CS 312: Algorithms

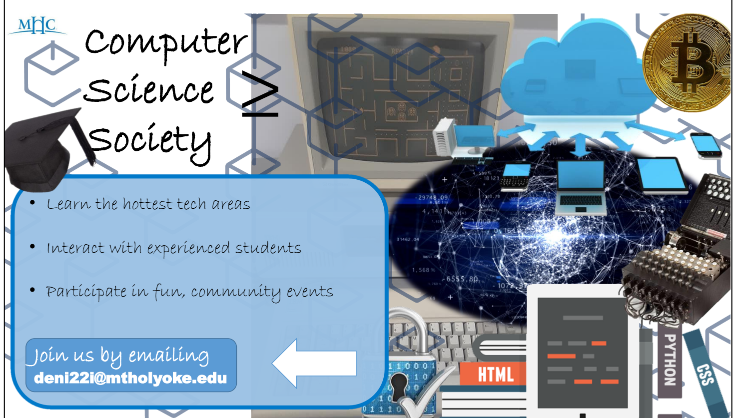
Dan Sheldon

Mount Holyoke College

Last Compiled: September 5, 2018

CS 312: Algorithms

- **Instructor**: Dan Sheldon
- **Where**: Clapp 306
- **When**: M/W 2:55-4:10
- **Fourth Hour**: Friday 2:55-3:45
- **TAs**: Vivian Le, Nhu Do, Jessica Feng, Steven Hill
- **Office hours**:
  - Dan: Mon 4-5, Wed 12:15-1:15
  - TAs: TBD



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

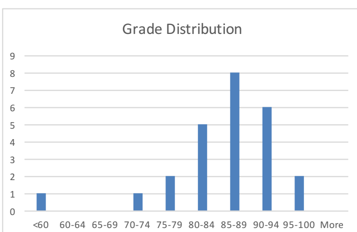
- Learn how to apply the algorithm design process. . . by practice!
- Learn specific algorithm design techniques
  - Greedy
  - Divide-and-conquer
  - Dynamic Programming
  - Network Flows
- Learn to communicate precisely about algorithms
  - Proofs, reading, writing, discussion
- Prove when no exact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

Grading Breakdown

- **Participation (10%):** Fourth Hour, lecture participation, Piazza
- **Quizzes (10%):** Online Moodle quizzes (weekly due Monday 8pm)
- **Homework (35%):** Homework (weekly due Thu)
- **Midterm 1 (15%):** Take-home, right before spring break
- **Midterm 2 (15%):** Take-home, ~first week in April
- **Final (15%):** Take-home, exam period

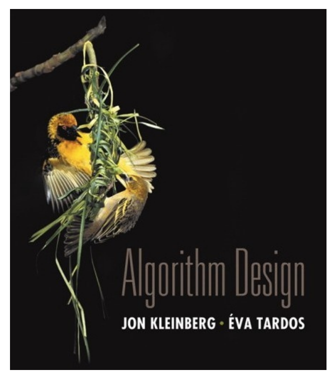
Grading

Grade distribution from Spring 2017:



Median grade = B. 32%: A-/A. 52% B-/B/B+. 16%: C+ or below

Required Textbook



Course Information

Course websites:

people.cs.umass.edu/~sheldon/teaching/cs312/	Slides, homework, course information, **pointers to all other pages**
moodle.mtholyoke.edu	Quizzes, solutions
piazza.com	Discussion forum, contacting instructors and TA's
gradescope.com	Submitting and returning homework

Announcements: Check your email daily and log into Piazza regularly for course announcements.

Policies

- **Online Quizzes:** Submit before 8pm Monday. Two attempts. No late submissions. Lowest is dropped.
- **Homework:** Submit via Gradescope. Points for readability. Late penalties: 0–24 hours = 33%; 24–48 hours = 66%; 48+ hours = no credit. **Three free late days**.

Collaboration and Academic Honesty

- **Homework:** Collaboration encouraged, but read/attempt on your own first. Writeup **must** be your own. List collaborators and any sources beyond notes or textbook at the top of each assignment.
- Honor code violations:
  - **Collaborating to write solutions**
  - **Looking at another student's solutions**
  - **Sharing your written solutions**
  - **Use of solutions to same or similar problems found online or elsewhere**
- I will refer **every** suspected violation to the academic honor board, no matter how "minor".

Collaboration and Academic Honesty

- **Quizzes:** Do entirely on your own. Use book, slides, notes. (I can see on Moodle when each student begins and ends the quiz.)
- **Fourth Hour:** Groups assigned randomly each session. Complete exercises with group.
- **Exams:** Take-home, open book and notes. No collaboration or outside sources (e.g. web).
- If in doubt whether something is allowed, ask!

Tips

- Start HW early! Talk to me, TAs, other students. Establish a weekly routine.
- Solving algorithms problems is a process. Solve over several days in small chunks, with discussion.
  - Failure mode 1: attempt to complete in one sitting the night before HW is due
  - Failure mode 2: sit and puzzle for many hours by yourself until you solve it
  - Failure mode 3: read the problem, think for 5 minutes, declare it is too hard
- Solving problems and writing great proofs is a skill. Expect to practice and improve throughout the semester. It's normal to feel uncertain at first, especially about proofs.

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 ranks all students and each student ranks all colleges. For simplicity, suppose each college can only admit one student. Example.
- Can we match students to colleges such that everyone is *happy*?
  - Not necessarily, e.g., Mount Holyoke is everyone's top choice.
- Can we match students to colleges in a *stable* way?
  - *Stable:* Don't match (c, s) and (c', s') if c and s' would both prefer to be matched with each other. Precise definition + examples on board.
  - Yes! And there's an efficient algorithm to find that matching.
  - Develop algorithm informally

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 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 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...
  - 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!

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'
  - By (F2), c must have proposed to s before proposing and becoming matched to s'
  - Since s isn't matched to c at the end of the algorithm, she must have rejected c's offer (either immediately or upon receiving a better proposal). By (F1), she prefers her final match 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
- Log into Moodle / Piazza, visit the course webpage.