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
- Scheduling final exams at a college
- Assign medical residents to hospitals
- 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!

Course Goals

- Learn how to apply the algorithm design process... by practice!
- Learn specific algorithm design techniques
  - Greedy
  - Divide-and-conquer
  - Dynamic Programming
  - Network Flows
- Learn to communicate precisely about algorithms
  - Proofs, reading, writing, discussion
- Prove when no exact efficient algorithm is possible
  - Intractability and NP-completeness
Prerequisites: CS 187 and 250

- Algorithms use data structures
- Familiarity
  - at programming level (lists, stacks, queues, ...)
  - with mathematical objects (sets, lists, relations, partial orders)
- Two key notions to revisit:
  - Recursion: many algorithm design and analysis patterns are based on recursion
  - Proofs: correctness of algorithms. contradiction, induction, ...

Grading Breakdown

- Participation (10%): Discussion section, in-class iClicker questions
- Homework (30%): Homework (every two weeks, usually due Thursday) and online quiz (every weekend due Monday).
- Midterm 1 (20%): Focus on ~first third of lectures. Thursday, Feb 21, 7pm
- Midterm 2 (20%): Focus on ~second third of lectures. Thursday, Apr 11, 7pm
- Final (20%): Covers all lectures. Thursday, May 8, 3:30pm

Course Information

Course websites:

- people.cs.umass.edu/~sheldon/teaching/cs311/ Slides, homework, course information, pointers to all other pages
- moodle.umass.edu Quizzes, solutions, grades
- piazza.com Discussion forum, contacting instructors and TA’s
- gradescope.com Submitting and returning homework

Announcements: Check UMass email / Piazza regularly for course announcements.

Policies

- Online Quizzes: Quizzes must be submitted before 8pm Monday. No late quizzes allowed but we’ll ignore your lowest scoring quiz.
- Homework: Submit via Gradescope by 11:59pm on due date.
  - Late up to 24 hours: 50% penalty
  - Late more than 24 hours: no credit
  - Each student is allowed to submit one homework up to 24 hours late without penalty.

Collaboration and Academic Honesty

- Homework: Collaboration OK (and encouraged) on homework, but read/attempt on your own first. The writeup and code must be your own. Looking at written solutions that are not your own (other students, web) is considered cheating. There will be formal action if cheating is suspected. You must list your collaborators and any printed or online sources at the top of each assignment.
- Online Quizzes: Should be done entirely on your own although it’s fine to consult the book and slides as you do the quiz. Again, there’ll be formal action if cheating is suspected.
- Discussions: Groups for the discussion section exercises will be assigned randomly at the start of each session. You must complete the discussion session exercise with your assigned group.
- Exams: Closed book and no electronics. Cheating will result in an F in the course.

Stable Matching Problem...
Stable Matching and College Admissions

- Suppose there are $n$ colleges $c_1, c_2, \ldots, c_n$ and $n$ students $s_1, s_2, \ldots, s_n$.
- Each college has a ranking of all the students and each student has a ranking of all the colleges. For simplicity, suppose each college can only admit one student.
- Can we match students to colleges such that everyone is happy?
- Not necessarily, e.g., if UMass was everyone’s top choice.
- Can we match students to colleges such that matching is stable? (economist’s view of “good”)
- Stability: Don’t want to unmatched college-student pairs to be incentivized to deviate from matching.

Problem Formulation

- Input: preference lists for $n$ colleges and $n$ students
- Output: need definitions first
- Matching: set $M$ of college-student pairs, each college/student participate in at most one pair.
- Perfect matching: each college/student in exactly one pair
- Instability or unstable pair (with respect to matching $M$): a pair $(c, s) \notin M$ such that
  - $(c, s') \in M$ but $c$ prefers $s$ to $s'$
  - $(c', s) \in M$ but $s$ prefers $c$ to $c'$
- Stable matching: perfect matching with no instabilities
- Output: a stable matching

Clicker Question 1

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>Xavier</td>
<td>Zeus</td>
</tr>
<tr>
<td>Boston</td>
<td>Yvette</td>
<td>Xavier</td>
</tr>
<tr>
<td>Chicago</td>
<td>Xavier</td>
<td>Yvette</td>
</tr>
</tbody>
</table>

Which pair is an unstable pair with respect to the matching $\{A - X, B - Z, C - Y\}$? (marked in bold above)

A: A - Y
B: B - X
C: B - Z
D: none of the above

Examples

Do stable matchings always exist? Are they unique? Let’s see…

Example 1: universal prefs

<table>
<thead>
<tr>
<th>Colleges</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: 1 2</td>
<td>1: a b</td>
</tr>
<tr>
<td>b: 1 2</td>
<td>2: a b</td>
</tr>
</tbody>
</table>

- $M = \{(a, 1), (b, 2)\}$: stable
- $M = \{(a, 2), (b, 1)\}$: not stable

Example 2: inconsistent prefs

<table>
<thead>
<tr>
<th>Colleges</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: 1 2</td>
<td>1: b a</td>
</tr>
<tr>
<td>b: 2 1</td>
<td>2: a b</td>
</tr>
</tbody>
</table>

Clicker Q2: You are given an arbitrary set of preferences. Does it have more than one stable matching?

A. Yes
B. No
C. It depends on the preference lists

- $M = \{(a, 1), (b, 2)\}$: stable
- $M = \{(a, 2), (b, 1)\}$: stable
Toward an Algorithm

Let’s use a slightly bigger example to try to develop an algorithm.

<table>
<thead>
<tr>
<th>Colleges</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: 1 2 3</td>
<td>1: c a b</td>
</tr>
<tr>
<td>b: 2 1 3</td>
<td>2: a b c</td>
</tr>
<tr>
<td>c: 1 3 2</td>
<td>3: a b c</td>
</tr>
</tbody>
</table>

Idea: build $M$ incrementally. What should colleges do? What should students do?

Propose-and-Reject (Gale-Shapley) Algorithm

Initially all colleges and students are free

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

Choose such a college $c$
Let $s$ be the highest ranked student to whom $c$ has not proposed

if $s$ is free then

c and $s$ become matched

else if $s$ is matched to $c'$ but prefers $c$ to $c'$ then

c' becomes unmatched

c and $s$ become matched

else

\[ s \text{ prefers } c' \]

$s$ rejects $c$ and $c$ remains free

end if

end while

Analyzing the Algorithm

- Some natural questions:
  - Can we guarantee the algorithm terminates?
  - Can we guarantee the every college and student gets a match?
  - Can we guarantee the resulting allocation is stable?
- Some initial observations:
  - (F1) Once matched, students stay matched and only "upgrade" during the algorithm.
  - (F2) College propose to students in order of college’s preferences.

Can we guarantee the algorithm terminates?

- Yes! Proof...
- Note that in every round, some college proposes to some student that they haven’t already proposed to.
- $n$ colleges and $n$ students $\implies$ at most $n^2$ proposals
- $\implies$ at most $n^2$ rounds of the algorithm

Can we guarantee all colleges and students get a match?

- Yes! Proof by contradiction...
  - Suppose not all colleges and students have matches. Then there exists unmatched college $c$ and unmatched student $s$.
  - $s$ was never matched during the algorithm (by F1)
  - But $c$ proposed to every student (by termination condition)
  - When $c$ proposed to $s$, she was unmatched and yet rejected $c$. Contradiction!

Clicker Question 3

Depending on the problem instance, which of the following can happen during a run of the Gale-Shapley algorithm?

A: Each student accepts their first offer and never switches.
B: Some student switches their choice more than once during a run.
C: Both A and B can happen for the same problem instance.
D: Both A and B can happen, but only in different problem instances.
Can we guarantee the resulting allocation is stable?

- Yes! Proof by contradiction
  - Suppose there is an instability \((c, s)\)
    - \(c\) is matched to \(s'\) but prefers \(s\) to \(s'\)
    - \(s\) is matched to \(c'\) but prefers \(c\) to \(c'\)
  - Did \(c\) offer to \(s\)? Yes, by (F2), since \(c\) offered to \(s'\) who is ranked lower
  - Did \(s\) accept offer from \(c\)? Maybe initially, but \(s\) must eventually reject \(c\) for another college, and, by (F1), \(s\) prefers final college \(c'\) to \(c\)
  - Contradiction!

For Next Time

- Think about:
  - Would it be better or worse for the students if we ran the algorithm with the students proposing?
  - Can a student get an advantage by lying about their preferences?

- Read: Chapter 1, course policies

- Enroll in Piazza, log into Moodle, and visit the course webpage.

A modern application

Content delivery networks. Distribute much of world's content on web.

User. Preferences based on latency and packet loss.
Web server. Preferences based on costs of bandwidth and co-location.
Goal. Assign billions of users to servers, every 10 seconds.

Algorithmic Nuggets in Content Delivery

Bruce M. Maggs
Duke and Akamai
ramesh@cs.umass.edu

ABSTRACT

Akamai is one of the largest content delivery networks (CDNs) in the world. Distributing content is more challenging than ever due to the increasing size and complexity of the Internet. This paper surveys algorithmic research that has been applied to the design of CDNs, and describes the benefits that these algorithms provide. We sketch how much of the content delivery network can be modeled as a many-to-many matching problem, and discuss how sophisticated algorithmic research has been adapted to this problem with great success.

The top-three objectives for the designers and operators of a content delivery network (CDN) are high reliability, low delay, and robustness to failures. In particular, the paper walks through the steps that take place from the Instant that a client's request is matched to a cache. If not, the server begins to query other servers in the routing network, and elect leaders in various contexts. In building one of the largest CDNs, Akamai has experienced a number of challenges. This paper "peeks under the covers" at the subsystems that demonstrate the benefits that these algorithms provide by research into industrial practice. In several instances, we focus on the translation of algorithms that are the fruits of original algorithm designers. Hence, much of this paper concerns and messy details that are not easily captured by the theoretical models, and finally describe what is intellectual creativity is often required to address practical applications.