Introduction to Contest Programming

Srinidhi Krishnamurthy and Mihir Patel

September 17, 2017

1 Introduction

Welcome to competitive programming! This lecture is going to detail some of the basic ideas in contest programming and is going to introduce complete beginners to the subject. We go over the various competitions that Senior Computer Team does and detail computational complexity. Typically, contest programs reward people for coming up with fast, space-efficient algorithms quickly and accurately.

2 USACO

USACO (the USA Computing Olympiad), is the contest that we focus on the most in Senior Computer Team. It is an individual contest and lasts for typically four hours on select weekends. You can log in at anytime on this select weekend and take the contest for your division. The competitions have a max score of 1000 points.

2.1 The Division System

USACO runs on the Division System where contestants are broken up into four division: Bronze, Silver, Gold, and Platinum. Every competitor initially starts out at Bronze and then moves up through the ranks determined by a cutoff. If a contestant is able to score a perfect 1000 during the contest, they are given the opportunity to "in-contest promote" and jump straight to the next division within that weekend. Because of this, we have seen classmates go from Bronze to Platinum in one contest (though it is quite rare). Here are the official descriptions for each of the divisions:

- *Bronze* - Bronze-level problems are designed to be accessible and engaging for students who have learned how to program but who do not yet have any formal algorithmic training, although they are still designed to require creativity and "algorithmic thinking". Compared to the "old" bronze division, problems in the new bronze division are planned to be much more accessible to novice students.
- *Silver* - Students in the silver division begin to encounter fundamental algorithms, such as recursive searching, "sort and scan" type algorithms, basic data structures, simple greedy algorithms, and others. Compared to the "old" silver division, problems in the new silver division are planned to be roughly between the old bronze and silver divisions in difficulty.
- *Gold* - The gold division requires knowledge of more advanced algorithmic concepts, such as dynamic programming, various graph algorithms, and more advanced data structures. Compared to the "old" gold division, problems in the new gold division are planned to be roughly between the old silver and gold divisions in terms of difficulty.
- *Platinum* - Platinum problems are quite challenging, requiring clever combinations of sophisticated algorithmic ideas. These problems are intended to challenge even the very top competitors.

USACO uses files for input and output (which is not really the standard). Most other competitions and platforms (such as CodeForces) use standard input and output (reading straight from *System.in* in Java or with the *cin* and *cout* objects in C++'s *iostream*). In Java, you can use the *Scanner* and *File* objects for file input and output and $PrintWriter$ for file output. You can find these in *java.util.*$*$. Furthermore, if you want to get an extra speed boost in your programs, you can try using $BufferedReader$ or $BufferedWriter$. In C++, you can use the $ofstream$ and $ifstream$ classes in $fstream$.

3 Computational Complexity

The *computational complexity* of an algorithm is a measure of its efficiency. It is the amount of time (*time complexity*) or space (*space complexity*) an algorithm takes to run as a function of the size of the input. More specifically, we use big-O notation to formalize this. We typically only consider the highest order/most dominant function within the computational complexity. In programming contests, we want a program to have a good enough complexity so that it can handle all of the specified input sizes. If a program gets accepted, it typically means that it has the intended computational complexity. If a program gets the "time limit exceeded" error, it means that the code did not run in the time allotted. As a good rule of thumb, most modern computers and processors can do 10^8 operations per second. If you input the maximum input size into your computational complexity function, it should not exceed 10^8.

4 Common Complexities

- $O(1)$ - constant
- $O(\log n)$ - logarithmic
- $O(n)$ - linear
- $O(n^2)$ - quadratic
- $O(n^3)$ - cubic
- $O(2^n)$ - exponential

5 Cheat Sheet

In USACO, for each test case, you are **typically** given 1 second for C++ and 2 seconds for Java. Your programs are run on machines that do approximately 10^8 operations per second. Based on the input size bounds given to you, here are around the complexities your programs should be:

- $N \leq 10 : O(N!)$
- $N \leq 25 : O(2^N)$
- $N \leq 50 : O(N^4)$
- $N \leq 500 : O(N^3)$
- $N \leq 5000 : O(N^2)$
- $N \leq 100000 : O(N \log N)$
- $N \leq 1000000 : O(N)$

6 Implementation Problems

Implementation Problems are those that really do not require much Computational Complexity analysis. They focus on being able to write code quickly and accurately. However,these problems are usually only found in easier contests, since they don't require too much thinking or creativity. In this case, the problem will typically layout what you need to do and you just have to carefully implement it. Most problems in competitive programming after the initial stage are asking you to come up with clever algorithms that are both fast and space efficient.

7 Some Inspiration

It is worth noting that there is way to systematically build up your experience. The only way to get better at these contests is to read up on topics and do a **LOT** of practice. Don't be discouraged if you don't do as well as you expect on your first contest. Greatness takes a little patience.