CS 361, Lecture 1

Jared Saia
University of New Mexico

Who am I?

- 1st year as a Professor at UNM, PhD last year from University of Washington. Master's degree from UNM.
- My research area is algorithms, My recent focus is on algorithms for creating p2p networks which are provably censorship resistant (check out my web page for papers)
- Call me Jared, Professor Saia, or, if you must, Herr Doktor Saia

Today's Outline

- Administrative Info
- Course Overview
- What is an Algorithm? Data Structure?
- Why study algorithms?
- Todo list for next class

What you should know

- concept of abstract data type (ADT)
- good programming skills in C, C++, or Java
- 2 semesters of calculus (infinite sums and limits, derivatives, etc)

CS361: Algorithms and Data Structures

- Instructor: Jared Saia, FEC 301H (Third floor of FEC, on the side facing Central Avenue)
- Phone: 277-5446
- Office Hours: Tuesdays 10:45-11:45, Fridays 2-3pm, or by appointment. Note that Tuedays and Fridays are the best time to schedule an appointment, other days I'm less likely to be in my office.
- TA: Kanglin Xu, Office Hours: TBA

What you will learn

- Learn to analyze and improve algorithms (both to improve time complexity and to prove correctness)
- Learn to see and solve problems abstractly
- Learn many data structures, some of which you will almost certainly use again in your career
- Learn many algorithms, some of which you will almost certainly use again in your career

What you will learn (details)

- Asymptotic analysis and recurrence relations (mathematical tools for finding the run time of an algorithm)
- Intermediate level data structures (heaps, balanced trees, hash tables)
- Intermediate level algorithms (sorting, introduction to graphs)
- Basic knowledge of how to construct formal proofs regarding algorithm efficiency and correctness
- How to code up intermediate algorithms and data structures, how to design experiments to compare efficiencies of implementations

Working in groups

- The class project, homeworks, and in-class exercises will be done in groups
- I'll divide class up into groups based on pretest (and your preferences)
- If necessary, you can switch groups or work on your own.
- For group assignments, I'll ask you to grade the performance of your fellow group members on that assignment.

Course Assessment

Grading:

- Class Project, 30%
- Homeworks, 30%
- Midterm and Final, 30% (20% final and 10% midterm)
- Class Participation, 10%

Why work in groups?

- Many studies show that students at all levels learn significantly better by working in groups (well-prepared students learn by teaching, less-prepared students learn by peer instruction)
- When you get a job, you will almost certainly work in a group, so working with a group is a good skill to have
- Will be more fun

Note on Homeworks

- Homeworks and the project are due at the *beginning* of class, and should be turned in on the table in front of me.
- Late hws will *not* be accepted without prior approval from Prof. Saia (easy to do for one, harder for two, very difficult for three)
- Put pages of hw *in order*. We don't care what order you solve the hw in, but before you turn it in, you must put the problems in order (this makes grading easier)
- *Staple* hws, do not use paper clips, folding, tape, putty, gum, etc. Prof Saia and Kanglin do not bring staplers to class, so make sure you staple the hw before class (stapling helps us keep together all pages of your hw)

Class Participation

- I expect you to show up for 90% of the classes
- There will be frequent in-class exercises
- You can work on these with your group
- At the end of these, I will call on a random member of a random group
- What I expect is either a correct answer, a partial answer, an idea on how to approach the problem, or a reasonably intelligent question about the problem.

What is an Algorithm?

- ???

Why study algorithms?

"Seven years of College down the toilet" - John Belushi in Animal House

- Q: Can I get a programming job without knowing something about algorithms and data structures?
- A: Yes, but do you really want to be programming GUIs your entire life?

What is an Algorithm?

- Mirriam-Webster says: "A step-by-step procedure for solving a problem or accomplishing some end especially by computer"
- Q: What's the difference between an algorithm and a program?
- A: An algorithm is a sequence of high-level, language independent operations (a mathematical recipe). A program is a sequence of instructions in a specific programming language.

Why study algorithms? (II)

- Almost all big companies want programmers with knowledge of algorithms: Microsoft, Google, Oracle, IBM, Yahoo, Sandia, Los Alamos, etc.
- In most programming job interviews, they will ask you several questions about algorithms and/or data structures
- Your knowledge of algorithms will set you apart from the large masses of interviewees who know only how to program
- If you want to start your own company, you should know that many startups are successful because they've found better algorithms for solving a problem (e.g. Google, Akamai, etc.)

What is a Data Structure?

- A specific way in which data is stored in a computer e.g. a linked list
- Q: How is this different from an ADT? (recall: an example ADT is a stack)
- A: An ADT is mathematical description of a data object and the set of operations defined on the object
- Q: Can we analyze the run time of operations on an ADT? a data structure?

Why Study Algorithms? (III)

- You'll write better, faster code
- You'll learn to think more abstractly and mathematically
- It's the most challenging and interesting area of CS!

A Real Job Interview Question

The following is a question commonly asked in job interviews in 2002 (thanks to Maksim Noy, see my web page for the full compilation of questions):

- You are given an array with integers between 1 and $1,000,000$.
- All integers between 1 and $1,000,000$ are in the array at least once, and one of those integers is in the array twice
- Q: Can you determine which integer is in the array twice? Can you do it while iterating through the array only once?

Solution

- Ideas on how to solve this problem?? What if we allowed multiple iterations?

Naive Algorithm

- Create a new array of ints between 1 and $1,000,000$, which we'll use to count the occurences of each number. Initialize all entries to 0
- Go through the input array and each time a number is seen, update its count in the new array
- Go through the count array and see which number occurs twice.
- Return this number

Naive Algorithm Analysis

- Q: How long will this algorithm take?
- A: We iterate through the numbers 1 to $1,000,000$ *three* times!
- Note that we also use up a lot of space with the extra array
- This is wasteful of time and space, particularly as the input array gets very large (e.g. it might be a huge data stream)
- Q: Can we do better?

Ideas for a better Algorithm

- Note that $\sum_{i=1}^{n} i = (n+1)n/2$
- Let S be the sum of the input array
- Let x be the value of the repeated number
- Then $S = (1,000,000+1)1,000,000/2 + x$
- Thus $x = S - (1,000,000+1)1,000,000/2$

A better Algorithm

- Iterate through the input array, summing up all the numbers, let S be this sum
- Let $x = S - (1,000,000+1)1,000,000/2$
- Return x

Analysis

- This algorithm takes iterates through the input array just once
- It uses up essentially no extra space
- It is at least three times faster than the naive algorithm
- Further, if the input array is so large that it won't fit in memory, this is the only algorithm which will work!
- These time and space bounds are the best possible

Take Away

- Designing good algorithms matters!
- Not always this easy to improve an algorithm
- However, with some thought and work, you can *almost always* get a better algorithm than the naive approach

Todo

- Work on pretest, due next Tuesday!
- Visit the class web page: www.cs.unm.edu/~saia/361/
- Sign up for the class mailing list (cs361)
- Start reading Chapter 1 in textbook