Competition Programming and Problem Solving

15-295 Spring 2020

In this course you will learn the techniques and skills needed to solve algorithmic programming contests problems such as those that appear on the ACM ICPC, Codeforces, and Topcoder. Most of your time will be spent writing programs on your own to solve problems.

Some students may go on to participate in the ACM ECNA regional event (which occurs in the fall), the ACM ICPC North America Championship (which occurs early in the spring), and possibly even the ACM ICPC World Finals (which occurs later in the spring).

But the skills you will pick up from the course are far more valuable than just enabling you to win contests. Many of the algorithms and techniques are classic ones that every computer scientist should know. You will also learn to think about algorithms in a deeper way, because many of the problems require you have to devise a new algorithm, not just apply a classic one. These skills will be of great value in your other classes, in your job interviews, and in your future work.

Basic Information

~~We have optional lectures on Monday evening at 6:00pm, and regular weekly meetings on Wednesdays at 6:30pm. The lectures will cover introductory content aimed at beginners of competitive programming. Although attendance to the lectures is optional, it is strongly encouraged for beginners.~~

Due to the cancellation of all in-person teaching, we will be cancelling lectures until further notice. We will not host remote lectures, but instead, we will post a brief tutorial document for each week's topic that you should read to familiarize yourself with the basic ideas. Competing in the weekly programming contest is now to be done remotely.

> Lectures (**Cancelled until futher notice**): Mondays from 6:00pm - 7:00pm, NSH 4305

Class Meetings (**Remote until futher notice**): Wednesdays from 6:30-9:00pm, GHC 4307
Instructor: Danny Sleator <sleator@cs.cmu.edu>, Office: Gates 7205, Phone: 412-268-7563
Teaching Assistant: Daniel Anderson <dlanders@cs.cmu.edu>, Office: Gates 7001
Administrative Assistant: Tony Mareino <amareino@andrew.cmu.edu>, Office Gates 7227
Google 15-295 Group: http://groups.google.com/group/15-295 (Required)
Codeforces 15-295 Group: http://codeforces.com/group/KIrM1Owd8u/contests (Required)
Name List: Put your names here (Required)
Grade Spreadsheet: Tabulation of Scores

For more information on how to join these groups, etc, see the Logistics section below.

Weekly Problems

https://codeforces.com/group/KIrM1Owd8u/contest/278255

Week #15 - The Finale Finale!

Contest Link: Here
Problem Set: Here

Week #14 - The Finale

There is no topic theme this week. Instead, you will solve problems from all of the topics that we have covered this semester!

Contest Link: Here
Problem Set: Here
Solutions: Here

Week #13 - Number Theory

Our last topic of the semester is number theory. We will focus a lot on prime numbers, and factoring, which show up in competitions fairly often. These kinds of techniques are often combined with counting problems, since sometimes you need to use clever number theory to come up with ways to count certain objects efficiently and without double counting.

Tutorial: Number Theory tutorial
Practice Problems: Link
Contest Link: Here

Problem Set: Here
Solutions: Here

Week #12 - Geometry

We are on our second-last topic of the course, and its one that everybody either loves or hates... geometry! Geometry problems are often some of the most difficult problems that appear in programming competitions because they can be full of corner cases and rounding issues. This week, we will practice the fundamental techniques behind solving geometry problems.

Tutorial: Geometry tutorial
Pratice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #11 - Advanced Data Structures

This week we will cover some advanced data structures that you won't find in your language's standard library. In particular, we will learn about data structures for two very fundamental problems: Lowest common ancestors in trees, and ranged minimum queries over arrays. It turns out that these two very different-sounding problems can actually be reduced to one another and hence we can use the same data structures to solve them.

Tutorial: LCA and RMQ tutorial
Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #10 - Strings

This week we will practice string problems. In particular, we will learn about using hashing to solve string problems, such as pattern matching.

Tutorial: Rabin-Karp and Pattern Matching
Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #9 -- Combinatorial Enumeration (aka Counting Stuff)

This week we will learn how to count things!

Tutorial: Counting tutorial
Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #8 -- Applications of Maximum Flow

This week, we will explore even more graph algorithms. In particular, we will be looking at applications of maximum network flow. One key difference compared to prior weeks is that we will not actually focus on the algorithms for maximum flow (you should learn these in your algorithms courses), but rather on the applications. It is therefore crucial that you have a working implementation of a maximum flow algorithm ready to go before this week's contest begins. A maximum flow code is an important component of any competitive programmer's code library. If you have not learned a maximum flow algorithm before and can not implement one yourself, you are allowed to use a prewritten implementation by someone else. I will provide you with a sample C++ implementation that you are welcome to use. The first of the practice problems this week will help you check your implementation, as it simply asks you to compute a maximum flow in a graph, and not solve any particular application. It is strongly recommended that you do this to avoid getting wrong answers during the contest due to a bug in your flow code.

Sample code: C++
Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #7 -- Mixed Contest

This week, instead of focusing on a particular algorithmic topic, we will have a contest that includes problems from the various topics that we have covered so far. Specifically, expect to find problems on data structures, binary search, dynamic programming, graphs, and shortest paths.

Contest Link: Here
Problem Set: Here
Solutions: Here

Week #6 -- Shortest Paths in Graphs

This week, we will continue the theme of graph algorithms and talk about shortest path problems in graphs. The fundamental algorithms that you will need to learn to solve these kinds of problems are breadth-first search and Dijkstra's algorithm. Other algorithms, such as Bellman-Ford and Floyd-Warshall may also come in handy, but are far less common.

Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #5: Graph Traversal

This week we will talk about graph algorithms. This is a topic that will probably span multiple weeks as we cover things like shortest paths, minimum spanning trees, and network flows. For this week, we will focus on the fundamentals, and cover things like how to represent graphs and how to implement graph traversal algorithms.

Practice Problems: Link
Contest Link Here
Problem Set: Here
Solutions: Here

Week #4: Dynamic programming

This week, we will practice the use of dynamic programming (DP) as a technique to solve programming competition problems. DP is quite possibly the most frequently occuring algorithmic technique used in competitions, with every ICPC contest always featuring at least one, if not several problems that require it. Masting this technique is key to becoming a strong competitive programmer. DP revolves around two key concepts, *optimal substructure*, which means that a problem can be solved by breaking it into smaller versions of itself (much in the same way as divide and conquer), and *memoization of overlapping subproblems*, which means to cache the solutions to the smaller problems in case they need to be solved multiple times. Of all topics in competitive programming, it probably requires the most practice in order to master, so get started!

Practice Problems: Link
Contest Link: Here
Problem Set: Here
Resources: 451 Notes

Week #3: Binary Search

This week, we will solve problems that use the *binary search* technique. You might think of binary search as an algorithm for finding an element in a sorted list, but it is actually a much more general technique than that, and can be used to solve a wide range of problems. Essentially, binary search allows you to solve problems where you must minimize or maximize some quantity subject to some contraint, as long as that constraint is monotone. A constraint is monotone if it is true/false for all values of x up to some value, and then becomes false/true after that value. Binary search allows us to efficiently

find this partition point, and hence solve the corresponding optimization problem. The practice problems this week should help you to learn to apply this powerfull technique.

Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #2: Fundamental Data Structures

This week, we will focus on the most commonly used and fundamental data structures in competitive programming. Specifically, we will review and practice using arrays, binary search trees, hashtables, and priority queues. Knowing and understanding how to use these data structures in your favourite programming language is crucial to being able to quickly implement solutions to problems. We will mainly use C++ to demonstrate, but the ideas should be applicable to any language that supports these data structures.

The three practice problems should give you some practice using binary-search-tree-based data structures (set and map in C++, or TreeSet and TreeMap in Java) and priority queues (priority_queue in C++, or PriorityQueue in Java). As always, remember to register for the practice contest via this page.

Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Week #1: Introduction

This week has no particular algorithmic theme. Instead, we will go over the basics of competitive programming and do our first contest on Codeforces. Please make sure you have a Codeforces account ready to go and understand how to upload your solutions to the problems. You should do some practice problems to get the hang of writing and submitting solutions to the Codeforces system.

We have created a practice contest to do just that. So go to this page and (1) join the group (right side) (2) register for the contest and (3) enter the contest. Then you will find three problems you can try to solve. The are: *Lights in the Morning*, *Deconstructed Password* and *Fine-tuned Resistance*.

Practice Problems: Link
Contest Link: Here
Problem Set: Here
Solutions: Here

Grading

This course is 5 units. Each week you will be given several problems to try to solve during class. You will be allowed (for half credit) to solve these problems during the week after the contest ends. You can also get credit for solving problems during rated contests on Codeforces. (This site run rated contests approximately every two weeks.)

To be more specific, you can earn points from the following sources:

- Solving problems during the in-class contests. Each problem you solve is worth one point
- Solving problems from the in-class contests during the week following the contest. Each problem solved in this way is worth 0.5 points
- Solving problems during rated competitions on Codeforces. Division 3 contests and Problem A from Division 2 contests are not counted. You will earn one point for each problem solved.

Here is how your grade is determined:

score ≥ 25: A
score ≥ 15: B
score ≥ 10: C
score ≥ 5: D

Rules

You can make use of generic on-line resources while solving problems. These include things like language documentation, API documentation, algorithm descriptions, terminology, etc. You should not search for or make use of code written by others to solve the specific assigned problem.

If you're stuck on a problem, you are welcome to discuss it with another student in the class, or the course staff. Keep the level down so as not to disturb those around you.

Logistics

- Everyone in 15-295 and/or on any programming team should be in http://groups.google.com/group/15-295. This is how we communicate with the class.
- Everybody in the class is required to have a Codeforces account. We will be running our weekly contests on the Codeforces website via a mechanism called "mashups". If you do not already have an account on Codeforces, please create one. And add your name along with your Codeforces handle name to this spreadsheet. (It's the same one in Basic Information list above.)
- To join the 15-295 codeforces group, first create your codeforces account. Then go to this 15-295 Codeforces Group page. On this page you should see a list of the contests created for this course. After you've logged in with your codeforces account and clicked

the link above to get to the group page, toward the right side of the page click on the link to join the group. (You only have to join once.) Then a "Register" link should appear next to the contest. Click on this link to register for the contest. After that another "Enter" link should be available for you to join the contest (if it has already started).

- As we proceed through the semester we will tabulate your results and put them into the grades document mentioned above.
- If you have a laptop bring it to class. If you don't you will have to work in one of the clusters.

Training Resources

There are many online resources available for you to train with if you intend to become a serious competitive programmer. You can find thousands of practice problems for you practice and improve your skills. Some good places to find practice problems include:

- Codeforces: We use Codeforces for all of our in-class contests. On the Codeforces website, you can choose to solve individual problems from their old contests at your leisure, do a "virtual contest", in which you will be given a time limit to solve a set of problems from an old contest, or you can participate in real live contests roughly once per week.
- Topcoder: Topcoder also hosts live contests and allows you to practice on problems from their old contests. You need to install their Java applet in order to browse and solve problems, however.
- UVA Online Judge: UVA has a huge database of problems, many from past ICPC contests. A good website for finding particular problems to focus on is uHunt, which catalogues problems from UVA Online Judge into convenient categories for you to focus on (e.g. data structures problems, graph problems, string problems, etc.)
- SPOJ: SPOJ has a big database of practice problems that you can use. You can search for problems by tags to find ones that focus on a specific topic.
- CodeChef: Codechef has a number of practice problems that are categorized by difficulty, so you can work your way up from easy problems to hard problems as you progress. You can also search by tags to find problems that focus on specific topics.

Learning Material

If you are a beginner looking for resources to learn the various topics that appear in typical contests, some good sources are:

- E-Maxx Algorithms in English: This site contains well-written descriptions of hundreds of important algorithms, along with details on how to code them.
- Topcoder Competitive Programming Tutorials: A good selection of tutorials on standard problems appearing in contests.

- USACO Training Site: A training platform for the USA Computing Olympiad. It has a series of lessons with corresponding problems that are accessible to beginners and will make you a strong contest programmer.
- Competitive Programming by Halim & Halim: A popular book that covers the standard topics in competitive programming and includes a thorough catalogue of practice problems from UVA Online Judge (the uHunt website is based on this book). An updated version is coming out soon.
- Competitive Programmer's Handbook by Laaksonen: An ebook that covers most of the standard topics in competitive programming

---

Danny Sleator
Last modified: Wed Apr 29 18:29:24 2020