Lecture 2
Analysis of Algorithms
- Stable matching problem
- Asymptotic growth

Adam Smith

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Stable Matching Problem

- **Unstable pair**: man \( m \) and woman \( w \) are unstable if
  - \( m \) prefers \( w \) to his assigned match, and
  - \( w \) prefers \( m \) to her assigned match
- Unstable pairs have an incentive to elope
- **Stable matching**: no unstable pairs.

### Men's Preference Profile

<table>
<thead>
<tr>
<th>favorite</th>
<th>least favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>Xavier</td>
<td>Amy</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
</tr>
</tbody>
</table>

### Women's Preference Profile

<table>
<thead>
<tr>
<th>favorite</th>
<th>least favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>Amy</td>
<td>Yancey</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
</tr>
<tr>
<td>Clare</td>
<td>Xavier</td>
</tr>
</tbody>
</table>

8/30/10

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Stable Matching Problem

• **Input:** preference lists of \( n \) men and \( n \) women

• **Goal:** find a stable matching if one exists

---

**Men’s Preference Profile**

<table>
<thead>
<tr>
<th>favorite</th>
<th>least favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>Xavier</td>
<td>Amy</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
</tr>
</tbody>
</table>

**Women’s Preference Profile**

<table>
<thead>
<tr>
<th>favorite</th>
<th>least favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>Amy</td>
<td>Yancey</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
</tr>
<tr>
<td>Clare</td>
<td>Xavier</td>
</tr>
</tbody>
</table>

8/30/10
A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Review Questions

• In terms of $n$, what is the length of the input to the Stable Matching problem, i.e., the number of entries in the tables?

• How many bits do they take to store?

  (Answer: $2n^2$ list entries, or $2n^2\log n$ bits)
Review Questions

- **Brute force algorithm:** an algorithm that checks every possible solution.

- In terms of $n$, what is the running time for the brute force algorithm for checking whether a given matching is stable?

- In terms of $n$, what is the running time for the brute force algorithm for Stable Matching Problem? (Assume your algorithm goes over all possible perfect matchings.)

  \[(Answer: \ n! \times (\text{time to check if a matching is stable}) = \Theta(n! \ n^2))\]
Propose-and-Reject Algorithm

- Propose-and-reject algorithm. [Gale-Shapley 1962]

Initialize each person to be free.

while (some man is free and hasn't proposed to every woman) {
    Choose such a man m
    w = 1\textsuperscript{st} 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

Three claims: The algorithm always
1. terminates,
2. matches everyone (matching is “perfect”), and
3. outputs a stable matching
### Men's Preference Profile

<table>
<thead>
<tr>
<th></th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victor</td>
<td>Bertha</td>
<td>Amy</td>
<td>Diane</td>
<td>Erika</td>
<td>Clare</td>
</tr>
<tr>
<td>Wyatt</td>
<td>Diane</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
<td>Erika</td>
</tr>
<tr>
<td>Xavier</td>
<td>Bertha</td>
<td>Erika</td>
<td>Clare</td>
<td>Diane</td>
<td>Amy</td>
</tr>
<tr>
<td>Yancey</td>
<td>Amy</td>
<td>Diane</td>
<td>Clare</td>
<td>Bertha</td>
<td>Erika</td>
</tr>
<tr>
<td>Zeus</td>
<td>Bertha</td>
<td>Diane</td>
<td>Amy</td>
<td>Erika</td>
<td>Clare</td>
</tr>
</tbody>
</table>

STOP

- Everyone matched.
- Stable matching!

### Women's Preference Profile

<table>
<thead>
<tr>
<th></th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Zeus</td>
<td>Victor</td>
<td>Wyatt</td>
<td>Yancey</td>
<td>Xavier</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Wyatt</td>
<td>Yancey</td>
<td>Victor</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Wyatt</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
<td>Victor</td>
</tr>
<tr>
<td>Diane</td>
<td>Victor</td>
<td>Zeus</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Wyatt</td>
</tr>
<tr>
<td>Erika</td>
<td>Yancey</td>
<td>Wyatt</td>
<td>Zeus</td>
<td>Xavier</td>
<td>Victor</td>
</tr>
</tbody>
</table>
Proof of Correctness: Termination

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

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

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victor</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Wyatt</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>Xavier</td>
<td>C</td>
<td>D</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Yancey</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Zeus</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>Bertha</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>V</td>
</tr>
<tr>
<td>Clare</td>
<td>Y</td>
<td>Z</td>
<td>V</td>
<td>W</td>
</tr>
<tr>
<td>Diane</td>
<td>Z</td>
<td>V</td>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>Erika</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

An instance where $n(n-1) + 1$ proposals required
### Propose-and-Reject Algorithm

- **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."

### Men's Preference Profile

<table>
<thead>
<tr>
<th></th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victor</td>
<td>Bertha</td>
<td>Amy</td>
<td>Diane</td>
<td>Erika</td>
<td>Clare</td>
</tr>
<tr>
<td>Wyatt</td>
<td>Diane</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
<td>Erika</td>
</tr>
<tr>
<td>Xavier</td>
<td>Bertha</td>
<td>Erika</td>
<td>Clare</td>
<td>Diane</td>
<td>Amy</td>
</tr>
<tr>
<td>Yancey</td>
<td>Amy</td>
<td>Diane</td>
<td>Clare</td>
<td>Bertha</td>
<td>Erika</td>
</tr>
<tr>
<td>Zeus</td>
<td>Bertha</td>
<td>Diane</td>
<td>Amy</td>
<td>Erika</td>
<td>Clare</td>
</tr>
</tbody>
</table>

### Women's Preference Profile

<table>
<thead>
<tr>
<th></th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Zeus</td>
<td>Victor</td>
<td>Wyatt</td>
<td>Yancey</td>
<td>Xavier</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Wyatt</td>
<td>Yancey</td>
<td>Victor</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Wyatt</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
<td>Victor</td>
</tr>
<tr>
<td>Diane</td>
<td>Victor</td>
<td>Zeus</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Wyatt</td>
</tr>
<tr>
<td>Erika</td>
<td>Yancey</td>
<td>Wyatt</td>
<td>Zeus</td>
<td>Xavier</td>
<td>Victor</td>
</tr>
</tbody>
</table>

*8/30/10*  
*A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne*
Proof of Correctness: Perfection

- **Claim.** All men and women get matched.
- **Proof:** (by contradiction)
  - Suppose, for sake of contradiction, some guy, say 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.

• **Proof:** (by contradiction)
  – Suppose A-Z is an unstable pair: they prefer each other to their partners in Gale-Shapley matching S*.
  – **Case 1:** Z never proposed to A.
    ⇒ Z prefers his GS partner to A.
    ⇒ A-Z is stable.
  – **Case 2:** Z proposed to A.
    ⇒ A rejected Z (right away or later)
    ⇒ A prefers her GS partner to Z.
    ⇒ A-Z is stable.
  – In either case A-Z is stable, a contradiction. •
Stable Roommate Problem

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

- **Exercise.** Where does the correctness proof break down for the roommates version?
Efficient Implementation

• We describe $O(n^2)$ time implementation.
• Assume men have IDs 1, ..., $n$, and so do women.
• Engagements data structures:
  – a list of free men, e.g., a queue.
  – 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 data structures:
  – an array $men\text{-}pref[m,i]=i^{th}$ women on $m^{th}$ list
  – an array $count[m]=how\ many\ proposals\ m\ made$.
• In Python: http://euler.slu.edu/~goldwasser/courses/slucsci314/2006_Spring/lectures/marriageAlgorithm/

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
**Efficient Implementation**

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

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

**Examples:**

- Amy prefers man 3 to 6 since $\text{inverse}[3] < \text{inverse}[6]$

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

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
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 every problem instance.
  – (Also proves that stable matching always exists)

• **Time and space complexity:**

  $O(n^2)$, linear in the input size.
Brief Syllabus

• Reminders
  – Worst-case analysis
  – Asymptotic notation
  – Basic Data Structures

• Design Paradigms
  – Greedy algorithms, Divide and conquer, Dynamic programming, Network flow and linear programming, randomization

• Analyzing algorithms in other models
  – Parallel algorithms, Memory hierarchies (?)

• P, NP and NP-completeness
Measuring Running Time

• Focus on **scalability**: parameterize the running time by some measure of “size”
  – (e.g. $n =$ number of men and women)

• Kinds of analysis
  – Worst-case
  – Average-case (requires knowing the distribution)
  – Best-case (how meaningful?)

• Exact times depend on computer; instead measure **asymptotic growth**
Asymptotic notation

\( O \)-notation (upper bounds):

We write \( f(n) = O(g(n)) \) if there exist constants \( c > 0, \ n_0 > 0 \) such that \( 0 \leq f(n) \leq cg(n) \) for all \( n \geq n_0 \).

**Example:** \( 2n^2 = O(n^3) \) \( (c = 1, n_0 = 2) \)

functions, not values

funny, “one-way” equality

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Set Definition

\[ O(g(n)) = \{ f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \} \]

**Example:** \( 2n^2 \in O(n^3) \)

(Logicians: \( \lambda n.2n^2 \in O(\lambda n.n^3) \), but it’s convenient to be sloppy, as long as we understand what’s really going on.)
Examples

• $10^6 n^3 + 2n^2 - n + 10 = O(n^3)$

• $n^{\frac{1}{2}} + \log(n) = O(n^{\frac{1}{2}})$

• $n (\log(n) + n^{\frac{1}{2}}) = O(n^{3/2})$

• $n = O(n^2)$
Ω-notation (lower bounds)

\[ \Omega(g(n)) = \{ f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \} \]

\textbf{Example:} \quad \sqrt{n} = \Omega(\log n) \quad (c = 1, \ n_0 = 16)

\textbf{O-notation} is an upper-bound notation. It makes no sense to say \( f(n) \) is at least \( O(n^2) \).
**Ω-notation (lower bounds)**

- **Be careful:** “Any comparison-based sorting algorithm requires at least $O(n \log n)$ comparisons.”
  - Meaningless!
  - Use $\Omega$ for lower bounds.
Θ-notation (tight bounds)

\[ \Theta(g(n)) = \Omega(g(n)) \cap \Omega(g(n)) \]

**Example:** \( \frac{1}{2} n^2 - 2n = \Theta(n^2) \)

Polynomials are simple:
\[ a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0 = \Theta(n^d) \]
o-notation and $\omega$-notation

$O$-notation and $\Omega$-notation are like $\leq$ and $\geq$.

$o$-notation and $\omega$-notation are like $<$ and $>$. 

\[
o(g(n)) = \{ f(n) : \text{for any constant } c > 0, \text{ there is a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0 \}
\]

**Example:**

\[2n^2 = o(n^3) \quad (n_0 = 2/c)\]
\( o - \text{notation and } \omega - \text{notation} \)

\( O\)-notation and \( \Omega\)-notation are like \( \leq \) and \( \geq \).
\( o\)-notation and \( \omega\)-notation are like \( < \) and \( > \).

\[ \omega(g(n)) = \{ f(n) : \text{for any constant } c > 0, \text{there is a constant } n_0 > 0 \text{ such that } 0 \leq cg(n) < f(n) \text{ for all } n \geq n_0 \} \]

**Example:** \( \sqrt{n} = \omega(\lg n) \) \( (n_0 = 1 + 1/c) \)
Common Functions: Asymptotic Bounds

- **Polynomials.** \( a_0 + a_1 n + \ldots + a_d n^d \) is \( \Theta(n^d) \) if \( a_d > 0 \).

- **Polynomial time.** Running time is \( O(n^d) \) for some constant \( d \) independent of the input size \( n \).

- **Logarithms.** \( \log_a n = \Theta(\log_b n) \) for all constants \( a, b > 0 \).

  - Can avoid specifying the base.
  - Log grows slower than every polynomial.

  For every \( x > 0 \), \( \log n = O(n^x) \).

- **Exponentials.** For all \( r > 1 \) and all \( d > 0 \), \( n^d = O(r^n) \).

- **Factorial.**

  \[
  n! = \left( \sqrt{2\pi n} \right) \left( \frac{n}{e} \right)^n \left( 1 + o(1) \right) = 2^{\Theta(n \log n)}
  \]

  - Grows faster than every exponential.

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Sort by asymptotic order of growth

\[ a) \quad n \log(n) \]
\[ b) \quad \sqrt{n} \]
\[ c) \quad \log(n) \]
\[ d) \quad n^2 \]
\[ e) \quad 2^n \]
\[ f) \quad n \]
\[ g) \quad n! \]
\[ h) \quad n^{1,000,000} \]
\[ i) \quad n^{1/\log(n)} \]
\[ j) \quad \log(n!) \]
\[ k) \quad \binom{n}{2} \]
\[ l) \quad \binom{n}{n/2} \]