Introduction:
Some Representative Problems
1.1 A First Problem: Stable Matching
Matching Residents to Hospitals

Goal. Given a set of preferences among hospitals and medical school students, design a self-reinforcing admissions process.

Unstable pair: applicant x and hospital y are unstable if:
- x prefers y to its assigned hospital.
- y prefers x to one of its admitted students.

Stable assignment. Assignment with no unstable pairs.
- Natural and desirable condition.
- Individual self-interest will prevent any applicant/hospital deal from being made.
**Stable Matching Problem**

**Goal.** Given n men and n women, find a "suitable" matching.

- Participants rate members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference from best to worst.

### Men's Preference Profile

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
</tbody>
</table>

### Women's Preference Profile

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Zeus</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
</tbody>
</table>
Perfect matching
Stable Matching Problem

Perfect matching: everyone is matched monogamously.
- Each man gets exactly one woman.
- Each woman gets exactly one man.

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: perfect matching with no unstable pairs.

Stable matching problem. Given the preference lists of $n$ men and $n$ women, find a stable matching if one exists.
Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?

<table>
<thead>
<tr>
<th>Men's Preference Profile</th>
<th>Women's Preference Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>favorite</strong> ↑</td>
<td><strong>favorite</strong> ↑</td>
</tr>
<tr>
<td><strong>least favorite</strong> ↓</td>
<td><strong>least favorite</strong> ↓</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1st</strong></td>
<td><strong>1st</strong></td>
</tr>
<tr>
<td>Xavier</td>
<td>Amy</td>
</tr>
<tr>
<td></td>
<td>Yancey</td>
</tr>
<tr>
<td><strong>2nd</strong></td>
<td><strong>2nd</strong></td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
</tr>
<tr>
<td></td>
<td>Yancey</td>
</tr>
<tr>
<td><strong>3rd</strong></td>
<td><strong>3rd</strong></td>
</tr>
<tr>
<td>Clare</td>
<td>Zeus</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Amy</td>
<td>Yancey</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
</tr>
<tr>
<td>Zeus</td>
<td>Yancey</td>
</tr>
<tr>
<td>Clare</td>
<td>Zeus</td>
</tr>
</tbody>
</table>
Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?
A. No. Bertha and Xavier will hook up.

<table>
<thead>
<tr>
<th>Men's Preference Profile</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Women's Preference Profile</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Zeus</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
</tbody>
</table>
Stable Matching Problem

Q. Is assignment X-A, Y-B, Z-C stable?
A. Yes.
Stable Roommate Problem

Q. Do stable matchings always exist?
A. Not obvious a priori.

Stable roommate problem.
- 2n people; each person ranks others from 1 to 2n-1.
- Assign roommate pairs so that no unstable pairs.

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Bob</td>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>Chris</td>
<td>A</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>Doofus</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

A-B, C-D \implies B-C unstable
A-C, B-D \implies A-B unstable
A-D, B-C \implies A-C unstable

Observation. Stable matchings do not always exist for stable roommate problem.
Propose-And-Reject 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
}
**Proof of Correctness: Termination**

**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."

**Claim.** Algorithm terminates after at most $n^2$ iterations of while loop.

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

\[
\begin{array}{c|c|c|c|c}
1^{\text{st}} & 2^{\text{nd}} & 3^{\text{rd}} & 4^{\text{th}} & 5^{\text{th}} \\
\hline
\text{Victor} & A & B & C & D & E \\
\text{Wyatt} & B & C & D & A & E \\
\text{Xavier} & C & D & A & B & E \\
\text{Yancey} & D & A & B & C & E \\
\text{Zeus} & A & B & C & D & E \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
1^{\text{st}} & 2^{\text{nd}} & 3^{\text{rd}} & 4^{\text{th}} & 5^{\text{th}} \\
\hline
\text{Amy} & W & X & Y & Z & V \\
\text{Bertha} & X & Y & Z & V & W \\
\text{Clare} & Y & Z & V & W & X \\
\text{Diane} & Z & V & W & X & Y \\
\text{Erika} & V & W & X & Y & Z \\
\end{array}
\]

$n(n-1) + 1$ proposals required
Proof of Correctness: Termination

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."

Claim. Algorithm terminates after at most $n^2$ iterations of while loop.

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

\[
\begin{array}{|c|c|c|c|c|}
\hline
& 1^{st} & 2^{nd} & 3^{rd} & 4^{th} & 5^{th} \\
\hline
\text{Victor} & A & B & C & D & E \\
\hline
\text{Wyatt} & B & C & D & A & E \\
\hline
\text{Xavier} & C & D & A & B & E \\
\hline
\text{Yancey} & D & A & B & C & E \\
\hline
\text{Zeus} & A & B & C & D & E \\
\hline
\end{array} \quad \begin{array}{|c|c|c|c|c|}
\hline
& 1^{st} & 2^{nd} & 3^{rd} & 4^{th} & 5^{th} \\
\hline
\text{Amy} & W & X & Y & Z & V \\
\hline
\text{Bertha} & X & Y & Z & V & W \\
\hline
\text{Clare} & Y & Z & V & W & X \\
\hline
\text{Diane} & Z & V & W & X & Y \\
\hline
\text{Erika} & V & W & X & Y & Z \\
\hline
\end{array}
\]

$n(n-1) + 1$ proposals required
Proof of Correctness: Perfection

**Claim.** All men and women get matched.

**Pf.** (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. □
Proof of Correctness: Stability

Claim. No unstable pairs.

Pf. (by contradiction)

- Suppose A-Z is an unstable pair: each prefers each other to partner in Gale-Shapley matching \( S^* \).

- Case 1: Z never proposed to A.
  \[
  \Rightarrow \quad Z \text{ prefers his GS partner to } A.
  \]
  \[
  \Rightarrow \quad A-Z \text{ is stable.}
  \]

- Case 2: Z proposed to A.
  \[
  \Rightarrow \quad A \text{ rejected } Z \text{ (right away or later)}
  \]
  \[
  \Rightarrow \quad A \text{ prefers her GS partner to } Z. \quad \leftarrow \text{ women only trade up}
  \]
  \[
  \Rightarrow \quad A-Z \text{ is stable.}
  \]

- In either case A-Z is stable, a contradiction. \( \blacksquare \)
Summary

Stable matching problem. Given n men and n women, and their preferences, find a stable matching if one exists.

Gale-Shapley algorithm. Guarantees to find a stable matching for any problem instance.

Q. How to implement GS algorithm efficiently?

Q. If there are multiple stable matchings, which one does GS find?
Efficient Implementation

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

Representing men and women.
- Assume men are named 1, ..., n.
- Assume women are named 1', ..., 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 $m$. 
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>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</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>

Amy prefers man 3 to 6
since \( \text{inverse}[3] < \text{inverse}[6] \)

```
for i = 1 to n
    inverse[pref[i]] = i
```
Understanding the Solution

Q. 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?

An instance with two stable matchings.
- A-X, B-Y, C-Z.
- A-Y, B-X, C-Z.
Understanding the Solution

Q. 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?

Def. Man m is a valid partner of woman w if there exists some stable matching in which they are matched.

Man-optimal assignment. Each man receives best valid partner.

Claim. All executions of GS yield man-optimal assignment, which is a stable matching!
- No reason a priori to believe that man-optimal assignment is perfect, let alone stable.
- Simultaneously best for each and every man.
**Man Optimality**

**Claim.** GS matching $S^*$ is man-optimal.

**Pf.** (by contradiction)

- Suppose some man is paired with someone other than best partner. Men propose in decreasing order of preference $\Rightarrow$ some man is rejected by valid partner.
- Let $Y$ be **first** such man, and let $A$ be **first** valid woman that rejects him.
- Let $S$ be a stable matching where $A$ and $Y$ are matched.
- When $Y$ is rejected, $A$ forms (or reaffirms) engagement with a man, say $Z$, whom she prefers to $Y$.
- Let $B$ be $Z$'s partner in $S$.
- $Z$ not rejected by any valid partner at the point when $Y$ is rejected by $A$. Thus, $Z$ prefers $A$ to $B$. $\uparrow$
- But $A$ prefers $Z$ to $Y$.
- Thus $A-Z$ is unstable in $S$. $\blacksquare$
Stable Matching Summary

**Stable matching problem.** Given preference profiles of $n$ men and $n$ women, find a stable matching.

- no man and woman prefer to be with each other than assigned partner

**Gale-Shapley algorithm.** Finds a stable matching in $O(n^2)$ time.

**Man-optimality.** In version of GS where men propose, each man receives best valid partner.

- $w$ is a valid partner of $m$ if there exist some stable matching where $m$ and $w$ are paired

**Q.** Does man-optimality come at the expense of the women?
**Woman Pessimality**

**Woman-pessimal assignment.** Each woman receives worst valid partner.

**Claim.** GS finds woman-pessimal stable matching $S^*$. 

**Pf.**
- Suppose A-Z matched in $S^*$, but Z is not worst valid partner for A.
- There exists stable matching $S$ in which A is paired with a man, say Y, whom she likes less than Z.
- Let B be Z's partner in S.
- Z prefers A to B. ← man-optimality
- Thus, A-Z is an unstable in S. ■
Deceit: Machiavelli Meets Gale-Shapley

Q. Can there be an incentive to misrepresent your preference profile?
   - Assume you know men’s propose-and-reject algorithm will be run.
   - Assume that you know the preference profiles of all other participants.

Fact. No, for any man yes, for some women. No mechanism can guarantee a stable matching and be cheatproof.

**Men’s Preference List**

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Yancey</td>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>Zeus</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

**Women’s True Preference Profile**

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>Bertha</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>Clare</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

*Amy Lies*
Extensions: Matching Residents to Hospitals

**Ex:** Men ≈ hospitals, Women ≈ med school residents.

**Variant 1.** Some participants declare others as unacceptable.

**Variant 2.** Unequal number of men and women.

**Variant 3.** Limited polygamy.

**Def.** Matching is **unstable** if there is a hospital h and resident r such that:
- h and r are acceptable to each other; and
- either r is unmatched, or r prefers h to her assigned hospital; and
- either h does not have all its places filled, or h prefers r to at least one of its assigned residents.
NRMP. (National Resident Matching Program)
- Original use just after WWII. ← predates computer usage
- Ides of March, 23,000+ residents.

Rural hospital dilemma.
- Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.
- Rural hospitals were under-subscribed in NRMP matching.
- How can we find stable matching that benefits "rural hospitals"?

Rural Hospital Theorem. Rural hospitals get exactly same residents in every stable matching!
Lessons Learned

What did we learn?

- Isolate underlying structure of problem.
  (Reduce to the simplest non-trivial “core”)
- Understand the problem structure
- Data structures make a difference in efficiency
- Later, extend to more general problems
- Analyze structure of the solution produced
1.2 Five Representative Problems
Interval Scheduling

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

**Goal.** Find maximum cardinality subset of mutually compatible jobs.

- jobs don’t overlap
Weighted Interval Scheduling

**Input.** Set of jobs with start times, finish times, and weights.

**Goal.** Find maximum weight subset of mutually compatible jobs.
Bipartite Matching

**Input.** Bipartite graph.

**Goal.** Find *maximum cardinality* matching.
**Independent Set**

**Input.** Graph.

**Goal.** Find **maximum cardinality** independent set.

subset of nodes such that no two joined by an edge
Competitive Facility Location

Input. Graph with weight on each node.

Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors have been selected.

Goal. Select a maximum weight subset of nodes.

Second player can guarantee 20, but not 25.
Five Representative Problems

Variations on a theme: independent set.

Interval scheduling: $n \log n$ greedy algorithm.
Weighted interval scheduling: $n \log n$ dynamic programming algorithm.
Bipartite matching: $n^k$ max-flow based algorithm.
Independent set: NP-complete.
Competitive facility location: PSPACE-complete.
"For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing." - Francis Sullivan
Polynomial-Time

**Brute force.** For many non-trivial problems, there is a natural brute force search algorithm that checks every possible solution.
- Typically takes $2^N$ time or worse for inputs of size $N$.
- Unacceptable in practice.

**Desirable scaling property.** When the input size doubles, the algorithm should only slow down by some constant factor $C$.

There exists constants $c > 0$ and $d > 0$ such that on every input of size $N$, its running time is bounded by $c N^d$ steps.

**Def.** An algorithm is *poly-time* if the above scaling property holds.

\[
\text{choose } C = 2^d
\]
Worst-Case Analysis

Worst case running time. Obtain bound on largest possible running time of algorithm on input of a given size $N$.
- Generally captures efficiency in practice.
- Draconian view, but hard to find effective alternative.

Average case running time. Obtain bound on running time of algorithm on random input as a function of input size $N$.
- Hard (or impossible) to accurately model real instances by random distributions.
- Algorithm tuned for a certain distribution may perform poorly on other inputs.
Worst-Case Polynomial-Time

**Def.** An algorithm is **efficient** if its running time is polynomial.

**Justification:** It really works in practice!
- Although $6.02 \times 10^{23} \times N^{20}$ is technically poly-time, it would be useless in practice.
- In practice, the poly-time algorithms that people develop almost always have low constants and low exponents.
- Breaking through the exponential barrier of brute force typically exposes some crucial structure of the problem.

**Exceptions.**
- Some poly-time algorithms do have high constants and/or exponents, and are useless in practice.
- Some exponential-time (or worse) algorithms are widely used because the worst-case instances seem to be rare.
Why It Matters

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds $10^{25}$ years, we simply record the algorithm as taking a very long time.

<table>
<thead>
<tr>
<th>n</th>
<th>$n$</th>
<th>$n \log_2 n$</th>
<th>$n^2$</th>
<th>$n^3$</th>
<th>$1.5^n$</th>
<th>$2^n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 10$</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>4 sec</td>
</tr>
<tr>
<td>$n = 30$</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>18 min</td>
<td>$10^{25}$ years</td>
</tr>
<tr>
<td>$n = 50$</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>11 min</td>
<td>36 years</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 100$</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>1 sec</td>
<td>12,892 years</td>
<td>$10^{17}$ years</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 1,000$</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>1 sec</td>
<td>18 min</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 10,000$</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>2 min</td>
<td>12 days</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 100,000$</td>
<td>&lt; 1 sec</td>
<td>2 sec</td>
<td>3 hours</td>
<td>32 years</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 1,000,000$</td>
<td>1 sec</td>
<td>20 sec</td>
<td>12 days</td>
<td>31,710 years</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
</tbody>
</table>