Competitive Programming

CS 104C
Introductions

I am Prof. Etienne Vouga
• evouga@cs.utexas.edu
• GDC 5.508

Coach of the ICPC teams

Office hours by appointment
Fellow Lecturers

Arnav Sastry and Ethan Arnold
Other Affiliated Faculty

Glenn Downing

Shyamal Mitra
What is Competitive Programming?
Example Problem

Input: An integer $N$
• $0 \leq N \leq 10^6$

Output: The number of zeroes at the end of $N!$

Time limit: 1 sec
Memory limit: 10 MB
What is Competitive Programming?

Given a concrete problem statement:
1. Analyzing the problem and determining what algorithm can solve it;
2. Identifying the possible pitfalls and corner cases;
3. Quickly producing an implementation that “Just Works the first time”

In other words, efficiently analyzing and solving low-level programming problems
<table>
<thead>
<tr>
<th></th>
<th>High-Level Design</th>
<th>Low-Level Design</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit of Concern</strong></td>
<td>Application; Library</td>
<td>Function</td>
</tr>
<tr>
<td><strong>Main Goals</strong></td>
<td>User Experience</td>
<td>Correctness</td>
</tr>
<tr>
<td></td>
<td>Maintainability</td>
<td>Performance</td>
</tr>
<tr>
<td><strong>Key Questions</strong></td>
<td>What libraries can I use?</td>
<td>What algorithm can I use?</td>
</tr>
<tr>
<td></td>
<td>Will these features be useful?</td>
<td>What are the corner cases?</td>
</tr>
<tr>
<td></td>
<td>Will this design scale to more users/data?</td>
<td>Will this run in reasonable time and memory?</td>
</tr>
<tr>
<td><strong>Skills Needed</strong></td>
<td>Planning</td>
<td>Problem-solving</td>
</tr>
<tr>
<td></td>
<td>Communication</td>
<td>Algorithms knowledge</td>
</tr>
<tr>
<td><strong>Scope of Effort</strong></td>
<td>Teams working for months</td>
<td>One programmer working for hours</td>
</tr>
</tbody>
</table>
Benefits of Competitive Programming

- Improved problem solving skills
- Improved knowledge of algorithms
- Write more optimized code
- Write less buggy code
- Great practice for technical interviews
- Have fun, win prizes and glory
Mechanics of a Competition

You receive several (3-10) problems of varying difficulty

Write solution, submit it to an automated judge that runs test suite
Mechanics of a Competition

You receive several (3-10) problems of varying difficulty
Write solution, submit it to an **automated judge** that runs test suite
Get back ~2 bits of information:
Accepted, Wrong Answer, (Compile Error), (Time Exceeded), (Memory Exceeded)
You are **not** told the failure cases!
Two Types: Online and Offline

**Online**: (TopCoder, CodeForces, Google CodeJam)
- Individual
- Can use Internet, your old code, …

**Offline**: (ICPC)
- Team (shares one computer)
- No Internet
Competitions vs Industry

In competitions:

- Severe time pressure
- no time for unit tests
- no time to write documentation
- Style and code quality doesn’t matter*
- Maintainability doesn’t matter
- Problems are self-contained
Example Problem II

Input: A sorted list of $N$ integers $x_i$ and an integer $y$
- $1 \leq N \leq 10^6$
- $-2^{30} \leq y, x_i \leq 2^{30}$

Output: The number of elements in the list strictly less than $y$
Time limit: 1 sec
Memory limit: 10 MB
# Efficiency Rule of Thumb

<table>
<thead>
<tr>
<th>complexity</th>
<th>maximum N</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N)</td>
<td>1000000</td>
</tr>
<tr>
<td>(N log N)</td>
<td>500000</td>
</tr>
<tr>
<td>(N^2)</td>
<td>5000</td>
</tr>
<tr>
<td>(N^3)</td>
<td>400</td>
</tr>
<tr>
<td>(N^4)</td>
<td>80</td>
</tr>
<tr>
<td>(2^N)</td>
<td>20</td>
</tr>
<tr>
<td>(N!)</td>
<td>10</td>
</tr>
</tbody>
</table>
Lessons?

Read the problem statement

• Look at the limits
• Check for corner cases

Almost always a performance vs complexity tradeoff – choose carefully!

Big-O critical (but don’t sweat log N)
Other Tips for Getting Started

Become proficient in **one** language, and know its libraries and I/O functions cold

- C++, Java, Python

Get out there and code

- try out online contests: codeforces, codechef, projecteuler, hackerrank
- you greatly improve by practicing
Assignment and Grading

500 points total

400 points (80%) homework exercises (3/week):

• **Vanilla Problem**: tests key concepts

• **Codeforces Exercises**
  • online judge
  • we pick three problems, you choose one
  • can do more for extra credit (5 pts per problem)

• **Find the Bug**
  • our solution has an issue;
    find the problem
Assignment and Grading

400 pts homework exercises (3/week)

100 pts programming contest participation
  • every two weeks on Friday night
  • first contest: September 8th

One contest (your choice) is required
Can do more for extra credit
  (10 per problem per contest)
Prerequisites

Data Structures (or equivalent)

Working knowledge of Java (for Find the Bug problems)

Strong working knowledge of one of:

• C/C++, Java, Python
Academic Dishonesty

Do look up:
• Language reference
• High level algorithmic concepts

Don’t look up:
• Problem solutions / editorials

• We’ve caught people in the past.
Tentative List of Topics

- state space search / graph algorithms
- recursion / backtracking
- binary search
- greedy algorithms
- dynamic programming
- advanced graph algorithms
- number theory
- probability/combinatorics
- advanced data structures
Coming Soon on Canvas

Links to online competitions
Instructions for setting up Codeforces account (needed for assignments)
First problem (due Wed. at midnight)