CSCI-UA.0380-001
Programming Challenges

Sean McIntyre

Week 1: Introduction

Today's agenda

- Course administration
- Course introduction
- Class introductions
- Lecture
- Break (~3:00-3:15pm)
- Practice problems
- Discussion of problems

Course Administration

- Class website
  - http://cs.nyu.edu/courses/summer13/CSCI-UA.0380-001/
- Contact
  - Contact me by email sm4266@nyu.edu
  - My goal is to respond within a day
- Office hours
  - No office
  - Email me / ask me to set up office hours

Course Administration

- Grading
  - Attendance (10%)
  - Homework (40%)
  - Midterm presentation (20%)
  - Final presentation (30%)

Course Administration

- Attendance (10%)
  - Expectation: You will attend every class
  - Programming contests require a lot of practice
  - We will have time-based practices in class that will help you measure your own growth
  - Of course, you will be excused without penalty for good reasons

Course Administration

- Homework (40%)
  - Expectation: You will do the exercises and try the problems that are assigned for homework.
  - Programming contests require a lot of practice
  - Homework gives you an opportunity to try problems without a time limit
  - Exercises are to be done alone; problems can be done together

Course Administration

- Midterm (20%) and final (30%) presentations
  - Expectation: You understand the problem, you can convey the problem and solution to me and your classmates.
  - Coding is but one aspect of programming contests
  - Opportunity to work on communication skills
  - 10-15 minutes long with questions
  - Chalkboard, slides, or interpretive dance

Course Administration

- Timetable
  - Tuesday and Thursday 1:30-4:45pm
  - WWH 101 (here!)
  - Midterm: July 30 in class
  - Final: August 15 in class

Course Administration

- Textbook
  - Steven and Felix Halim's “Competitive Programming”
  - https://sites.google.com/site/stevenhalim/
  - I will be teaching from the **Second Edition**
    - It's possible you can follow along with the First Edition (free), but do so at your own risk
  - Well-respected people in the programming competition community

Course Introduction

How this course fits in

- Introduction to programming class
  - How to write programs
  - Basic software engineering and design
- Data structure class
  - Learn about sorting algorithms
  - Basic data structures and how they differ
  - Basic graph algorithms

How this course fits in

- Theory of algorithms class
  - An overview of a number of “classical” algorithms
  - Dive into the algorithms: how they work, and why they work
  - Proof of correctness
  - Some discussion on how to apply the algorithms

How this course fits in

- CSCI-UA.0380-001 (this class)
  - Combines a lot of these classes
  - Problem-based learning
  - No formal emphasis on proof of correctness
  - Uses narratives with either contrived or practical scenarios challenge the learner
  - Teamwork

What you will learn

- Algorithms that will be practiced
  - Dynamic Programming (DP)
    - state-space search, games
  - Data structures
    - binary indexed tree, suffix array, union-find
  - Computational geometry
    - convex hull

What you will learn

- Reading comprehension
- Comparative problem evaluation
- Parsing and formatting text
- Tricks to reduce code and bugs
  - bitmasks, traversing 2D spaces
- Generate test cases for your code
- Teamwork

Homework

- Solve n problems through the semester
- At least m problems must be solved from p categories
- $|\text{categories}| > p$

Programming Contests

- ACM International Collegiate Programming Contest (ICPC)
- TopCoder
  - Weekly online individual competitions
- Google Code Jam
- Internet Problem Solving Competition
  - Annual, fun, different style of problems
- IOI, USACO

ACM ICPC

- 3 people
- 1 computer
- 5 hours
- 10 problems

Why compete

- Makes you a better programmer and thinker in many situations
- Intangible skill that will set you apart in the workforce
- You're all invited to join NYU progteam
- It's fun :)

NYU competition history

- 2012 Greater New York Regional (GNYR)
  - 3rd, 10th, 11th
- 2011 GNYR
  - 8th, 11th, 20th, 21st
- 2004 GNYR
  - **1st!!!**

Join progteam

- Join the progteam mailing list!
  - http://www.cs.nyu.edu/mailman/listinfo/progteam
  - (link on course website)

Course instructor

- About me
  - Coach of the NYU programming team since 2011
  - ACM ICPC world finalist 2007 and 2008
  - Coach, problem setter, and volunteer for ~7 years

Introductions

1) Your name, school, year, Java/C++, code IDE
2) What programming contests have you participated in?
3) If you had to choose a problem, which would you choose?
    1) Implement and solve a rubiks cube
    2) Which player would win a checkers game?
    3) Fastest way to escape a maze?

Lecture

Competitive Programming

- Given *well-known* computer science problems, solve them *as fast as possible*
  - Find a solution that reduces down to a well-known problems, not research problems
  - Pass all the judge data correctly
  - Solution should run fast enough
  - Do not over-engineer the solution

First problem

- There are 2*N houses scattered on a 2D plane
- Find N pairs of houses such that the sum of the distances between the houses is minimized
- e.g.
  - Houses: (1, 1), (8, 6), (6, 8), (1, 3)
  - Sum of distances: 4.83

First problem

- **Beginner** style of solving this problem (0-3 practices)
  - Never seen this kind of problem
  - Takes awhile to comprehend problem statement
  - Starts coding without knowing a solution
  - Tries either a greedy solution (“pick the closest two”) or a complete search (backtracking)

First problem

- **Inexperienced** style (3-9 practices)
  - Recognizes the problem from a prior practice
  - Realizes that greedy and backtracking doesn't work
  - Thinks there is a DP solution
  - Gives up and moves on

First problem

- **Non-competitive programmer** style (10+ practices)
  - Realizes the solution is matching on a graph
  - Realizes the input size is small so can solve it with DP
  - Realizes that bitmasks help solve the problem
  - Makes a mistake in implementation, has to debug
  - Gets accepted answer after a couple of hours

First problem

- **Competitive programmer** style (20+ practices)
  - Realizes all of this and can solve the problem in 30 minutes or less without mistakes

How to be a competitive programmer

1) Type fast and correctly

- Know your IDE (in ICPC competition, Eclipse!)
- In competition, you may bring in a limited set of notes which contain code that you can type from the paper

How to be a competitive programmer

2) Quickly identify problem types

- Ad hoc
- Complete search
- Divide and conquer
- Greedy
- Dynamic programming
- Graph
- Mathematics
- String processing
- Computational geom.
- HARD

How to be a competitive programmer

- Moreover, identify whether or not you can solve the problem type
  - **Solved** before / can solve again quickly
  - **Solved** before / will take time to solve again
  - **Seen** before / will solve this if all easier ones are solved
  - Not sure

How to be a competitive programmer

What kind of problem is this?

Given an M*N integer matrix Q (1 <= M, N <= 50), check if there exists a sub-matrix of Q of size A*B (1 <= A <= M, 1 <= B <= N) where mean(Q) = 7?

How to be a competitive programmer

3) Algorithm analysis

- After discovering a solution, convince yourself that it runs in time
- Look at the constraints of the problem
- Worst-case analysis *before* starting to code
- Rule of thumb: 1 billion operations ~= 1 second

How to be a competitive programmer

4) Master a programming language

- After thinking of a solution, convey the solution in code as quickly as possible
- Use libraries, shortcuts, and write simple code
- Know the C++ STL or Java API without having to look at the reference
- Like being a painter, photographer, or musician

How to be a competitive programmer

- Java:
  - Scanner, BigInteger, String static functions, Collections, different data types
  - Integer.parseInt()
  - String.substring()
  - etc
- C++
  - next_permutation()

How to be a competitive programmer

5) Test your code. There are many ways to fail:

- Presentation Error (PE)
- Wrong Answer (WA)
- Time Limit Exceeded (TLE)
- Memory Limit Exceeded (MLE)
- Runtime Error (RTE)

How to be a competitive programmer

- Submit correctly
  - Competitions only care about correct code
  - Is it worth the 20 minute penalty to submit without test cases?
  - The best teams write test cases before submitting their solutions

How to be a competitive programmer

6) Practice

- Talking about programming contests only get you so far
  - UVa Online Judge
    - http://uva.onlinejudge.org
  - TopCoder
    - http://topcoder.com
  - Project Euler
    - http://projecteuler.net/

How to be a competitive programmer

7) Teamwork

- Knowing your teammates
  - Delegating problems to each other
- Sharing the computer time effectively
- Creating test cases for each other
- Being able to convey ideas
- Pair programming

Practicing

Problem recipe

- Problem narrative
  - Can be unnecessarily long or misleading
- Input and output description
  - Usually very precise
  - You may assume that all input will be formatted like this
- Sample input and output
  - One or more inputs and expected outputs

Steps to solving a problem

- Read the problem
- Decide whether or not you know how to solve it
- If you think you can solve it:
  - Parse the problem
  - Write the code
  - Check that the program works on the sample input/output
  - Submit!
- If you're not sure, move onto the next problem

First practice

For next class

- Readings:
  - Programming Challenges: Chapter 1
  - Discussion of 1.2.2, 1.2.3, and 1.2.4 next class
- First challenge:
  - Register at http://uva.onlinejudge.org
  - Email me your user name
  - Solve mandatory first problem listed on website