Course Introduction

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Welcome to CS 491 CAP!

Your Objectives:

- Describe the goals and prerequisites of this course.
- Describe the grading scheme.
- Be able to practice effectively.

Why take this course?

- Primary course goal: make you good at competitive programming!
- Why should you want to do that?
  - It's fun!
  - Opportunity to learn:
    - useful data structures, algorithms, and mathematical insights;
    - practical applications of data structures and algorithms;
    - how to code and debug effectively; and
    - how to work well on a team.
  - You'll do really well on job interviews!
- But I'm not as good as those others!

Am I ready for this?

Do I Need CS 225 or 374?

- No! They help, but in this course it's more important to know how to **use** the algorithms than to implement them.

Skills Needed

- Familiarity with C, C++, or Java (CS 125 / 128)
- Willing to learn basic data structures (CS 225).
- Comfortable with recursion and algorithmic explanations (CS 173).
- Most important: eagerness to learn and practice!!

Textbook *Competitive Programming 4* by Steven and Felix

Also *Guide to Competitive Programming* by Antti Laaksonen

Online Judges

- Real contest problems
- Immediate Feedback
- Can emulate contest environment
- List of online judges:
  - UVa Online Judge `https://uva.onlinejudge.org/`
  - Code Forces `https://codeforces.com/`
  - Open Kattis `https://open.kattis.com/`
  - Peking Online Judge `http://poj.org`
  - ACM ICPC Live Archive `https://icpcarchive.ecs.baylor.edu/`
  - Sphere Online Judge (SPOJ): `http://www.spoj.com/`
  - Saratov State Online Judge: `http://acm.sgu.ru/`
- **Get an account on each of these!**
- But... we will primarily use UVa, Code Forces, and Kattis. We will send you a link to collect your online judge IDs later.

Online Contests

- Occur 6–8 times per month.
- Code Forces
  `http://codeforces.com/`
- Top Coder Single Round Matches (SRMs).
  `https://www.topcoder.com/`

UIUC ICPC Team Meetings

- SIG ICPC Website: `http://icpc.cs.illinois.edu/ipl.html`
  - Contains announcements, practice summaries, and practice resources.
  - Currently not being maintained...
- **Tryouts**
  - Two of them!
  - Top 15 students will get to compete in the regional contest.
- Practice contests on subsequent Saturdays.
- Details on http://icpc.cs.illinois.edu/calendar.html

Class Organization and Assignments

- Each period will have the following workflow:

  Lecture Video or Reading About half of the periods have an introductory video; otherwise there will be a reading in the textbook.

  Sample Problem(s)
  - These will be posted to the web page and announced on campuswire.
  - The problem(s) should be solved (or at least attempted) before class.
    - Class will begin with a short discussion of the problems.
    - Then a new problem will be given in class.

  Problem Set You will also get a biweekly problem set.
  - Typical format: 10 problems, you must solve 6.
  - Problems should be submitted on corresponding online judge.

  Contests You should participate in some contests.

NB: Please do not copy-paste code from other sources. You are only hurting yourself if you do!

Grading

- Course is Pass/Fail: Passing is 70%.
- Attendance is highly encouraged, but not mandated due to COVID concerns.
  - Measured by submission of practice problems.
- Completion of problem sets is worth 100%.
  - Most problem sets will have 10 problems: you must complete 6 of them.
  - We will let you drop one problem set. But really, you should do them all.
- Ungraded activity: creating a team reference document and/or template code.
  - You can use this for the class and for e.g., Code Forces.

Extra Credit

There are opportunities for extra credit here too!

- Attending a tryout counts as one contest or problem set.
- You can get points by contributing new problems to our problem sets.

Approach to Solving ICPC Problems

1. **Read the problem statement carefully!**
   - Pay attention to the input/output format specification.
2. Abstract the problem.
3. Design an algorithm.
4. Implement and debug.
5. Submit.
6. AC!
   - (else `GO TO 4`... or maybe even 3)
7. **If you want to improve rapidly:**
   - Read the problem commentary afterwards.
   - After a contest, "upsolve" any problems you couldn't finish.

What to Expect

- You will get better over time if you keep at it.
- Your progress will **NOT** be linear though!
- Certain classes of problems will become easier more quickly.
- You will get better over time if you keep at it.
- It is possible to get to world finals level from zero in just a few years!