Introduction

Jaehyun Park

CS 97SI
Stanford University

June 29, 2015

Welcome to CS 97SI

- Introduction
- Programming Contests
- How to Practice
- Problem Solving Examples
- Grading Policy

Coaches

- Officially: Jerry Cain
- Actually: Jaehyun Park

Why Do Programming Contests?

- You can learn:
  - Many useful algorithms, mathematical insights
  - How to code/debug quickly and accurately
  - How to work in a team
- Then you can rock in classes, job interviews, etc.
- It's also fun!

Prerequisites

- CS 106 level programming experience
  - You'll be coding in either C/C++ or Java
- Good mathematical insight
- Most importantly, eagerness to learn

Topics

1. Introduction
2. Mathematics
3. Data structures
4. Dynamic programming (DP)
5. Combinatorial games
6. Graph algorithms
7. Shortest distance problems
8. Network flow
9. Geometric algorithms
10. String algorithms

Programming Contests

- Stanford Local Programming Contest
- ACM-ICPC
  - Pacific Northwest Regional
  - World Finals
- Online Contests
  - TopCoder, Codeforces
  - Google Code Jam
- And many more...

How to Practice

- USACO Training Program
- Online Judges
- Weekly Practice Contests

USACO Training Program

- `http://ace.delos.com/usacogate`
- Detailed explanation on basic algorithms, problem solving strategies
- Good problems
- Automated judge system

Online Judges

- Websites with automated judges
  - Real contest problems
  - Immediate feedback
- A few good OJs:
  - Codeforces
  - TopCoder
  - Peking OJ
  - Sphere OJ
  - UVa OJ

Weekly Practice Contests

- Every Saturday 11am-4pm at Gates B08
  - Free food!
- Open to anyone interested
- Real contest problems from many sources
- Subscribe to the `stanford-acm-icpc` email list to get announcements

Example

1. Read the problem statement
   - Check the input/output specification!
2. Make the problem abstract
3. Design an algorithm
   - Often the hardest step
4. Implement and debug
5. Submit
6. AC!
   - If not, go back to 4

Problem Solving Example

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

POJ 1000 Code in C/C++

```c
#include<stdio.h>
int main()
{
    int a, b;
    scanf("%d%d", &a, &b);
    printf("%d\n", a + b);
    return 0;
}
```

Another Example

- POJ 1004: Financial Management
  - Input: 12 floating point numbers on separate lines
  - Output: Average of the given numbers
- Just a few more bytes than POJ 1000...

POJ 1004 Code in C/C++

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

Something to think about

- What if the given numbers are HUGE?
- Not all the input constraints are explicit
  - Hidden constraints are generally “reasonable”
- Always think about the worst case scenario, edge cases, etc.

Grading Policy

- You can either:
  - Solve a given number of POJ problems on the course webpage
  - OR, participate in 5 or more weekly practice contests
- If you have little experience, solving POJ problems is recommended
  - Of course, doing both of them is better

Stanford ACM Team Notebook

- `http://stanford.edu/~liszt90/acm/notebook.html`
- Implementations of many algorithms we'll learn
- Policy on notebook usage:
  - Don't copy-paste anything from the notebook!
  - At least type everything yourself
  - Let me know of any error or suggestion

Links

- Course website: `http://cs97si.stanford.edu`
- Stanford ACM Team Notebook: `http://stanford.edu/~liszt90/acm/notebook.html`
- Peking Online Judge: `http://poj.org`
- USACO Training Gate: `http://ace.delos.com/usacogate`
- Online discussion board: `http://piazza.com/class#winter2012/cs97si/`