COMPSCI 311 Section 1: Introduction to Algorithms

Dan Sheldon
University of Massachusetts

Last Compiled: September 2, 2021

CS 311: Intro to Algorithms

- Instructor: Dan Sheldon
- Where: Goessmann Lab 20
- When: T/Th 11:30–12:45
- Discussion Sections: F 10:10–11 Engineering Lab II Room 119, F 11:15–12:05 Hasbrouck 134 (Please stick to assigned section)
- TAs: Zhanna Kaufman, Yunfei Luo, Hasnain Heickal, Sylee Dandekar, Cooper Sigrist
- Office hours: Mon 11–noon (zoom), Thu 1:30–2:30 (CS 246). Complete list on Piazza.

Run jointly with Section 3 (Prof. Barrington): same TAs, HW, quizzes, midterms, moodle, Piazza, Gradescope; different final exam times

COVID-19

- Welcome back! It’s exciting to be in person, but we all need to do our part to slow the spread of COVID-19. Some community members are vulnerable or have vulnerable family members. (My kids are too young to be vaccinated.)
- Wear masks at all times in all CS 311 course meetings
  - Lecture, discussion, office hours
  - Masks available in kiosks if you forget
  - Prior written notice required for medical exemptions
- Complete daily self-check-list, do not come to any meeting if ill
  - Moodle questionnaire for excused absence with no penalty
  - All lectures recorded, pre-recorded videos also available

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.

What is Algorithm Design?

- 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

- Familiarity with:
  - data structures (lists, stacks, queues, ...)
  - mathematical objects (sets, lists, relations, partial orders)
  - recursion: many algorithm design patterns based on recursion
  - proofs: correctness of algorithms. contradiction, induction, ...

Course Information

Course websites:

- people.cs.umass.edu/~sheldon/
teaching/cs311/
- umass.moonami.com
- piazza.com
- gradescope.com

Slides, homework, course information/policies, pointers to all other pages

Quizzes, solutions, grades
Discussion forum, contacting instructors and TA's (course staff will enroll you)
Submitting and returning homework (signup code on Moodle)

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

A Week in the Life of CS 311

Mon Moodle quiz due 8pm (lecture, reading engagement)
Tue Lecture (iClicker)
Wed Midterm exams on 9/22, 10/20, 11/10
Thu Lecture (iClicker)
Fri Discussion section in morning (worksheets), homework and/or challenge problems due 11:59pm

Weekly Homework (Gradescope Online Assignments)

- due most Fridays. HW 1 posted, due 9/10
- focused on specific learning goal mastery
  (see detailed learning goals on course page)
- midterms will look similar
- Collaboration: OK to ask for help on how to solve a problem, but do them on your own. Copying, sharing, or viewing any solutions that are not your own is a violation of course policy (and you won’t learn what you need to know for midterms)
**Challenge Problems**

**COMPSCI 311: Introduction to Algorithms**  
Fall 2021

Challenge Problems 1  
due 9/17/2021 at 11:59pm in Gradescope

Instructions. Limited collaboration is allowed while solving problems, but you must write solutions yourself. Let collaboration on your assignment.

You can choose which problems to complete, but must submit at least one problem per week. See the course page for information about how challenge problems are graded and contribute to your homework grade. Since you don’t need to complete every problem, you are encouraged to focus your efforts on producing high-quality solutions to the problems you feel confident about. There is no benefit to guessing or writing vague answers.

If you are asked to design an algorithm, please: (a) give a precise description of your algorithm using either pseudocode or language, (b) explain the intuition of the algorithm, (c) justify the correctness of the algorithm; give a proof if needed, (d) state the running time of your algorithm, (e) justify the running-time analysis.

Submissions. Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. Please assign pages to questions in Gradescope.

Problem 1. Stable Matching Running Time. In class, we saw that the Propose-and-reject algorithm terminates in at most n^2 iterations, when there are n students and n colleges.

**Grade Criteria**

A Complete at least 12 challenge problems with ✓ or better; including at least 6 with ✓+
B Complete at least 8 challenge problems with ✓ or better; including at least 4 with ✓+
C Complete at least 6 challenge problems with a ✓
D Attempt at least one challenge problem on each assignment

• Don’t need to complete every problem, so focus on high-quality solutions to ones you can solve
• No benefit to guessing, vague answers

**Grading Breakdown**

• Participation (10%): discussion section, in-class iClicker questions
• Moodle quizzes (5%): weekly, engagement with course material
• Homework (10%): ~8–10 weekly assignments
• Challenge problems: (20%): 6 assignments, roughly bi-weekly
• Challenge problem self-assessments (5%): Review solutions and post self-assessment of your challenge problems solutions after due date
• Midterms 1, 2, 3 (10% each): each covers about one quarter of the course
• Final (20%): covers all course materials

**Late Policies**

• Online Quizzes: Quizzes must be submitted before 8pm Monday. No late quizzes allowed but we’ll ignore your lowest scoring quiz.
• Homework and challenge problems: Submit via Gradescope by 11:59pm on due date.
  • Late: no credit (after short grace period)
  • Each student is allowed to submit three assignments up to 24 hours late without penalty

**Challenge Problems**

• Solutions typed or written neatly and uploaded to Gradescope as high-quality pdf
• Usually involve designing an algorithm and proving it correct
• Choose which problems to submit (at least one per assignment)
• Graded as one of x, ✓–, ✓, or ✓+ using rubric on course web page.
  • ✓ and ✓+ indicate mastery (fairly high standards)
  • contribute to grade as follows

• Collaboration OK (e.g. discuss problem, generate ideas, work on whiteboard), but read/attempt on your own first. The written solution 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. List collaborators and any printed or online sources at the top of each assignment.
Collaboration and Academic Honesty

- **Homework and challenge problems**: see above
- **Online Quizzes**: Should be done entirely on your own although it’s fine to consult the book and slides as you do the quiz.
- **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.
- Formal action will be pursued for suspected cheating. Penalty may be an F in course.
- If in doubt whether something is allowed, ask!

---

Stable Matching Problem

Matching applicants to medical residency programs:

- m applicants
- n slots at hospitals
- Applicants have preferences over hospitals and vice versa
- National Resident Matching Program (nrmp.org) makes matches

**What is a “good” way to match applicants to programs?**

- economists: matching should be **stable**. no incentive to switch
- Gale-Shapley algorithm

---

Examples

Do stable matchings always exist? Are they unique?

**Example 1**: universal prefs

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

Which pair is an instability (unstable pair) with respect to the matching \{ (a,1), (b,3), (c,2) \} (marked in **bold** above)

- A. (a, 2)
- B. (b, 1)
- C. (b, 3)
- D. none of the above

---

Problem formulation (colleges and students)

**Input:**
- n colleges
- n students
- preference lists

**Output**: a **stable** matching. But what does this mean?

**Matching:**
- assignment students to colleges
- set M of college-student pairs, each college/student in one pair

**Instability or unstable pair** in a matching: an unmatched pair that prefer each other to their assigned matches

**Stable matching**: matching with no instabilities

**Goal**: output a stable matching
Examples

Example 2: inconsistent prefs

Colleges

| a | 1 | 2 |
| b | 2 | 1 |

Students

| 1 | b | a |
| 2 | a | b |

Clicker: Which matching is stable?

A. $M = \{(a, 1), (b, 2)\}$
B. $M = \{(a, 2), (b, 1)\}$
C. neither
D. both

▶ Answer: D, both are stable
▶ Fact: there can be multiple stable matchings

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 made offers to every student do

Choose such a college $c$

Let $s$ be the highest ranked student to whom $c$ has not offered

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 $s$ prefers $c'$

$s$ rejects $c'$ and $c$ becomes unmatched

end if

end while

Analyzing the Algorithm

Goal: prove that the algorithm always returns a stable matching

Initial observations:

▶ (F1) Students accept their first offer, after which they stay matched and only “upgrade” during the algorithm
▶ (F2) Colleges propose to students sequentially in order of preferences.

Propose-and-Reject (Gale-Shapley) Algorithm

Initially all colleges and students are free

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

Choose such a college $c$

Let $s$ be the highest ranked student to whom $c$ has not offered

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 $s$ prefers $c'$

$s$ rejects $c'$ and $c$ becomes unmatched

end if

end while

Termination

Does the algorithm terminate?

| a | 1 | 2 | 3 |
| b | 2 | 1 | 3 |
| c | 1 | 3 | 2 |

▶ in each round, some college proposes to a new student in their list (by F2)
▶ at most $n^2$ proposals $\implies$ at most $n^2$ rounds
Are all colleges and students matched?

Yes. Suppose, for contradiction that college \( c \) and student \( s \) are unmatched at the end of the algorithm.

- \( s \) was never matched during the algorithm (by F1)
- But \( c \) proposed to every student (by F2 and termination condition)
- When \( c \) proposed to \( s \), she was unmatched and yet rejected \( c \). Contradiction!

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?
- Read: Chapter 1, course policies
- Enroll in Piazza, log into Moodle, and visit the course webpage.