Welcome to Com S 311 C
Steve?

- 25 years teaching Math and CS
  - At ISU since 2008
- 8 years industry, big variety of projects
- Last taught 311 about 5 years ago
  - taking a break from 227 😊
  - 311 is the most fun (and important) course in the curriculum
Why you should switch to section D

• Better time (TR 2:10)
• More reasonably-sized room
• Fewer people, more congenial
• Section D instructor is smarter and better organized than Steve
Some examples

• Interval scheduling
  – Brute force?
  – Earliest start time?
  – Shortest first?
  – Earliest end time?

• (See Chapter 1 of Skiena for details)
Moral?

- We need to be able to *reason* about correctness!
  - i.e., proof
Why not just rely on unit testing?

• may not be clever enough to think of counterexamples
• difficult to validate test cases by hand
• have to invest time in implementation first, before knowing whether algorithm is correct!
Some examples

• Robot arm tour
  – Nearest neighbor?
  – Closest pair?
  – Brute force?

• (See Chapter 1 of Skiena for details)
Moral?

• We need to be able to recognize problems that are *intractable*
  – Don’t try to write an exact solution for an intractable problem, concentrate on approximation strategies
  – There are many common problems in this category
Some examples

• Sorting
  – Selection sort vs merge sort

• SumOfTwo
  – Brute force
  – Using binary search
  – Linear
Moral?

• Efficiency matters to your life
  – Scalability is important!

• Need to be able to analyze asymptotic (”Big-O”) running time
Why not just measure wall clock time?

• can be misleading (e.g. dependent on memory or cache size)
• how do you know whether your algorithm is flawed, or your implementation is flawed?
• have to invest time in implementation first
• Textbook?
  – Kleinberg
  – Skiena
  – Corman
• Syllabus?
• Recitations?
Summary of course objectives

• Know a set of “standard” algorithms (and related data structures) and be able to use them to model real-world problems.

• Design algorithms based on common techniques (greedy, divide and conquer, dynamic programming).

• Reason about correctness, either by proof or counterexample.

• Be able to implement an algorithm from a high-level description and verify its runtime properties.

• Be able to recognize intractable problems and have an idea how to develop approximation algorithms.
“I’ll never use any of this 311 stuff, I’m only going to do X in my career”

• No matter what you think X is, you’re wrong

• What you will work on the most in your life, doesn’t even exist yet
  – I learned programming in 1980. Think about it.

• Your perspective comes from being an intern
  – They don’t give the important stuff to interns, duh
Don’t forget

• READ the syllabus
• READ the first chapter of Skiena
• SKIM the first chapter of Kleinberg
• SWITCH to Section D