose to s at some point in the algorithm.
  - Then s must have at some point rejected h and thus, must prefer h' to h. (Not Unstable)



Proof Time!

Proof: …

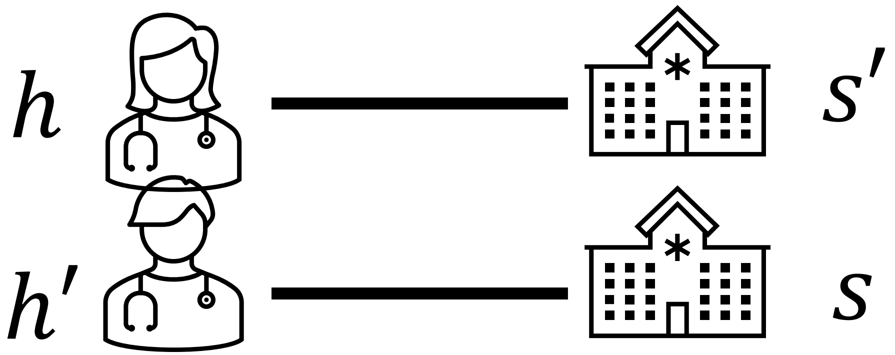
- Suppose h did not propose to s at some point in the algorithm.



Proof Time!

Proof: …

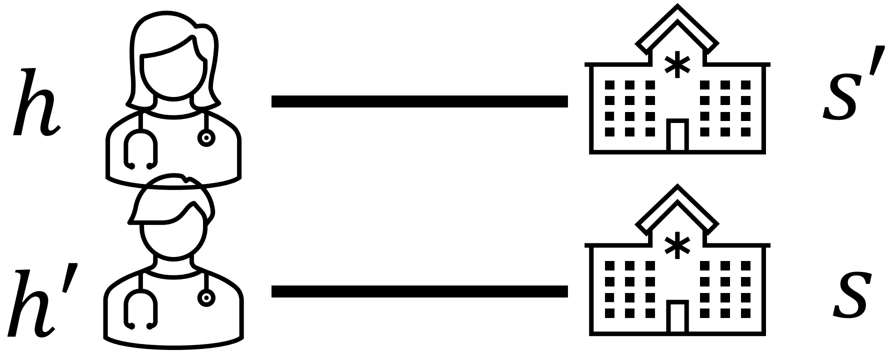
- Suppose h did not propose to s at some point in the algorithm.
  - Then h must have proposed to s' first and thus, must prefer s' to s. (Not Unstable)



Proof Time!

Proof: …

- In all cases we have concluded that if there is an unstable pair then the pair is can't be unstable. This is a contradiction and thus there can be no unstable pairs.



Proof Idea Time!

Claim: The algorithm always outputs a stable matching.

Proof (Ideas):

1. **Show that every unmatched pair is not unstable.**
2. Show that the existence of an unstable pair would be a contradiction.