CPSC 500: Fundamentals of Algorithm Design and Analysis
Outline for today

• Course goals, topics, work, resources
• Course prerequisites

• First topic: stable matchings
Course goals

• Help you learn algorithms fundamentals that have stood the test of time, as well as some more recently discovered “fundamentals”

• Look at algorithms both for traditional combinatorial optimization problems, as well as more modern variants, e.g., for online settings

• Think both about problem formulation as well as algorithmic solutions. Solving the right problem is key!
Course topics

• *Algorithms on graphs*: classical problems and new variants such as online matching
• *Linear* programming, duality, and their applications
• *Algorithms on strings*: from classical algorithms to new innovations that are motivated by massive datasets in genomics
• *Algorithms for clustering*: a problem where problem formulation is tricky
• *Approximation algorithms* for hard combinatorial optimization problems
Course prerequisites

• Introductory algorithms class or equivalent (e.g., CPSC 320). A good treatment of introductory material is Chapters 1 through 6 of Kleinberg and Tardos' "Algorithm Design".

• You should also be familiar with the theory of NP-completeness and how it is used to show that certain combinatorial optimization problems are intractable. See Chapter 8 of Kleinberg and Tardos' text, at least through section 8.4. Chapters 3 and 4 of Mertens and Moore's text also has great coverage of this topic.
Course project

• The goal of your project will be to find an algorithm or problem of interest to you, learn what you can about it, and present what you’ve learned to the class at the end of the semester.
• I’ll put suggested topics and papers on the website soon, but don’t feel limited by them.
• I recommend groups of size 2-4.
Course work

• Three homework assignments: 50%
• Student reading project and presentations: 20% (more on this soon...)
• Final exam: 30%

• Practice homeworks, and opportunities for problem-solving during class
• Lots of room for individualized learning
Course work

• Three homework assignments: 50%
• Student reading project and presentations: 20% (more on this soon…)
• Final exam: 30%

• Practice homeworks, and opportunities for problem-solving during class
• Lots of room for individualized learning
First topic is algorithms for matching: ties in with course goals of

• covering fundamentals
• considering both classical problems and modern variants
• working on problem formulation
Stable matching

- See online slides and references for background
Stable matching

• See online slides and references for background
• Many applications!
  – Canadian resident matching service (and U.S. equivalent, the National resident matching program) that match students into medical training programs
  – Assigning TAs to courses, assigning mentors to mentees, ...
Stable matching: input

- n profs p₁, ..., pₙ, and m students s₁, ..., sₘ
- TA capacity c(pi) for each professor (course)
- students have partial preference lists of acceptable profs
- professors have partial preference lists of acceptable students

(each partial preference list is a total order of the acceptable matches)
Stable matching: notation

- Acceptable pair (p,s): p is on s's list and vice versa.
- Assignment A: a subset of acceptable pairs
- A(s): set of professors assigned to student s
- A(p): set of students assigned to professor p
- Matching M: an assignment with |M(pi)| ≤ c(pi) and |M(si)| ≤ 1
Stable matching: blocking pair

• With respect to a matching M, pair \((p,s)\) is *blocking* if the following are all true.
  
  (i) Pair \((p,s)\) acceptable and is not in M.
  
  (ii) Either \(s\) is unassigned, or \(s\) prefers \(p\) to \(M(s)\).
  
  (iii) Either \(M(p) < c(p)\), or \(p\) prefers \(s\) to at least one member of \(M(p)\).
Gale-Shapley Algorithm (student optimal)

\[ M := \text{nullset}; \]

\[
\text{WHILE (some student } s \text{ is unmatched and hasn’t applied to every prof on its list)}
\]

\[
p \leftarrow \text{first prof on } s’s \text{ preference list that } s \text{ has not yet approached} //s \text{ now approaches } p
\]

\[
\text{IF } (|M(p)| < c(p) \text{ and } s \text{ is acceptable to } p)
\]

\[
\text{Add } (p,s) \text{ to } M
\]

\[
\text{ELSE IF } (p \text{ prefers } s \text{ to some student in } M(p))
\]

\[
\text{remove } (p,s’) \text{ from } M, \text{ where } s’ \text{ is } p’s \text{ least preferred student in } M(p)
\]

\[
\text{add } (p,s) \text{ to } M
\]

\[
\text{RETURN matching } M
\]
Gale-Shapley Algorithm (professor optimal)

\[
M := \text{nullset;}
\]

WHILE (some professor \( p \) is not at capacity and hasn’t approached every student on its list)

\[
s \leftarrow \text{student most preferred by } p \text{ that } p \text{ has not yet approached} \; // p \text{ now approaches } s
\]

IF ( \( s \) is unassigned and \( p \) is acceptable to \( s \))

Add \((p,s)\) to \( M \)

ELSE IF ( \( s \) prefers \( p \) to \( M(s) \))

remove \((M(s),s)\) \( M \)

add \((p,s)\) to \( M \)

RETURN matching \( M \)
Gale-Shapley Algorithm: termination

- Does the algorithm terminate? Yes, after at most nm iterations. At each iteration, some prof approaches some student. There are n profs and m students, and so if the algorithm runs for nm iterations, all profs must have approached all students. At this point, the condition of the while loop no longer holds and the algorithm returns M.
Does the professor-optimal algorithm produce a stable matching $M$? Yes. Suppose to the contrary that $M$ is produced by an execution of the algorithm, and contains a blocking pair $(p,s)$. We consider two cases.

1. $p$ never approached $s$ during execution of the algorithm. Then $p$ must have reached capacity by the end of the algorithm, and $M(p)$ contains only students that $p$ prefers over $s$. Thus $(p,s)$ cannot be a blocking pair, and we get a contradiction.
2. p approached s during execution of the algorithm. Then, since (p,s) is not in M, some professor that s prefers to p must have also approached p at some point in the algorithm. (Note that students only “trade up” their preference lists as the algorithm proceeds, never down.) So, at the end, s must prefer M(s) to p. Again, this implies that (p,s) is not blocking, contradiction.

Since we get a contradiction in both cases, we conclude that M must not have a blocking pair.