COMPSCI 311 Section 1: Introduction to Algorithms

Dan Sheldon

University of Massachusetts

{Last Compiled: February 4, 2025}

CS 311: Intro to Algorithms

Please sign up at classquestion.com/students with class code XZCDN

- **Instructor**: Dan Sheldon
- **Where**: ILC S131
- **When**: M/W 2:30–3:45pm
- **Discussion Sections**: F 9:05–9:55 LGRC A104A, F 10:10–11:00 LGRT 121, F 12:20–1:10 Ag Engineering 119, F 1:25–2:15 CS 145 (Please stick to assigned section)
- **TAs**: Md Abdual Aowal, Miguel Fuentes, Purna Dutta
- **Office hours**: TBA, see list on Campuswire

Run jointly with Section 2 (Prof. Minea): same TAs, HW, exams, Canvas, Campuswire, Gradescope

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
  - $s_1 =$ AGGCTACC
  - $s_2 =$ 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 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: CICS 210 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/	Slides, homework, **course information/policies, pointers to all other pages**
classquestion.com	In-class "clicker" questions
Canvas: umamherst.instructure.com	Solutions, grades
campuswire.com	Discussion forum, contacting instructors and TAs
gradescope.com	Submitting and returning homework

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

A Week in the Life of CS 311

Mon	Lecture (classquestion), weekly homework due 11:59pm
Wed	Lecture (classquestion)
Thu	Challenge problems (due 11:59pm) OR midterm exam 7–9pm
Fri	Discussion section (worksheets, submit by 6pm)

Weekly Homework (Gradescope Online Assignments)

HW 1: Stable Matchings and Big-O Proofs

STUDENT NAME

Search students by name or email...

Q1 Stable Matching Example

2 Points

Consider the following instance of the stable matching problem with colleges {A, B, C} and students {1, 2, 3}.

College	Preference list
A	1, 2, 3
B	2, 1, 3
C	3, 2, 1

Student	Preference list
1	B, A, C
2	A, B, C
3	C, B, A

Q1.1 First Stable Matching

1 Point

Find the stable matching returned by the Gale-Shapley algorithm.

College A is matched to:

- ○ 1
- ○ 2
- ○ 3

Weekly Homework (Gradescope Online Assignments)

- due most Mondays. HW 1 released tomorrow, due Monday 2/10
- 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 (including AI) 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 2022**

Challenge Problems 1

due 9/21/2022 at 11:59pm in Gradescope

Instructions. Limited collaboration is allowed while solving problems, but you must write solutions yourself. List collaborators on your submission.

You can choose which problems to complete, but must submit at least one problem per assignment. 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.

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, $\checkmark-$, $\checkmark$, or $\checkmark+$ using rubric on course web page.
  - $\checkmark$ and $\checkmark+$ indicate mastery (fairly high standards)
  - contribute to grade as follows

Grade	Criteria
A+	Complete at least 14 challenge problems with ✓ or better; including at least 7 with ✓+
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	Complete at least 6 challenge problems with a ✓- or better; including at least 3 with a ✓

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

- **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, AI) 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.

Grading Breakdown

- **Participation** (10%): discussion section (7%), lecture participation via classquestion (3%)
- **Homework** (12.5%): ~8–10 weekly assignments
- **Challenge problems**: (25%): 6 assigments, roughly bi-weekly
- **Challenge problem self-assessments** (2.5%): 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

- *Homework and challenge problems:* Submit via Gradescope by 11:59pm on due date.
  - Late: no credit
  - Each student is allowed to submit *three* assignments up to 24 hours late without penalty
  - At most one late day per assignment (solutions posted after 24 hours)

Collaboration and Academic Honesty

- *Homework and challenge problems*: see above (**no AI!**)
- *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:

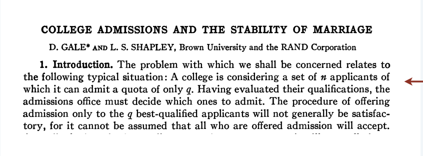
- m applicants
- n slots at hospitals
- Applicants have preferences over hospitals and vice versa
- National Resident Matching Program (nrmp.org) makes matches

What is a "good" way to match applicants to programs?

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

2012 Nobel Prize in Economics

Lloyd Shapley. Stable matching theory and Gale–Shapley algorithm.



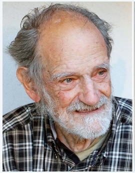
COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only q. Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the q best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept.

original applications:
college admissions and
opposite-sex marriage

Alvin Roth. Applied Gale–Shapley to matching med-school students with hospitals, students with schools, and organ donors with patients.



Lloyd Shapley

Alvin Roth

slide credit: Kevin Wayne / Pearson

Problem formulation (colleges and students)

Input:

- n colleges
- n students
- preference lists

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

Matching:

- assignment of 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

Clicker

Colleges

a:	**1**	2	3
b:	2	1	**3**
c:	1	**2**	3

Students

1:	b	**a**	c
2:	a	b	**c**
3:	a	**b**	c

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

Examples

Do stable matchings always exist? Are they unique?

Example 1: universal prefs

Colleges

a:	1	2
b:	1	2

Students

1:	a	b
2:	a	b

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

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.

Colleges

a:	1	2	3
b:	2	1	3
c:	1	3	2

Students

1:	c	a	b
2:	a	b	c
3:	a	b	c

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

Summary

- Unmatched colleges take turns offering to students and propose in order of preference
- Students tentatively accept first offer and then “trade up” if they receive better offers

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 $\triangleright$ s prefers c'
    s rejects c and c remains free

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.

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 $\Longrightarrow$ 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
- Visit course webpages: canvas, Campuswire, etc.