Introduction to Competitive Programming

MHS Competitive Programming Leadership

September 17, 2024

Welcome to the McLean High School Competitive Programming Club! This year we will be using handouts during meetings.

To clear any misconceptions: from now on we will often abbreviate "Competitive Programming" using "CP".

1 Breaking down a CP problem

1.1 Components of a CP problem

CP problems have a few standard components:

- **Problem Statement**: The problem statement explains a computational task to the competitor. The competitor then writes a program to take in the input, perform the computational task, and return the proper output.
- **Input Format**: When your program runs, it will receive input from the standard input (as if a user is typing it in). The text format of this input will be described here.
- **Output Format**: Your program should print the output for the problem in a specific text format, which will be described here.
- **Sample Cases**: These are examples of inputs and their proper outputs to better explain the computational task.
- **Constraints**: These are the limits on the input size, which will dictate how efficient your program must be.
- **Computational Constraints**: These are the limits on the time and memory your program can use. If your program exceeds these limits, it will be rejected.

1.2 Notation

CP problem statements often borrow basic mathematical notations. Here are some fundamental ones:

- **Any letter**: Describes a variable. We deal with numerical variables, e.g. "let x be an integer...", and array variables, e.x. "let a be an array of integers...".
- **Subscript under a letter**: Refers to an element of an array. For example, a_3 refers to the 3rd element of the array a. The subscript can also be a variable itself, for example a_i refers to the ith element of the array a.
- **Σ**: The summation symbol. For example, $\Sigma_{i=1}^{n} a_i$ is the sum of all elements in an array a which has length n.
- **Inequality operators**: $<, >, \leq, \geq$ are used to compare two numbers. You will commonly see inequalities such as: $1 \leq n \leq 10^5$. This means that the value of n is constrained to be between 1 and 10^5. Note the use of exponents to denote large constraints.

2 Contests

2.1 Contest Structure

CP competitions are hosted on online judge platforms. These platforms are responsible for serving the problem statement and grading submissions. In a typical CP contest, you will have a set of problems to work through under a time limit, usually a few hours. Oftentimes the problems are sorted by difficulty (as will be the case in our practice contests), but note that this isn't guaranteed.

2.2 Code Submission

After you finish your code for the problem, you will submit it to the online judge. The judge will then run your code on many test cases, then issue a single verdict:

- **Accepted**: Your code passed all the test cases, it is deemed correct.
- **Wrong Answer**: Your code printed the wrong output for at least one test case.
- **Time Limit Exceeded**: Your code took too long to run- you must think of a more efficient solution.
- **Memory Limit Exceeded**: Your code used too much memory- you must think of a more efficient solution.
- **Runtime Error**: Your code errored and crashed during at least one test case.
- **Compile Error**: For compiled languages like Java and C++. It means your code failed to compile.

The idea of one verdict per problem is the case with most judges. However, some judges (most notably USACO) may grant verdicts on a per-testcase basis, so your code could be "Accepted" on some test cases and "Wrong Answer" on others. This allows partial credit to be given.

2.3 Strategy

Your contest strategy will vary depending on the scoring system. We will explain two scoring systems: ICPC-style and IOI-style.

In ICPC-style contests contestants are ranked first by their number of problem solved, then by total time taken to solve them. A penalty of 20 minutes is added per failed submission. Notice how all problems are weighted equally. This means that it's a better strategy to snag multiple easy problems rather than spend a lot of time on a hard problem.

In IOI-style contests there are usually 3 problems, each weighted equally. Partial credit is allowed. In USACO for example, each problem is worth 333.33 points for a total of 1000 points after summing and rounding up. Partial credit means that, for instance, solving 7/10 testcases of a problem would grant you $333.33 \times (7/10) = 233.33100$ points. In USACO promotions are usually awarded for scores above 750, so two fullsolves (solving all testcases of a problem) and one partial solve is usually enough. This means that it is better to submit a "dumb" solution to one problem to get a partial solve if it means saving enough time to fullsolve the other problems.

3 Practicing

3.1 Practice Strategy

It is best to practice on a site where problem difficulties are rated, such as `codeforces.com`. Then, start from the easiest difficulty and move up until you face problems that are just within/without reach. Don't spend too long on a problem. If around 30 minutes to an hour of thinking doesn't suffice, it's ok to just read the solution- you will learn more from doing so than pouring more time into the problem.

3.2 Contests

We will hold practice contests each meeting, but there are many online contests that are regularly held:

- **Codeforces**: Regular contests, varied schedule, usually 2-3 hours long.
- **AtCoder**: Regular contests during Saturday/Sunday mornings, usually 2 hours long.
- **LeetCode**: Regular contests during Saturday nights/mornings, usually 2 hours long.

It's important to do practice contests, you will need to get used to the contest environment and time pressure!

3.3 Programming Languages

We can give advice for which programming language to choose, but the choice is up to you.

C++ is the most commonly used language by far. It is the fastest langauge by execution time. It is also the hardest to learn by a noticeable (but not huge) margin.

Java is very another popular language. Many students already know it from AP CS classes. If this is you, then continuing with Java is a good choice.

Python is perhaps the most controversial langauge. It is the most popular language (in terms of general programming), but it's execution time is very slow. In harder problems, the optimal solution written in Python may still not pass the time limit. If you only know Python, we suggest sticking to it for now, but learning C++ or Java in the future.