CMPSCI 311: Introduction to Algorithms

Akshay Krishnamurthy

University of Massachusetts

Last Compiled: February 13, 2018

CS 311: Intro to Algorithms

- **Instructor**: Akshay Krishnamurthy
- **Where**: Goessman Laboratory 64
- **When**: MW 4:00-5:15
- **Discussion Sections**: Fridays 10:10, 12:20, 1:25 (Please stick to assigned section)
- **TAs**: Jesse Lingeman and Russell Lee
- **Office hours**:
  - Akshay: 530pm Monday in CS 258.
  - Jesse: 1pm Tuesday in CS 207.
  - Russell: 2pm Wednesday in CS 207.

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**: Verify that the algorithm is correct
  - with a proof
- **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 (25%):** Homework (every two weeks due Wednesday) and online quiz (every weekend due Monday).
- **Midterm 1 (20%):** Focus on first third of lectures. 7pm Wednesday 28th February.
- **Midterm 2 (20%):** Focus on second third of lectures. 7pm Wednesday 11th April.
- **Final (25%):** Covers all lectures. 3:30pm Friday 4th May.

Course Information

Course websites: Slides, homework, course information:

people.cs.umass.edu/~akshay/courses/cs311/

Quizzes, solutions, grades will be posted at:

moodle.umass.edu

Forums for discussion and contacting instructors and TA's:

piazza.com

Homework submissions:

https://gradescope.com/courses/14832

Fill out consent form on Piazza.

Policies

- *Announcements:* Check your UMass email daily and log into Piazza regularly for course announcements.
- *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, always due 11:59pm on a Wednesday. Submit a pdf.
- *Late policy:* Up to half credit for any homework that is late by up to 24 hours. Homework later than 24 hours receives no credit.

Collaboration and Academic Honesty

- *Homework:* You may discuss homework with other students but the writeup must be your own. Any suspicion of cheating will be raised with the university. You must list your collaborators and any printed or online sources at the top of each assignment.
- Homeworks are *hard*! Start early. Come to office hours!
- *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
```

An example

$$\text{Colleges: } \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \\ 2 & 1 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix} \qquad \text{Students: } \begin{pmatrix} 4 & 3 & 1 & 2 \\ 4 & 3 & 1 & 2 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}$$

Solution: $(3,1),(4,2),(1,3),(2,4)$

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:
  - College propose to students in order of college's preferences.
  - Every student only “upgrades" during the algorithm.

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.
  - Any college can propose to at most n different students and there are only n colleges.
  - There are at most n^2 different 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.
  - When algorithm terminates c has proposed to every student.
  - Student s was never matched during the algorithm.
  - When c proposed to student s, s 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 exists colleges c and c' and students s and s' such that c is matched to s' and c' is matched to s but c and s would prefer to be matched to each other.
  - Case 1: s got an offer from c
    - Since s isn't matched to c at the end of the algorithm, they must have rejected c's offer at some point and must therefore be matched to a college they prefer.
  - Case 2: s never got an offer from c
    - c must have been matched to a student that they prefer.
  - Either way, at least one of c or s would not want to be matched to the other more than their current assignment.

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 (fill out consent form!), log into Moodle, Gradescope, and visit the course webpage.