Algorithm Design
Representative Problems

Prof. Dr. Brahim Hnich

March 11, 2013

---

1

---

1Slides based on those by Kevin Wayne. Copyright ©2005 Pearson-Addison Wesley. All rights reserved.
Representative Problems

- Stable Matching
Representative Problems

- Stable Matching
- Interval Scheduling
Representative Problems

- Stable Matching
- Interval Scheduling
- Weighted Interval Scheduling
Representative Problems

- Stable Matching
- Interval Scheduling
- Weighted Interval Scheduling
- Bipartite Matching
Representative Problems

- Stable Matching
- Interval Scheduling
- Weighted Interval Scheduling
- Bipartite Matching
- Independent Set
Stable Matching

- There are $n$ men and $n$ women
- There are $n$ men and $n$ women
- Each man has a preference list, so does the woman.
Stable Matching

- There are $n$ men and $n$ women
- Each man has a preference list, so does the woman.
- These lists have no ties.
Stable Matching

- There are $n$ men and $n$ women
- Each man has a preference list, so does the woman.
- These lists have no ties.
- Devise a system by which each of the $n$ men and $n$ women can end up getting married.
Similar Problems

- Given a set of colleges and students pair them. (Internship Company assignments)
- Given airlines and pilots, pair them.
- Given two images, pair the points belonging to the same point in 3D to extract depth from the two images.
- Dorm room assignments.
- Hospital residency assignments.
Similar Problems

▶ Given a set of colleges and students pair them. (Internship Company assignments)
▶ Given airlines and pilots, pair them.
Similar Problems

- Given a set of colleges and students pair them. (Internship Company assignments)
- Given airlines and pilots, pair them.
- Given two images, pair the points belonging to the same point in 3D to extract depth from the two images.
Given a set of colleges and students pair them. (Internship Company assignments)

Given airlines and pilots, pair them.

Given two images, pair the points belonging to the same point in 3D to extract depth from the two images.

Dorm room assignments.
Similar Problems

▶ Given a set of colleges and students pair them. (Internship Company assignments)
▶ Given airlines and pilots, pair them.
▶ Given two images, pair the points belonging to the same point in 3D to extract depth from the two images.
▶ Dorm room assignments.
▶ Hospital residency assignments
Definition
A Perfect matching is a matching in which everyone is matched monogamously.
Perfect matching?

Definition
A Perfect matching is a matching in which everyone is matched monogamously.

- Each man gets exactly one woman.
- Each woman gets exactly one man.
What is a good matching?

- Maximize the number of people who get their first match?
What is a good matching?

- Maximize the number of people who get their first match?
- Maximize the average satisfaction?
What is a good matching?

- Maximize the number of people who get their first match?
- Maximize the average satisfaction?
- Maximize the minimum satisfaction?
What is a good matching?

- Maximize the number of people who get their first match?
- Maximize the average satisfaction?
- Maximize the minimum satisfaction?
- Can anything go wrong?
Sample Preference Lists

What goes wrong?

<table>
<thead>
<tr>
<th>Man</th>
<th>1\textsuperscript{st}</th>
<th>2\textsuperscript{nd}</th>
<th>3\textsuperscript{rd}</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Y</td>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>Z</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Woman</th>
<th>1\textsuperscript{st}</th>
<th>2\textsuperscript{nd}</th>
<th>3\textsuperscript{rd}</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>
# Sample Preference Lists

What goes wrong?

- Unstable pairs: \((X, C)\) and \((B, Y)\)

<table>
<thead>
<tr>
<th>Man</th>
<th>1(^{st})</th>
<th>2(^{nd})</th>
<th>3(^{rd})</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Y</td>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>Z</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Woman</th>
<th>1(^{st})</th>
<th>2(^{nd})</th>
<th>3(^{rd})</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>
### Sample Preference Lists

- **What goes wrong?**
- **Unstable pairs:** $(X,C)$ and $(B,Y)$
- **They prefer each other to current pairs.**

<table>
<thead>
<tr>
<th>Man</th>
<th>1&lt;sup&gt;st&lt;/sup&gt;</th>
<th>2&lt;sup&gt;nd&lt;/sup&gt;</th>
<th>3&lt;sup&gt;rd&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Y</td>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>Z</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Woman</th>
<th>1&lt;sup&gt;st&lt;/sup&gt;</th>
<th>2&lt;sup&gt;nd&lt;/sup&gt;</th>
<th>3&lt;sup&gt;rd&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>
Definition
Stability: no incentive for some pair of participants to undermine assignment by joint action.
Definition
Stability: no incentive for some pair of participants to undermine assignment by joint action.

- In matching $M$, an unmatched pair $m - w$ is unstable if man $m$ and woman $w$ prefer each other to current partners.
Stability

Definition
Stability: no incentive for some pair of participants to undermine assignment by joint action.

- In matching $M$, an unmatched pair $m - w$ is unstable if man $m$ and woman $w$ prefer each other to current partners.
- Unstable pair $m - w$ could each improve by eloping.
Stable Matching

Definition

A stable matching is a perfect matching with no unstable pairs.

No Pairs creating *instability*.

<table>
<thead>
<tr>
<th>Man</th>
<th>1ˢᵗ</th>
<th>2ⁿᵈ</th>
<th>3ʳᵈ</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Y</td>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>Z</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Woman</th>
<th>1ˢᵗ</th>
<th>2ⁿᵈ</th>
<th>3ʳᵈ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>
Another Stable Matching

No Pairs creating *instability.*
Stable Matching Algorithm


Initialize each person to be free.

while (some man is free and hasn't proposed to every woman) {
    Choose such a man m
    \( w = \text{1st woman on m's list to whom m has not yet proposed} \)
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
}
Stable Matching Algorithm

Initialize each person to be free.

while (some man is free and hasn't proposed to every woman) {
    Choose such a man m
    w = 1st woman on m's list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
}

- Propose-and-reject algorithm. [Gale-Shapley 1962] Intuitive method that guarantees to find a stable matching.

- Alvin Roth and Lloyd Shapley won the Nobel Prize in Economic Sciences for innovative application of the Gale-Shapley algorithm.
Main Idea

- Allow the pairs to keep breaking up and reforming until they become stable
- DEMO
Analysis

Termination?
Analysis

- Termination?
- Perfection?
Analysis

- Termination?
- Perfection?
- Stability?
Observation 1
Men propose to women in decreasing order of preference.

Observation 2
Once a woman is matched, she never becomes unmatched; she only "trades up."

Theorem
*The GS algorithm terminates after at most $n^2$ iterations of the While loop.*

Proof.
Each time through the while loop a man proposes to a new woman. There are only $n^2$ possible proposals. 

□
Theorem

All men and women get matched.

Proof.

(By contradiction) Suppose, for sake of contradiction, that Zeus is not matched upon termination of algorithm. Then some woman, say Amy, is not matched upon termination. By Observation 2, Amy was never proposed to. But, Zeus proposes to everyone, since he ends up unmatched.
Theorem
Consider an execution the GS algorithm that returns a set of pairs $S$. The set $S$ is a stable matching.

Proof.
(By contradiction) We already have shown that $S$ is a perfect matching. Suppose there is an instability. So we have pairs $m - w$ and $m' - w'$ in $S$ such that
- $m$ prefers $w'$ to $w$, and
Theorem
Consider an execution the GS algorithm that returns a set of pairs $S$. The set $S$ is a stable matching.

Proof.
(By contradiction) We already have shown that $S$ is a perfect matching. Suppose there is an instability. So we have pairs $m - w$ and $m' - w'$ in $S$ such that

- $m$ prefers $w'$ to $w$, and
- $w'$ prefers $m$ to $m'$.
Proof.
(continued) In the GS algorithm that produced $S$, $m$'s last proposal must have been $w$. Now, did $m$ propose to $w'$ at some earlier point in this execution?

- If he didn’t, then $w$ must occur higher on $m$’s preference list than $w'$, contradicting our assumption that $m$ prefers $w$ to $w'$.
Proof.
(continued) In the GS algorithm that produced $S$, $m$’s last proposal must have been $w$. Now, did $m$ propose to $w'$ at some earlier point in this execution?

- If he didn’t, then $w$ must occur higher on $m$’s preference list than $w'$, contradicting our assumption that $m$ prefers $w$ to $w'$.

- If he did, then he was rejected by $w'$ in favor of some other man $m''$, whom $w'$ prefers to $m$. $m'$ is the final partner to $w'$, so either $m'' = m'$ or $w'$ prefers her final partner $m'$ to $m''$; either way this contradicts our assumption that $w'$ prefers $m$ to $m'$.
The Gale-Shapley algorithm guarantees to find a stable matching for any problem instance.
What do you have so far?

- The Gale-Shapley algorithm guarantees to find a stable matching for any problem instance.
- How to implement GS algorithm efficiently?
The Gale-Shapley algorithm guarantees to find a stable matching for any problem instance.

How to implement GS algorithm efficiently?

If there are multiple stable matchings, which one does GS find?
Efficient implementation I

We describe $O(n^2)$ time implementation.

- Representing men and women.
  - Assume men are named 1, \ldots, $n$.
  - Assume women are named 1', \ldots, $n'$.

- Engagements
  - Maintain a list of free men, e.g., in a queue.
  - Maintain two arrays $\text{wife}[m]$ and $\text{husband}[w]$.
  - Set entry to 0 if unmatched; if $m$ matched to $w$ then $\text{wife}[m]=w$ and $\text{husband}[w]=m$.

- Men proposing
  - For each man, maintain a list of women, ordered by preference.
  - Maintain an array $\text{count}[m]$ that counts the number of proposals made by man $m$. 

Prof. Dr. Brahim Hnich
Algorithm Design Representative Problems
Efficient implementation I

- We describe $O(n^2)$ time implementation.
- Representing men and women.

Engagements

Maintain a list of free men, e.g., in a queue.
Maintain two arrays $\text{wife}[m]$ and $\text{husband}[w]$.
Set entry to 0 if unmatched; if $m$ matched to $w$ then $\text{wife}[m]=w$ and $\text{husband}[w]=m$.

Men proposing

For each man, maintain a list of women, ordered by preference.
Maintain an array $\text{count}[m]$ that counts the number of proposals made by man $m$. 
Efficient implementation I

- We describe O(n^2) time implementation.
- Representing men and women.
  - Assume men are named 1, \ldots, n.

Engagements

- Maintain a list of free men, e.g., in a queue.
- Maintain two arrays wife[m] and husband[w].
- Set entry to 0 if unmatched; if m matched to w then wife[m]=w and husband[w]=m.

Men proposing

- For each man, maintain a list of women, ordered by preference.
- Maintain an array count[m] that counts the number of proposals made by man m.
Efficient implementation I

- We describe $O(n^2)$ time implementation.
- Representing men and women.
  - Assume men are named $1, \ldots, n$.
  - Assume women are named $1', \ldots, n'$.
We describe $O(n^2)$ time implementation.

Representing men and women.

- Assume men are named 1, \ldots, $n$.
- Assume women are named $1', \ldots, n'$.

Engagements
Efficient implementation I

- We describe $O(n^2)$ time implementation.
- Representing men and women.
  - Assume men are named $1, \ldots, n$.
  - Assume women are named $1', \ldots, n'$.
- Engagements
  - Maintain a list of free men, e.g., in a queue.
Efficient implementation I

- We describe $O(n^2)$ time implementation.
- Representing men and women.
  - Assume men are named 1, \ldots, $n$.
  - Assume women are named 1', \ldots, $n'$.
- Engagements
  - Maintain a list of free men, e.g., in a queue.
  - Maintain two arrays $wife[m]$, and $husband[w]$. 
Efficient implementation I

- We describe $O(n^2)$ time implementation.
- Representing men and women.
  - Assume men are named $1, \ldots, n$.
  - Assume women are named $1', \ldots, n'$.
- Engagements
  - Maintain a list of free men, e.g., in a queue.
  - Maintain two arrays $\text{wife}[m]$, and $\text{husband}[w]$.
  - Set entry to 0 if unmatched; if $m$ matched to $w$ then $\text{wife}[m] = w$ and $\text{husband}[w] = m$
Efficient implementation I

- We describe $O(n^2)$ time implementation.
- Representing men and women.
  - Assume men are named $1, \ldots, n$.
  - Assume women are named $1', \ldots, n'$.
- Engagements
  - Maintain a list of free men, e.g., in a queue.
  - Maintain two arrays $\text{wife}[m]$, and $\text{husband}[w]$.
  - Set entry to 0 if unmatched; if $m$ matched to $w$ then $\text{wife}[m] = w$ and $\text{husband}[w] = m$
- Men proposing
Efficient implementation I

- We describe $O(n^2)$ time implementation.
- Representing men and women.
  - Assume men are named $1, \ldots, n$.
  - Assume women are named $1', \ldots, n'$.
- Engagements
  - Maintain a list of free men, e.g., in a queue.
  - Maintain two arrays $\text{wife}[m]$, and $\text{husband}[w]$.
  - Set entry to $0$ if unmatched; if $m$ matched to $w$ then $\text{wife}[m]=w$ and $\text{husband}[w]=m$
- Men proposing
  - For each man, maintain a list of women, ordered by preference.
Efficient implementation I

▶ We describe $O(n^2)$ time implementation.
▶ Representing men and women.
  ▶ Assume men are named $1, \ldots, n$.
  ▶ Assume women are named $1', \ldots, n'$.
▶ Engagements
  ▶ Maintain a list of free men, e.g., in a queue.
  ▶ Maintain two arrays $\text{wife}[m]$, and $\text{husband}[w]$.
  ▶ Set entry to 0 if unmatched; if $m$ matched to $w$ then $\text{wife}[m]=w$ and $\text{husband}[w]=m$
▶ Men proposing
  ▶ For each man, maintain a list of women, ordered by preference.
  ▶ Maintain an array $\text{count}[m]$ that counts the number of proposals made by man $m$. 
Efficient implementation II

- Women rejecting/accepting

- For each woman, create inverse of preference list of men.

  - Constant time access for each query after $O(n)$ preprocessing.

- Example: Amy prefers man 3 to 6 since $\text{inverse}[3] < \text{inverse}[6]$. 
Efficient implementation II

- Women rejecting/accepting
  - Does woman $w$ prefer man $m$ to man $m'$?
Efficient implementation II

- Women rejecting/accepting
  - Does woman $w$ prefer man $m$ to man $m'$?
  - For each woman, create inverse of preference list of men.
Women rejecting/accepting

- Does woman $w$ prefer man $m$ to man $m'$?
- For each woman, create inverse of preference list of men.
- Constant time access for each query after $O(n)$ preprocessing.
Women rejecting/accepting

- Does woman w prefer man m to man m’?
- For each woman, create inverse of preference list of men.
- Constant time access for each query after O(n) preprocessing.

<table>
<thead>
<tr>
<th>Amy</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
<th>8th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pref</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Amy</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
<th>8th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverse</td>
<td>4th</td>
<td>8th</td>
<td>2nd</td>
<td>5th</td>
<td>6th</td>
<td>7th</td>
<td>3rd</td>
<td>1st</td>
</tr>
</tbody>
</table>

```latex
def inverse(pref, i):
    inverse[pref[i]] = i
```

for i = 1 to n
    inverse[pref[i]] = i
Women rejecting/accepting

- Does woman $w$ prefer man $m$ to man $m'$?
- For each woman, create inverse of preference list of men.
- Constant time access for each query after $O(n)$ preprocessing.

<table>
<thead>
<tr>
<th>Amy</th>
<th>Pref</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>8</td>
<td>4th</td>
</tr>
<tr>
<td>2nd</td>
<td>3</td>
<td>8th</td>
</tr>
<tr>
<td>3rd</td>
<td>7</td>
<td>2nd</td>
</tr>
<tr>
<td>4th</td>
<td>1</td>
<td>5th</td>
</tr>
<tr>
<td>5th</td>
<td>4</td>
<td>6th</td>
</tr>
<tr>
<td>6th</td>
<td>5</td>
<td>7th</td>
</tr>
<tr>
<td>7th</td>
<td>6</td>
<td>3rd</td>
</tr>
<tr>
<td>8th</td>
<td>2</td>
<td>1st</td>
</tr>
</tbody>
</table>

```python
for i = 1 to n
    inverse[pref[i]] = i
```

Amy prefers man 3 to 6 since $\text{inverse}[3] < \text{inverse}[6]$
For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

**Definition**
Woman $w$ is a valid partner of man $m$ if there exists some stable matching in which they are matched.

**Definition**
Woman $w$ is the best valid partner of man $m$ if $w$ is a valid partner of $m$ and there exists no woman whom $m$ ranks higher than $w$ is a valid partner of his.
Theorem

*G-S matching $S$ is man-optimal.*

Proof.

(by contradiction) Suppose some man is paired with someone other than best partner in a matching $S$. Men propose in decreasing order of preference then some man is rejected by (best) valid partner.
Proof.

Let $m$ be first such man, and let $w$ be first valid woman that rejects him. So, $w$ is engaged to some other man $m'$ in $S$. 

Let $S'$ be another stable matching where $m$ and $w$ are matched.

Who is $m'$ matched to in $S'$? Suppose it is woman $w' \neq w$.

$m'$ is not rejected by any valid partner at the point when $m$ is rejected by $w$ in $S$. Thus, $m'$ prefers $w$ to $w'$.

But $w$ prefers $m'$ to $m$ because she rejected $m$ in favor of $m'$.

Thus $m'-w$ is unstable in $S'$. 

Prof. Dr. Brahim Hnich

Algorithm Design Representative Problems
Proof.

- Let $m$ be first such man, and let $w$ be first valid woman that rejects him. So, $w$ is engaged to some other man $m'$ in $S$.
- Let $S'$ be an another stable matching where $m$ and $w$ are matched.

Let $m'$ be matched to in $S'$. Suppose it is woman $w' \neq w$.

$m'$ is not rejected by any valid partner at the point when $m$ is rejected by $w$ in $S$. Thus, $m'$ prefers $w$ to $w'$.

But $w$ prefers $m'$ to $m$ because she rejected $m$ in favor of $m'$.

Thus $m' - w$ is unstable in $S'$. 
Man-optimal assignment II

Proof.

- Let $m$ be first such man, and let $w$ be first valid woman that rejects him. So, $w$ is engaged to some other man $m'$ in $S$.
- Let $S'$ be a another stable matching where $m$ and $w$ are matched.
- Who is $m'$ matched to in $S'$? Suppose it is woman $w' \neq w$.
Proof.

- Let $m$ be first such man, and let $w$ be first valid woman that rejects him. So, $w$ is engaged to some other man $m'$ in $S$.
- Let $S'$ be another stable matching where $m$ and $w$ are matched.
- Who is $m'$ matched to in $S'$? Suppose it is woman $w' \neq w$.
- $m'$ is not rejected by any valid partner at the point when $m$ is rejected by $w$ in $S$. Thus, $m'$ prefers $w$ to $w'$. 
Proof.

- Let $m$ be first such man, and let $w$ be first valid woman that rejects him. So, $w$ is engaged to some other man $m'$ in $S$.
- Let $S'$ be another stable matching where $m$ and $w$ are matched.
- Who is $m'$ matched to in $S'$? Suppose it is woman $w' \neq w$.
- $m'$ is not rejected by any valid partner at the point when $m$ is rejected by $w$ in $S$. Thus, $m'$ prefers $w$ to $w'$.
- But $w$ prefers $m'$ to $m$ because she rejected $m$ in $S$ in favor of $m'$.


Proof.

Let $m$ be first such man, and let $w$ be first valid woman that rejects him. So, $w$ is engaged to some other man $m'$ in $S$.

Let $S'$ be a another stable matching where $m$ and $w$ are matched.

Who is $m'$ matched to in $S'$? Suppose it is woman $w' \neq w$.

$m'$ is not rejected by any valid partner at the point when $m$ is rejected by $w$ in $S$. Thus, $m'$ prefers $w$ to $w'$.

But $w$ prefers $m'$ to $m$ because she rejected $m$ in $S$ in favor of $m'$.

Thus $m' - w$ is unstable in $S'$.
Definition
Man \( m \) is a valid partner of woman \( w \) if there exists some stable matching in which they are matched.

Definition
Man \( m \) is the worst valid partner of woman \( w \) if \( m \) is a valid partner of \( w \) and there exists no man whom \( w \) ranks lower than \( m \) who is a valid partner of hers.

Theorem
In the stable matching \( S \), each woman is paired with her worst valid partner.
Proof.
Suppose there were a pair $m - w$ in $S$ such that $m$ is not the worst valid partner of $w$. Then, there is a stable matching $S'$ in which $w$ is paired with $m' \neq m$ whom she likes less than $m$. In $S'$, $m$ is paired with $w' \neq w$; since $w$ is the best valid partner of $m$ (because $S$ is stable), and $w'$ is a valid partner of $m$ (because of $S'$), we see that $m$ prefers $w$ to $w'$. Thus, $m - w$ is an instability in $S$ which contradicts that $S'$ is stable.
Lessons learned

Process of algorithm design
Lessons learned

Process of algorithm design

1. Formulating the problem with enough mathematical precision
Lessons learned

Process of algorithm design

1. Formulating the problem with enough mathematical precision
2. Designing an algorithm
Lessons learned

- Process of algorithm design
  1. Formulating the problem with enough mathematical precision
  2. Designing an algorithm
  3. Algorithm analysis (correctness and efficiency)
Interval scheduling

- **Input.** Set of jobs with start times and finish times.

![Diagram showing intervals for jobs a, b, c, d, e, f, g, and h.]}
Interval scheduling

- **Input.** Set of jobs with start times and finish times.
- **Goal.** Find maximum cardinality subset of mutually compatible jobs. I.e., jobs that do not overlap.
Interval scheduling

▶ **Input.** Set of jobs with start times and finish times.
▶ **Goal.** Find maximum cardinality subset of mutually compatible jobs. I.e., jobs that do not overlap.
▶ **Approach.** Greedy algorithm
Weighted interval scheduling: dynamic programming

- **Input.** Set of weighted jobs with start times and finish times.
Weighted interval scheduling: dynamic programming

- **Input.** Set of weighted jobs with start times and finish times.
- **Goal.** Find maximum weight subset of mutually compatible jobs. I.e., jobs that do not overlap.
Weighted interval scheduling: dynamic programming

- **Input.** Set of weighted jobs with start times and finish times.
- **Goal.** Find maximum weight subset of mutually compatible jobs. I.e., jobs that do not overlap.
- **Approach.** Dynamic programming
Definition

A graph \( G = (V, E) \) is bipartite if its node set \( V \) can be partitioned into sets \( X \) and \( Y \) in such a way that every edge has one end in \( X \) and the other end in \( Y \).
Bipartite matching II

- **Input.** Bipartite graph.
Bipartite matching II

- **Input.** Bipartite graph.
- **Goal.** Find maximum cardinality matching.
Input. Bipartite graph.

Goal. Find maximum cardinality matching.

Approach. Augmentation (network flow)
Matchings in bipartite graph model situations in which objects are being assigned to other objects
Matchings in bipartite graph model situations in which objects are being assigned to other objects
  1. $X$ can represent jobs and $Y$ machines
Matchings in bipartite graph model situations in which objects are being assigned to other objects

1. $X$ can represent jobs and $Y$ machines
2. $X$ can represent professors and $Y$ courses
Matchings in bipartite graph model situations in which objects are being assigned to other objects

1. $X$ can represent jobs and $Y$ machines
2. $X$ can represent professors and $Y$ courses
3. $X$ can represent drivers and $Y$ vehicles
Independent set

- **Input.** Graph.

![Graph](image)
**Input.** Graph.

**Goal.** Find maximum cardinality independent set. I.e., subset of nodes such that no two joined by an edge.
Independent set

- **Input.** Graph.
- **Goal.** Find maximum cardinality independent set. I.e., subset of nodes such that no two joined by an edge.
- **Approach.** NP-Complete. No known efficient algorithm.
Competitive facility location I

- A set of locations; each location represents a market share
Competitive facility location I

- A set of locations; each location represents a market share
- A graph whose nodes are locations and edges represent location adjacency
Competitive facility location I

- A set of locations; each location represents a market share
- A graph whose nodes are locations and edges represent location adjacency
- Two players $P_1$ and $P_2$ alternately selecting locations
Competitive facility location I

- A set of locations; each location represents a market share
- A graph whose nodes are locations and edges represent location adjacency
- Two players $P_1$ and $P_2$ alternately selecting locations
- At all times, the set of selected locations form an independent set
▶ **Input.** Graph with weight on each node, and an integer $B$.

![Graph diagram]

Second player can guarantee 20, but not 25.
Competitive facility location II

▷ Input. Graph with weight on each node, and an integer $B$.

▷ Goal. Is there a strategy for $P2$ so that no matter how $P1$ plays, $P2$ will be able to select a set of nodes with total value at least $B$.

Second player can guarantee 20, but not 25.
Input. Graph with weight on each node, and an integer $B$.

Goal. Is there a strategy for $P_2$ so that no matter how $P_1$ plays, $P_2$ will be able to select a set of nodes with total value at least $B$.

Approach. PSPACE-Complete. No known efficient algorithm.

Second player can guarantee 20, but not 25.