CS 231: Fundamental Algorithms

Spring 2019
What is an Algorithm?

“A procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation.” — webster.com

“An algorithm is a finite, definite, effective procedure, with some input and some output.”

— Donald Knuth
What is Algorithm Design?

How do you write a computer program to solve a complex problem?

- Computing similarity between DNA sequences
- Routing packets on the Internet
- Find all occurrences of a phrase in a large collection of documents
- Finding the smallest number of coffee shops that can be built in the US such that everyone is within 20 minutes of a coffee shop.
DNA Sequence Similarity

- **Input:** two n-bit strings $s_1$ and $s_2$
  - $s_1 = \text{AGGCTACC}$
  - $s_2 = \text{CAGGCTAC}$

- **Output:** minimum number of insertions/deletions to transform $s_1$ into $s_2$

- **Algorithm:** ???

- Even if the objective is precisely defined, we are often not ready to start coding right away!
What is Algorithm Design?

- Step 1: Formulate the problem precisely
- Step 2: Design an algorithm
- Step 3: Prove the algorithm is correct
- Step 4: Analyze its running time

**Important:** this is an iterative process, e.g., sometimes you’ll even want to redesign the algorithm to make it easier to prove that it is correct
Course Goals

- Learn the design and analysis of algorithms to solve problems
- Learn specific algorithm design techniques
  - Greedy
  - Divide-and-conquer
  - Dynamic Programming
- Learn to communicate precisely about algorithms
  - Proofs, reading, writing, discussion
- Prove when no exact efficient algorithm is possible
  - Intractability and NP-completeness
Why take this course?

- **Understanding and Remembering:**
  - Recognize algorithmic techniques used to solve a problem
  - Identify the correctness, or lack thereof, of an algorithm

- **Critical Thinking:**
  - Dissect new problems to identify their input and corresponding output.

- **Practical Thinking:**
  - Determine appropriate algorithmic techniques to solve new problems, by relating new problems to ones in their foundation knowledge.
  - Critique existing algorithms.
  - Calculate the asymptotic run time complexity of new algorithms.
Why take this course?

- Projects and Research:
  - Coordinate tasks and collaborate on writing a final paper.
  - Identify high quality scholarly articles, and their contributions.
  - Summarize existing algorithmic research on a topic of their choice
  - Present summary of research to peers, as part of a team.

- Interpersonal Relationships:
  - Collaborate with peers on dissecting new problems.
  - Take responsibility for work performed as part of a group.
Administrivia

- Syllabus and schedule on website: http://cs.wellesley.edu/~cs231/
  - Will post slides, handouts and assignments on it

- Textbook
  - Algorithm Design, by Jon Kleinberg and Eva Tardos
Administrivia

- Assignment are posted on the website and submitted through Gradescope
  - Sign up using the course code: (9PYWJN)
- Assignments need to be typeset using LaTeX
  - LaTeX resources posted on course site; template will be provided
- Late policy - 4 x 24h late passes
- Collaboration policy
  - Discussion on assignment problems is encouraged
  - You must write your own solutions; do not share/copy solutions
  - On every assignment, you must cite your collaborator and your sources
Administrivia

- Exams are in-class and open book
  - February 25th
  - April 8th
  - May 2nd

- Final short paper and presentation
Stable Matching Problem
Matching Residents to Hospitals

- **Problem:** Given a set of preferences among hospitals and medical school students
  - Can we match applicants with hospitals such that everyone is happy?
  - Can we match applicants with hospitals such that the matching is stable?

- **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 residents

- **Stable assignment:** No unstable pairs exists
  - Natural and desirable condition
  - Individual self-interest will prevent any applicant/hospital deal from being made
Stable Matching Problem

- **Input:** given $n$ residents and $n$ hospitals, with their rating of each other
  - Each resident lists hospitals in order of preference from best to worst
  - Each hospital lists residents in order of preference from best to worst

- **Output:** a “good” matching of residents and hospitals

<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>

<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>

*Residents’ Preference Profile*  
*Hospitals’ Preference Profile*
Definitions

- **Perfect matching**: everyone is matched monogamously
  - Each resident gets exactly one hospital
  - Each hospital gets exactly one resident

- **Stability**: no incentive for some pair of participants to undermine assignment by joint action
  - In a matching S, an unmatched pair r-h is unstable if
  - Unstable pair r-h could each improve by breaking contracts

- **Stable matching**: perfect matching with no unstable pairs

- **Stable matching problem**: Given the preference lists of n residents and n hospitals, find a stable matching if one exists
Is matching X-C, Y-B, Z-A stable?

<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>

*Residents’ 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>

*Hospitals’ Preference Profile*
Is matching X-C, Y-B, Z-A stable?

- No. Bertha and Xavier would break contract.
- An unstable pair could improve by joint action

<table>
<thead>
<tr>
<th>Residents’ Preference Profile</th>
<th>Hospitals’ Preference Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1&lt;sup&gt;st&lt;/sup&gt;</strong></td>
<td><strong>1&lt;sup&gt;st&lt;/sup&gt;</strong></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>
Is assignment X-A, Y-B, Z-C stable?

Residents’ Preference Profile

<table>
<thead>
<tr>
<th></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>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>

Hospitals’ Preference Profile

<table>
<thead>
<tr>
<th></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>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>
Is assignment X-A, Y-B, Z-C stable?

- Yes!

<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>

Residents’ 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>

Hospitals’ Preference Profile
Propose and Reject (Gale-Shapley) Algorithm

Initially all hospitals and student applicants are free

while some hospital is free and hasn’t proposed to every student

do

Choose such a hospital h
Let s be the highest ranked student to whom h has not proposed

if s is free then

   h and s become matched

else if s is matched to h’ but prefers h to h’ then

   h’ becomes unmatched
   h and s become matched

else

   s rejects h and h remains free

end if

end while
Analyzing the Algorithm

● Does the algorithm terminate?

● Does the algorithm guarantee that every student and hospital gets a match?

● Does the algorithm return a stable match?

● Observe that:
  ○ Hospital propose to students in order of hospital’s preferences
  ○ Every student only “upgrades” during the algorithm
Does the algorithm terminate?

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

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