Course Introduction

Introduction to Competitive Programming

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Spring 2024

Welcome to CS 491 CAP!

Your Objectives:

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

Course Goals

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 what if I'm not as good as those others?

Am I ready for this?

Do I Need CS 225 or 374?

- We will assume familiarity with CS 225 concepts.
- CS 374 is optional, but will help.
- Skills Needed
  - Familiarity with C, C++, or Java (CS 125)
  - Willing to learn basic data structures (CS 225).
  - Comfortable with recursion and algorithmic explanations (CS 173).
  - Most important: eagerness to learn and practice!!
- Textbooks
  - Competitive Programming 4 by Steven and Felix
  - Optional: Guide to Competitive Programming by Antti Laaksonen

Lecture Format

- Each period will have the following workflow:
  - Lecture or Video
    - A short lecture or video or two will introduce the topic.
    - Usually in class. Your precedessors were lazy. :)
    - Meant as a big-picture introduction, not to teach the algorithm.
    - You get to teach yourself by studying the implementation!
  - Sample Problem(s)
    - One "easy" sample problem to do to get started.
    - One or more interesting problems during/after class.
    - Plenty of class time for coding and explanations.

Assignments

- Problem Sets You will also get a biweekly problem set.
  - Typical format: 10 problems.
  - Submit all problems to the corresponding online judge.
- Contests You can also participate in some contests.
  - A 5 hour contest will replace one problem set.
  - Let the instructor know if you do this.

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% for both problems and attendance.

- Attendance:
  - We have an attendance app that we will use.
  - You must attend at least 70% of the available lectures.
- Completion of problems:
  - Complete 70% of all problems assigned.
  - Of course, we can't know in advance how many total problems there will be!

Extra Credit

There are opportunities for extra credit here too!

- Attending a tryout counts. You get two problems credit for each problem you solve.

Online Judges

- Code Forces `https://codeforces.com/`
- Real contest problems
- See syllabus for class link.

Other Judges

It's worth getting accounts here too.

- UVa Online Judge `https://uva.onlinejudge.org/`
- 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/`

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.
- Meetings: Tuesdays from 18:00–20:00
- Tryouts
  - Two of them! (One this Saturday)
- Join the IPL Campuswire group (Use code 4080).

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.

Example Problem

- POJ 1000: A + B Problem
  - Input: two space separated integers, a and b.
  - Constraints: $0 \leq a, b \leq 10$.
  - Output: $a + b$

C Code for POJ 1000

```c
#include <stdio.h>

int main() {
  int a, b;

  scanf("%d %d", &a, &b);
  printf("%d\n", a + b);
  return 0;
}
```

C++ Code for POJ 1000

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
  int a, b;

  cin >> a >> b;
  cout << a+b << endl;
}
```

Java Code for POJ 1000

```
import java.io.*;
import java.util.*;

public class Main {
  public static void main(String args[])
  throws Exception{
    Scanner cin=new Scanner(System.in);
    int a=cin.nextInt(), b=cin.nextInt();
    System.out.println(a+b);
  }
}
```

Example Problem

- POJ 1004 — Financial Management
  - Input: 12 floating-point numbers, each on a separate line
  - Output: Average of the numbers, rounded to two decimal places
  - Note that the answer must be preceeded by a dollar sign $!

C Code for POJ 1004

```c
#include<stdio.h>

int main() {
  double sum = 0, buf;
  for(int i = 0; i < 12; i++) {
    scanf("%f", &buf);
    sum += buf;
  }
  printf("$%.2f\n", sum / 12.0);
  return 0;
}
```

C Code for POJ 1004

```
#include <bits/stdc++.h>

using namespace std;

int main() {
  double sum = 0, buf;
  for(int i = 0; i < 12; i++) {
    cin >> buf;
    sum += buf;
  }
  printf("$%.2f\n", sum / 12.0);
  return 0;
}
```

Java Code for POJ 1004

```
import java.util.*;

class Main {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    double d = 0;
    for (int i = 0; i < 12; ++i) {
      d += in.nextDouble();
    }
    System.out.printf("$%.2f\n", d/12.0);
  }
}
```

Questions?