Welcome!
Algorithm Design & Analysis

Course Focus

- Effective fundamental design paradigms
- Tools & techniques for analysis of resource use
- Ability to accurately gauge problem difficulty
- Strategies for “intractable” problems
- Enhanced problem-solving skills
Who’s Who

Instructor

• Bill Lenhart (wlenhart@williams.edu)
• Office: TPL 304 (x2371)
• Office Hours: M/T/Th: 2:30-3:50 pm; T: 9:00-10:00 am
  • And by Appointment as needed/possible

TAs

• Ari Ball-Burack, Lester Lee, Dzung Pham, Haoyu Sheng, Tongyu Zhou
What’s What

The Text

- *Algorithm Design* by Kleinberg & Tardos (1st Ed.)

Problem Sets, Exams, Projects

- Weekly Problem Sets
  - Assigned on Thursday and submitted by 11:00am (in class) the following Wednesday
  - Collaboration is fine but must be cited
  - Problem Set 0 is out now; due next Wednesday, Sept. 12
  - Problem Sets must be typeset using LaTeX
- Mid-Term (10/19 - 10/22) and Final Exams
  - Take-home; collaboration prohibited

The Course Website

- [https://www.cs.williams.edu/~lenhart/cs256/index.html](https://www.cs.williams.edu/~lenhart/cs256/index.html)
The Fine Print

Expectations

• Read all information on the course website
• Come to class (reasonably) prepared
  • Read in advance to get general idea
• Contribute
  • Attend and participate
  • Help make the course a great experience for all
• Assignments
  • Start early, finish on time
• Adhere to the Honor Code
• Have fun & learn (a lot)
An Illustrative Example

Matching Problems

• Assigning first year students to advisors
• Pairing job candidates with employers
• The “Algorithm of Happiness”
  • The National Resident Matching Program

Fundamental Problem

• Find a matching that protects against opportunistic swapping
### 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>

### Matching

- **V** connected to **D**
- **W** connected to **A**
- **X** connected to **E**
- **Y** connected to **C**
- **Z** connected to **B**

### Instability

- **W** connected to **A**
- **Z** connected to **B**
State the Problem

Two groups: Applicants & Employers

• Each applicant has ranked desired employers
• Each employer has ranked desired applicants

Goal

• Match as many applicants as possible, avoiding instabilities
  • Instability: $a$ is not matched to $e$, but $a$ & $e$ both prefer each other to their matches
# Play With the Problem

## Men’s Preference Profile

<table>
<thead>
<tr>
<th></th>
<th>0&lt;sup&gt;th&lt;/sup&gt;</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>
<th>4&lt;sup&gt;th&lt;/sup&gt;</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>0&lt;sup&gt;th&lt;/sup&gt;</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>
<th>4&lt;sup&gt;th&lt;/sup&gt;</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>

## Matching

```
X ——— B
```

```
W ——— A
Z ——— D
```
Find the Simplest Interesting Version

Distracting Complications

• Applicants/Employers have incomplete rankings
• Employer might make several hires
• Might not be enough jobs for all applicants

The Original: Stable Marriage Problem

• $n$ men & $n$ women
• every man and woman submits a complete ranking

Goal

• Find a stable matching
Restate the Problem

Instability in matching M

• Unmatched pair \{m,w\}, with \{m,w’\}, \{w,m’\} in M where m prefers w to w’ and w prefers m to m’

Stable Matching: No instabilities

• For every unmatched \{m, w\},
  • Either m prefers his match to w,
  • Or w prefers her match to m

Does such a matching always exist?

Can we find it? What if n = 50,000?

If so, how?
Design an Algorithm


• Iterative process in which
  • A man proposes to a woman (G & S were men)
  • A woman can break engagement in favor of man she prefers
• Stop when everyone is matched (hopefully!)
• Claim: Stable matching has been produced
Design: Precise Version

Gale & Shapley: Propose - Reject Algorithm (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 = m’s most preferred woman to whom m hasn’t yet proposed
    if (w is free)
        assign m and w to be engaged  // m & w no longer free
    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  // m remains free

Questions

• Does it halt?
• Does it match everyone?
• Is the matching stable?
Analyzing the Algorithm: Correctness

Look for Helpful Features of Algorithm

• Every iteration eliminates a potential proposal
  • So algorithm at least halts [! Find a progress measure !]
• Once engaged, a woman stays engaged
• Women always ‘trade up’
  • If w breaks engagement with m in favor of m’ then w prefers m’ to m
• Every woman gets matched
  • So the matching is complete (perfect matching)
Analyzing the Algorithm: Correctness II

The Matching is Stable

- Suppose not: Then final matching has an instability \{m,w\}
  - m is matched to w’ but m prefers w over w’
  - w is matched to m’ but w prefers m over m’
- So m proposed to w before proposing to w’
  - So either w broke engagement to m at some point, or w already had a match m’’ that she preferred over m
  - But women always trade up, so final spouse of w (m’) must be preferred by w to m—>Contradiction!
Analyzing the Algorithm: Performance

Establish Running Time and Space Requirements

- Space: $2n$ lists of length $n$: $O(n^2)$
- Time: Count a representative set of operations
  - Select a free man $m$
  - Find most preferred $w$ not yet proposed to by $m$
  - Find $w$'s ranking of a given man
  - Make an engaged man free again
  - Add to & delete from set of engaged couples
Wait! What’s the Input?

- Assume each person provides ordered list (most to least preferred) of partners

Identify Efficient Data Structures for Operations

- Assume women are named 1..n, and men are named 1..n
- Free men: get(), put() : Queue: $O(1)$ for both
- Find $m$’s most preferred, not yet asked (by $m$) woman
  - $m$ provided a sorted list: $O(1)$
Identify Efficient Data Structures for Operations

• Find w’s ranking of a given man
  • For each w, an array, indexed by man, of her ranking of men (inverted index)
    • $w(m) = i$ if $m$ has rank $i$ in w’s list
  • Build this from w’s ordered list of men: $O(n)$ per list
• Add to & delete from set of engaged couples:
  • Array, Engaged(w) = $m$ currently engaged to w (or ‘free’): Creation time $O(n)$; update time $O(1)$
Analyzing the Algorithm: Performance IV

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  // m & w no longer free
    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  // m remains free
}

Overall Running Time

- Preprocessing Time + Loop Time
  - Preprocessing: $O(n^2)$ (for inverted lists) + $O(n)$ : (for Engaged())
  - Loop: $O(n^2) \times O(1)$ (all steps take constant time)
Summary: How to Solve a Problem

State the Problem

Play With the Problem

Find the Simplest Interesting Version of the Problem

Clearly restate the Simplest Version of the Problem

Design an Algorithm & Make it Precise

Identify Properties of Algorithm to Prove Correctness
  • Go back to previous step if (as is most likely) necessary

Establish Running Time and Space Requirements
  • Identify Efficient Data Structures for Operations

But Wait, There’s More….
“In theory, there is no difference between theory and practice; in practice, there is.”
What Usually Happens

Poor Understanding of Problem

No Intuition About Problem

Confusion Due to Overwhelming Detail

Poor Statement of Problem

Multiple Failed Attempts to Design Algorithm

Inability to Prove Correctness
  • Often because algorithm does not actually work correctly

Trade-Offs Between Time & Space Resources Create Difficult Choices

But Wait, There’s More....
Next Time

Return to Original Problem
Exploit Algorithm for Deeper Insight into Problem
Explore Extensions
Sampler of Representative Problems from Course