CMPSCI 311 Section 2: Introduction to Algorithms

Dan Sheldon

University of Massachusetts

Last Compiled: January 23, 2017

CS 311: Intro to Algorithms

- **Instructor**: Dan Sheldon
- **Where**: Goessmann Lab Room 20
- **When**: M/W 4:00-5:15
- **Discussion Sections**: Tuesday 4-4:50, Thursday 4-4:50, Hasbrouck Lab Add room 113. (Please stick to assigned section)
- **TA**: Pardis Malekzadeh
- **Office hours**:
  - Dan: 2-3pm Thursday in CS 246
  - Pardis: 10-11am Thursday, Friday in LGRC room A303

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%):** Discussion assignments and other contributions.
- **Homework (30%):** Homework (every two weeks due Friday) and online quiz (every weekend due Monday).
- **Midterm 1 (20%):** Focus on first third of lectures. 7pm Thursday, March 2
- **Midterm 2 (20%):** Focus on second third of lectures. 7pm Thursday, April 13
- **Final (20%):** Covers all lectures. 3:30pm Friday, May 5th

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 your UMass email daily and log into 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:* To be submitted via Gradescope. A 50% penalty will be applied for homework that is late by 5 minutes up to 24 hours. Homework that is late by more than 24 hours receives no credit. However, 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 is considered cheating. There'll 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.
- If in doubt whether something is allowed, ask!

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 that they could admit 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*?
  - *Stability:* Don't want to match c with s and c' with s' if c and s' would prefer to switch to being matched with each other.
  - Yes! And there's an efficient algorithm to find that matching.

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 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!

Can we guarantee the resulting allocation is stable?

- Yes! Proof by contradiction with a case analysis...
  - 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'
  - Case 1: c offered to s
    - Since s isn't matched to c at the end of the algorithm, she must have rejected c's offer at some point and therefore be matched to a college she prefers to c (by F1). Contradiction.
  - Case 2: c did not offer to s
    - We know c proposed to and was matched to s'. Since s' is less preferred, c must have also proposed to s (by F2). Contradiction. (This case cannot happen.)

For Wednesday

- 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.