Example Problem: Stable Matching

CS 4104: Data and Algorithm Analysis

Yoseph Berhanu Alebachew

May 11, 2025

Virginia Tech

Table of contents

The Problem

Origin

- Originated in 1962 by David Gale and Lloyd Shapley
- They wanted to implement a self-enforcing college admissions process
- This is also called Gale-Shapley Matching
- National Resident Matching Program had been using a very similar procedure

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- What is stable?

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- What is stable?
  - For every applicants $a \in A$ who is not scheduled to work for $e \in E$, at least one of the following is true

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- What is stable?
    - For every applicants $a \in A$ who is not scheduled to work for $e \in E$, at least one of the following is true
        - e prefers every one of its accepted applicants to a; or

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- What is stable?
    - For every applicants $a \in A$ who is not scheduled to work for $e \in E$, at least one of the following is true
        - e prefers every one of its accepted applicants to a; or
        - a prefers her current situation over working for employer e.

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- What is stable?
  - For every applicants $a \in A$ who is not scheduled to work for $e \in E$, at least one of the following is true
    - e prefers every one of its accepted applicants to a; or
    - a prefers her current situation over working for employer e.
  - Conversely, a matching of applicants $(a, e) and (a', e')$ for $e, e' \in E and a, a' \in A$ is unstable if both of the following hold

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- What is stable?
  - For every applicants $a \in A$ who is not scheduled to work for $e \in E$, at least one of the following is true
    - e prefers every one of its accepted applicants to a; or
    - a prefers her current situation over working for employer e.
  - Conversely, a matching of applicants $(a, e) and (a', e')$ for $e, e' \in E and a, a' \in A$ is unstable if both of the following hold
    - e prefers a' over a

Goal

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- What is stable?
  - For every applicants $a \in A$ who is not scheduled to work for $e \in E$, at least one of the following is true
    - e prefers every one of its accepted applicants to a; or
    - a prefers her current situation over working for employer e.
  - Conversely, a matching of applicants $(a, e) and (a', e')$ for $e, e' \in E and a, a' \in A$ is unstable if both of the following hold
    - e prefers a' over a
    - a' prefers e over e'

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.
- A set of ordered pairs $S \in (MxW)$ is

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.
- A set of ordered pairs $S \in (MxW)$ is
  - A

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.
- A set of ordered pairs $S \in (MxW)$ is
    - A **matching**, if each member of M and each member of W appears in at most *most one* pair in S.

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.
- A set of ordered pairs $S \in (MxW)$ is
  - A **matching**, if each member of M and each member of W appears in at most *most one* pair in S.
  - A **perfect matching**, if each member of M and each member of W appears in *exactly one* pair in S.

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.
- A set of ordered pairs $S \in (MxW)$ is
  - A **matching**, if each member of M and each member of W appears in at *most one* pair in S.
  - A **perfect matching**, if each member of M and each member of W appears in *exactly one* pair in S.
  - **Stable**, if one of the following holds for every pair $a, e \in S$

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.
- A set of ordered pairs $S \in (MxW)$ is
  - A **matching**, if each member of M and each member of W appears in at *most one* pair in S.
  - A **perfect matching**, if each member of M and each member of W appears in *exactly one* pair in S.
  - **Stable**, if one of the following holds for every pair $a, e \in S$
    - e prefers every one of its accepted applicants to a; or

Reformulation

- So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women.
- Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$.
- A set of ordered pairs $S \in (MxW)$ is
    - A **matching**, if each member of M and each member of W appears in at *most one* pair in S.
    - A **perfect matching**, if each member of M and each member of W appears in *exactly one* pair in S.
    - **Stable**, if one of the following holds for every pair $a, e \in S$
        - e prefers every one of its accepted applicants to a; or
        - a prefers her current situation over working for employer e.

Example: Input 1

- Each man ranks all the women in order of preference.
- Each woman ranks all the men in order of preference.
- Each person uses all ranks from 1 to n, i.e., no ties, no incomplete lists.

Example: Input 1

- Each man ranks all the women in order of preference.
- Each woman ranks all the men in order of preference.
- Each person uses all ranks from 1 to n, i.e., no ties, no incomplete lists.

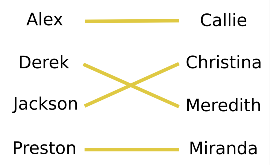
Men	**1**	**2**	**3**	**4**
Alex	Callie	Christina	Meredith	Miranda
Derek	Meredith	Miranda	Christina	Callie
Jackson	Meredith	Miranda	Christina	Callie
Preston	Christina	Miranda	Callie	Meredith

Example: Input 2

Women	1	2	3	4
Callie	Alex	Derek	Jackson	Preston
Christina	Derek	Preston	Jackson	Alex
Meredith	Derek	Jackson	Preston	Alex
Miranda	Derek	Jackson	Alex	Preston

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

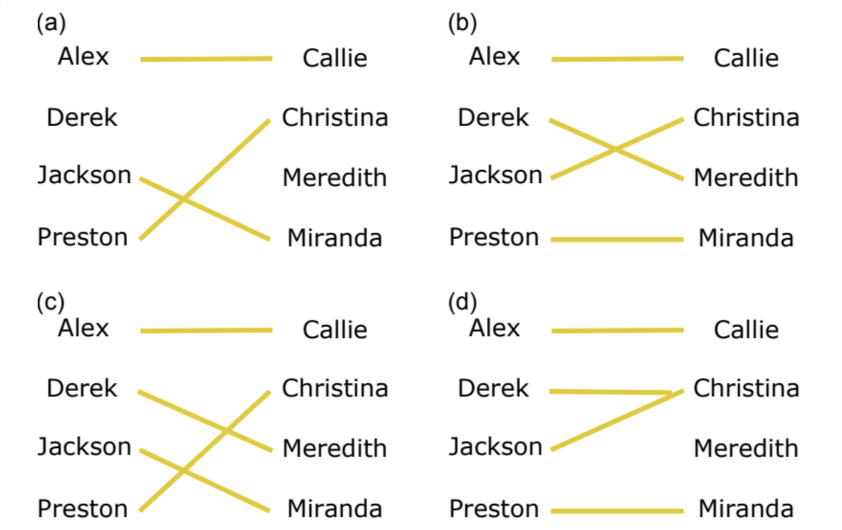


- **Matching**: each man is paired with ≤ 1 woman and vice versa.
- **Perfect matching**: each man is paired with exactly one woman and vice versa.

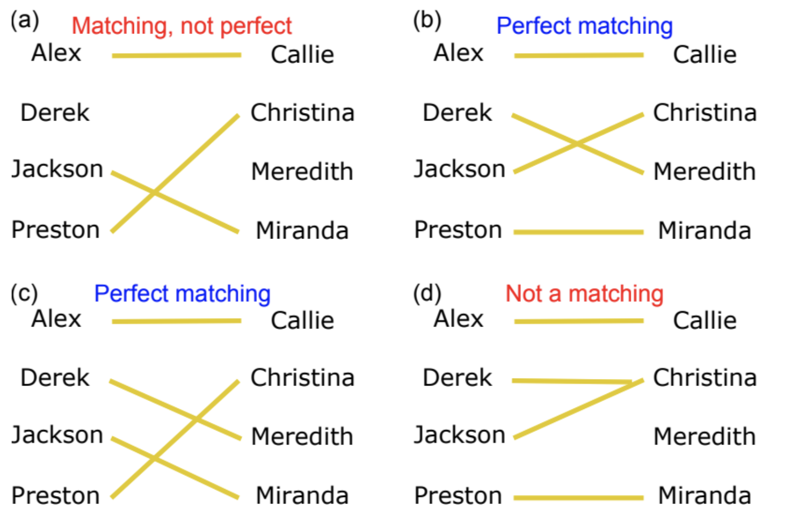
Note

"Perfect": only means one-to-one mapping, not that people are happy with matches or its stable.

Other "matching"



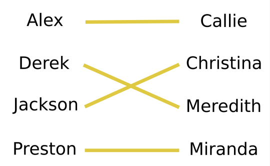
Other "matching"



Our Matching

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

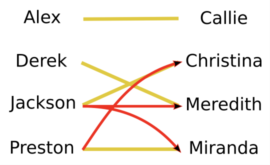


Are there problems with this matching?

Our Matching

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

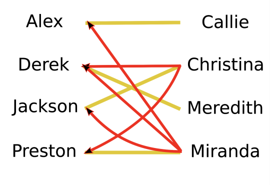


Are there problems with this matching?

Our Matching

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

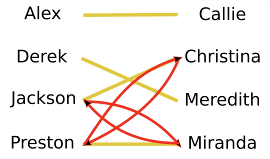


Are there problems with this matching?

Our Matching

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

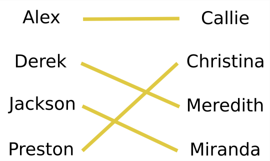


Rogue couple: a man and a woman who are not matched but prefer each other to their current partners.

Stable Matching

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

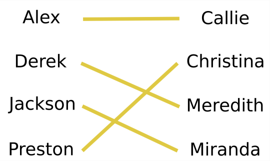


Stable matching: A perfect matching without any rogue couples.

Stable Matching

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4



Stable matching: A perfect matching without any rogue couples.

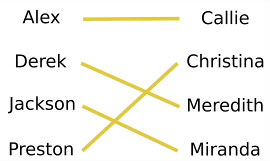
Questions

1. Given preferences for every woman and every man, does a stable matching exist?

Stable Matching

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4



Stable matching: A perfect matching without any rogue couples.

Questions

1. Given preferences for every woman and every man, does a stable matching exist?
2. If it does, can we compute it? How fast?

The Algorithm

Objective

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- We will start with a simple version of the problem
    - Let's assume there are only two men and two women
    - What are the possible permutations of preferences?

Objective

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- We will start with a simple version of the problem
  - Let's assume there are only two men and two women
  - What are the possible permutations of preferences?
    - $M_1[W_1, W_2]$; $M_1[W_2, W_1]$
      $M_2[W_1, W_2]$; $M_2[W_2, W_1]$

Objective

- Given of preferences among employers (E) and applicants (A), can we come up with a stable matching?
- We will start with a simple version of the problem
  - Let's assume there are only two men and two women
  - What are the possible permutations of preferences?
    - $M_1[W_1, W_2]$; $M_1[W_2, W_1]$
      $M_2[W_1, W_2]$; $M_2[W_2, W_1]$
    - $W_1[M_1, M_2]$; $W_1[M_2, M_1]$
      $W_2[M_1, M_2]$; $W_2[M_2, M_1]$

Possible Matching

Example

	W1	W2
M1	1	2
M2	1	2

Stable Matching

Possible Matching

Example

	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	1	2
W2	1	2

Stable Matching

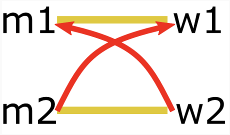
Possible Matching

Example

	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	1	2
W2	1	2

Stable Matching



Possible Matching

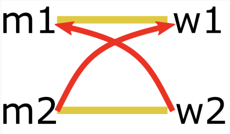
Example

	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	1	2
W2	1	2

	W1	W2
M1	1	2
M2	1	2

Stable Matching



Possible Matching

Example

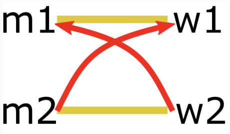
	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	1	2
W2	1	2

	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	2	1
W2	2	1

Stable Matching



Possible Matching

Example

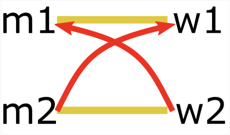
	W1	W2
M1	1	2
M2	1	2

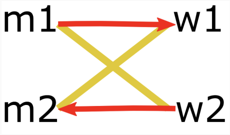
	M1	M2
W1	1	2
W2	1	2

	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	2	1
W2	2	1

Stable Matching





Possible Matching

Example

	W1	W2
M1	1	2
M2	1	2

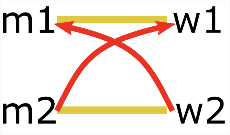
	M1	M2
W1	1	2
W2	1	2

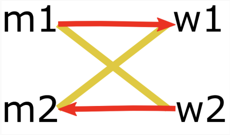
	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	2	1
W2	2	1

	W1	W2
M1	1	2
M2	2	1

Stable Matching





Possible Matching

Example

	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	1	2
W2	1	2

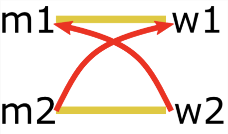
	W1	W2
M1	1	2
M2	1	2

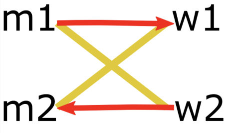
	M1	M2
W1	2	1
W2	2	1

	W1	W2
M1	1	2
M2	2	1

	M1	M2
W1	2	1
W2	1	2

Stable Matching





Possible Matching

Example

	W1	W2
M1	1	2
M2	1	2

	M1	M2
W1	1	2
W2	1	2

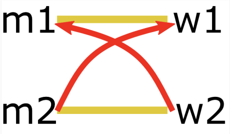
	W1	W2
M1	1	2
M2	1	2

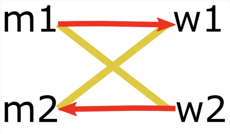
	M1	M2
W1	2	1
W2	2	1

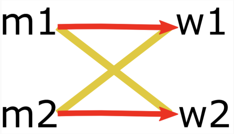
	W1	W2
M1	1	2
M2	2	1

	M1	M2
W1	2	1
W2	1	2

Stable Matching







Challenge

Can you create an example that does not have a stable matching?

GS Algorithm

```
Initially all men and women are free
while there is a man m who is free and hasn't proposed to ev
    Choose such a man m
    m proposes to the highest-ranked woman in m's preference
    if w is free then
        (m, w) become engaged -> Add (m,w) from S
    else if w is engaged to m' but prefers m to m' then
        m' becomes free -> Delete (m',w) from S
        (m, w) become engaged -> Add (m,w) from S
    else
        m remains free
return the set S of engaged pairs
```

Does it work ?

The Algorithm

- Each man proposes to each woman, in decreasing order of preference.
- Woman accepts if she is free or prefers new prospect to current fiance.

What can go wrong?

- Does the algorithm even terminate?
- If it does, how long does the algorithm take to run?
- If it does, is S a perfect matching? A stable matching ?

Observations

- Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa.
- Man's status: Can alternate between being free and being engaged.
- Woman's status: Remains engaged after first proposal.
- Ranking of a man's partner: Remains the same or goes down.
- Ranking of a woman's partner: Can never go down.

Proof?

Can we prove that that GS algorithms produces a **terminates** with **stable matching**

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
  - Number of free men?

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
  - Number of free men?
  - Number of free women?

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
  - Number of free men?
  - Number of free women?
    - No, since both can remain unchanged in an iteration.

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
  - Number of free men?
  - Number of free women?
    - No, since both can remain unchanged in an iteration.
  - Number of proposals made after k iterations?

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
  - Number of free men?
  - Number of free women?
    - No, since both can remain unchanged in an iteration.
  - Number of proposals made after k iterations?
    - Must increase by one in each iteration.

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
    - Number of free men?
    - Number of free women?
        - No, since both can remain unchanged in an iteration.
    - Number of proposals made after k iterations?
        - Must increase by one in each iteration.
    - How many total proposals can be made?

Proof: Algorithm Terminates/Runtime

- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
  - Number of free men?
  - Number of free women?
    - No, since both can remain unchanged in an iteration.
  - Number of proposals made after k iterations?
    - Must increase by one in each iteration.
  - How many total proposals can be made?
    - n^2

Proof: Algorithm Terminates/Runtime

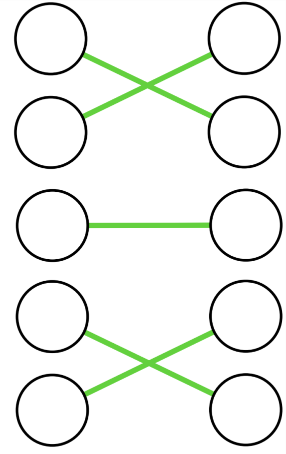
- Is there some quantity that we can use the measure the progress of the algorithm in each iteration?
  - Number of free men?
  - Number of free women?
    - No, since both can remain unchanged in an iteration.
  - Number of proposals made after k iterations?
    - Must increase by one in each iteration.
  - How many total proposals can be made?
    - n^2
    - The algorithm must terminate in n^2 iterations

Correctness Proof: Matching Computed is Perfect

- Suppose the set S of pairs returned by the GS algorithm is not perfect.
- S is a matching. Therefore, there must be at least one free man m.
- m has proposed to all the women (since algorithm terminated).
- Therefore, each woman must be engaged (since she remains engaged after the first proposal to her).
- Therefore, all men must be engaged, contradicting the assumption that m is free.
- Proof that matching is perfect relies on
  - proof that the algorithm terminated and
  - the very specific termination condition.

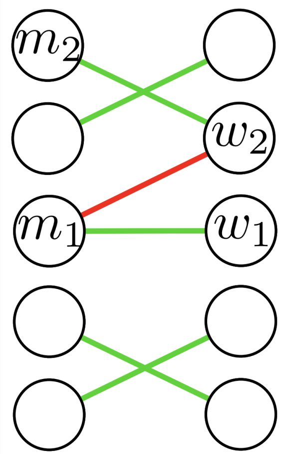
Correctness Proof: Matching Computed is Stable

Perfect matching S returned by algorithm



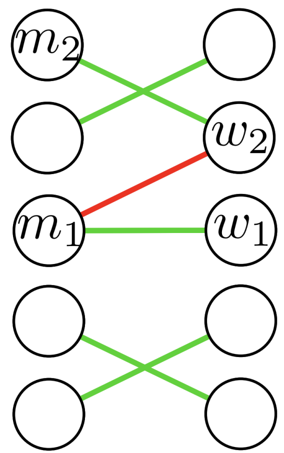
Correctness Proof: Matching Computed is Stable

- Not stable: m_1 paired with w_1 but prefers w_2;
- w_2 paired with m_2 but prefers m_1



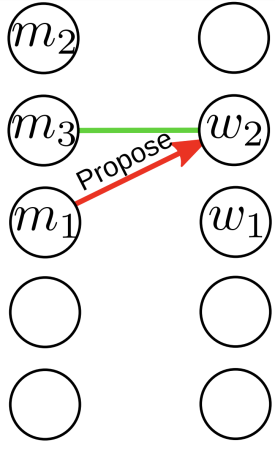
Correctness Proof: Matching Computed is Stable

m_1 proposed to w_2 before proposing to w_1



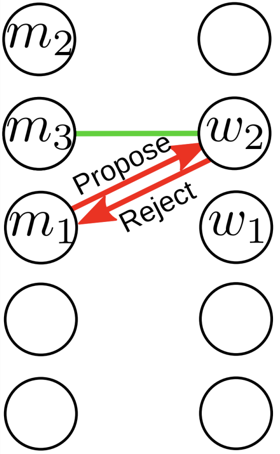
Correctness Proof: Matching Computed is Stable

Remember: What happened when m_1 proposed to w_2?



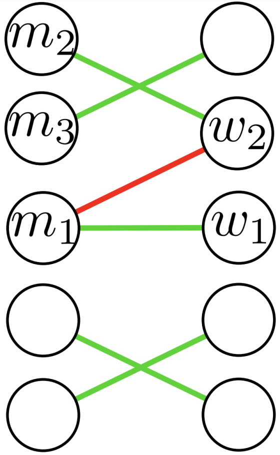
Correctness Proof: Matching Computed is Stable

Case 1: w_2 rejected w_1 because she preferred current partner m_3?



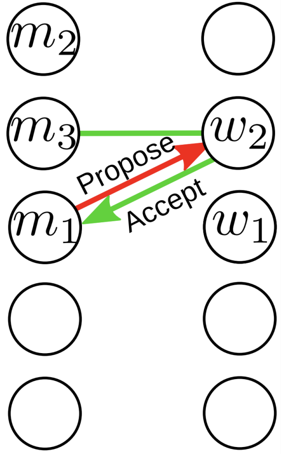
Correctness Proof: Matching Computed is Stable

Case 1: At termination w_2 must prefer her final partner m_2 to m_3.
Contradicts consequence of instability: m_2 prefers m_1 to m_2?



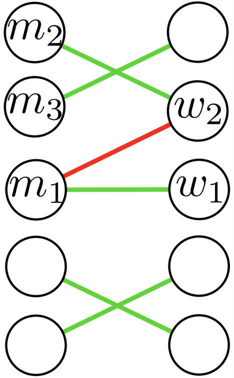
Correctness Proof: Matching Computed is Stable

Case 2: w_2 accepted m_1 because she had no partner or preferred m_1 to current partner $m3$?



Correctness Proof: Matching Computed is Stable

Case 2: By instability, we know w_2 prefers m_1 to m_2. But at termination, w_2 is matched with m_2, which contradicts property that a woman switches only to a better match.



Correctness Proof: In Words

- Suppose S is not stable,
  - there are two pairs (m_1, w_1) and (m_2, w_2) in S such that m_1 prefers w_2 to w_1 and w_2 prefers m_1 to m_2.
- m_1 must have proposed to w_2 before w_1
- At that stage w_2 must have rejected m_1
  - otherwise, the algorithm would pair m_1 and w_2,
  - would prevent the pairing of m_2 and w_2 in a later iteration of the algorithm.
- When w_2 rejected m_1, she must have been paired with some man, say m_3, whom she prefers to m_1.
- Since m_2 is paired with w_2 at termination, w_2 must prefer to m_2 to m_3 or $m_2 = m_3$,
  - contradicts our conclusion that w_2 prefers m1 to m_2.

Variants: Hospitals and residents

- Multiple residents
  - Each hospital can take multiple residents.
  - Modification of Gale-Shapley algorithm works.
  - Some residents may not be matched.
  - Some hospitals may not fill quota.
- Hospitals and residents with couples
  - Each hospital can take multiple residents.
  - A couple must be assigned together, either to the same hospital or to a specific pair of hospitals chosen by the couple
  - **NP-Complete**

Variants: Others

- Stable roommates
    - There is only one pool of people
    - Stable matching may not exist.
    - Irving's algorithm; more complex than Gale-Shapley.
- Complex preferences
    - Preferences may be incomplete or have ties or people may lie.
    - Several variants are NP-hard, even to approximate.

A Few More Problems

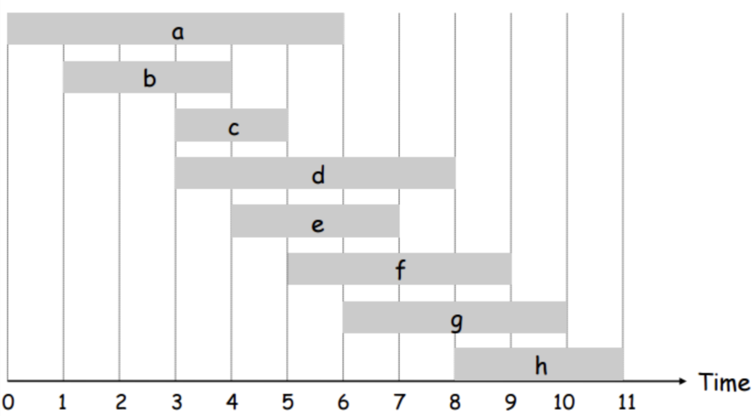
Interval Scheduling: Idea

- Say you have a resource to be scheduled for
  - It may be a lecture room, a supercomputer, or an electron microscope
- Many people request to use the resource for periods of time.
- A request takes the form:
  - Can I reserve the resource starting at time s, until time f ?
- We will assume that the resource can be used by at most one person at a time.
- A scheduler wants to accept a subset of these requests, rejecting all others, so that the accepted requests do not overlap in time.
- The goal is to maximize the number of requests accepted.

Interval Scheduling: Formally

- There will be n requests labeled $1, ..., n$
- Each request i specifying a start time s_i and a finish time f_i
- We have $s_i < f_i$ for all i
- Two requests i and j are compatible if the requested intervals do not overlap:
    - either request i is for an earlier time interval than request $j (f_i \leq s_j)$,
    - or request i is for a later time than request $j (f_j s_i)$.
- Generally that a subset A of requests is compatible if all pairs of requests $i, j \in A, i \neq j$ are compatible.
- The goal is to select a compatible subset of requests of maximum possible size.
- Interval Scheduling has a **Greedy Algorithm Solution**

Interval Scheduling: Visually



Visualization of Interval Scheduling [1]

[1] Image Credit: `https://stumash.github.io/Algorithm_Notes/greedy/intervals/intervals.html`

Weighted Interval Scheduling

- A modification to Interval Scheduling Problem
- Suppose more generally that each request interval i has an associated value, or weight, $v_i > 0$
  - We could picture this as the amount of money we will make from the i^th individual if we schedule his or her request.
- Our goal will be to find a compatible subset of intervals of maximum total value.
- The case in which $v_i = 1$ for each i is simply the basic Interval Scheduling Problem
- The appearance of arbitrary values changes the nature of the maximization problem quite a bit.

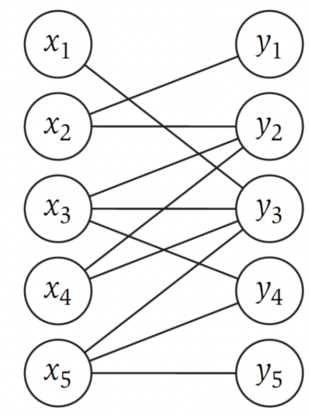
Weighted Interval Scheduling

- Consider, for example, that if v_1 exceeds the sum of all other v_i, then the optimal solution must include interval 1 regardless of the configuration of the full set of intervals.
- So any algorithm for this problem must be very sensitive to the values, and yet degenerate to a method for solving (unweighted) interval scheduling when all the values are equal to 1.
- There appears to be no simple greedy rule that walks through the intervals one at a time, making the correct decision in the presence of arbitrary values.
- Instead, we employ a technique, **dynamic programming**
- It builds up the optimal value over all possible solutions in a compact, tabular way that leads to a very efficient algorithm.

Bipartite Matching

- A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V.
- Bipartite matching involves finding a maximum matching, which is the largest subset of edges such that no two edges share a common vertex.
- Used in job assignments, network flows, and resource allocation.

Bipartite Matching



A bipartite graph [2]

[2] Image Credit: "Algorithm Design" Jon Kleinberg and Eva Tardos - Addison Wesley (2005)

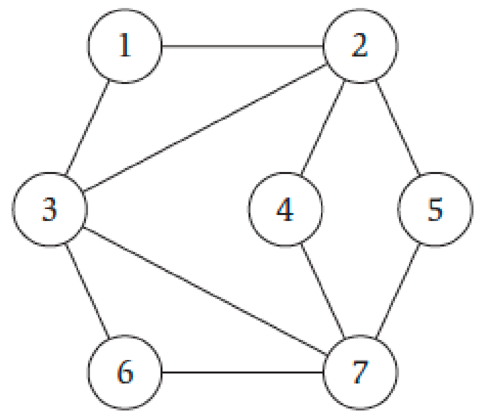
Bipartite Matching

- **Maximum matching** is the largest set of edges with no shared vertices.
- **Perfect matching** a matching that covers every vertex in the graph.
- Augmenting Path a path that can increase the size of the current matching.
- Algorithms for Bipartite Matching
  - Hungarian Algorithm: Efficient for finding maximum matching in bipartite graphs.
  - Hopcroft-Karp Algorithm: Improves performance for large bipartite graphs.
- Interval scheduling can be transformed into a bipartite matching problem by representing intervals as nodes in a bipartite graph.

Independent Set: The Problem

- Given a graph G = (V, E), we say a set of nodes S V is independent if no two nodes in S are joined by an edge.
- The Independent Set Problem is, then, the following: Given G, find an independent set that is as large as possible.
- The Independent Set Problem encodes any situation in which you are trying to choose from among a collection of objects and there are pairwise conflicts among some of the objects.
- Say you have n friends, and some pairs of them don't get along.
- How large a group of your friends can you invite to dinner if you don't want any interpersonal tensions?
- This is simply the largest independent set in the graph whose nodes are your friends, with an edge between each conflicting pair.

Independent Set: Example



A graph whose largest independent set has size 4 (1,4,5,6).[3]

[3] Image Credit: "Algorithm Design" Jon Kleinberg and Eva Tardos - Addison Wesley (2005)

Independent Set: Runtime

- No efficient algorithm is known for the Independent Set problem, and it is conjectured that no such algorithm exists.
- The solution we have is the obvious brute-force algorithm
- Once a solution is found, we can check if it is correct in polynomial time
- This is a group of problems called NP-Complete

Competitive Facility Location

- The Competitive Facility Location Problem is a strategic decision problem where companies compete to place their facilities (e.g., stores, warehouses) in a market.
- The goal is to maximize market share, profit, or another performance measure while considering the actions of competitors.
- Constraints to consider in location decisions:
    - Proximity to consumers to minimize transportation costs.
    - Legal and environmental regulations affecting feasible locations.
    - Spatial strategies to counteract competitors' locations.
- Solution
    - No efficient solution,
    - Not even an efficient way of check a solution
    - Heuristic methods
    - Approximation methods

Conclusion

- In this lecture we discussed
    - Stable Matching Problem
    - A greedy algorithm as a solution
    - Analysis of the proposed algorithm (less formal)
    - Correctness
    - Runtime complexity
- Next lecture
    - No class on Monday
    - Algorithm Analysis
    - Read Chapter 2 of the textbook
    - Lecture note will be provided as a reference

Questions?

Acknoladgment

Slide adaåpted from T. M. Murali with additional content from "Algorithm Design" Jon Kleinberg and Eva Tardos - Addison Wesley (2005)