Competitive Programming: The Initial Hurdle

From a Beginner to a Division 1 Competitive Programmer

Prakhar Gupta

August 20, 2021

Contents

1. Introduction

Wikipedia defines Competitive Programming, or CP, as a mind sport usually held over the Internet, involving participants trying to program according to provided specifications. A programming competition generally involves the host presenting a set of logical or mathematical problems to the contestants, and contestants are required to write computer programs capable of solving each problem. Judging is based mostly upon the number of problems solved and time spent for writing successful solutions, but may also include other factors (quality of output produced, execution time, program size, etc.)

In this book, I will not teach you any new algorithms or data structures, but rather tell you exactly what, how, and from where to learn and practice Competitive Programming. This way, this handbook will be a valuable guide to any beginner or intermediate level Competitive Programmer, seeking to improve and become a Division 1 programmer.

2. Benefits of Competitive Programming

Before we begin, let us look at a few reasons for starting with competitive programming.

- **Coding Interviews:** The problems asked in the coding rounds of most interviews to prestigious companies like Facebook, Apple, Amazon, Netflix, and Google (FAANG) are just easy to medium competitive programming problems. If you are an experienced competitive programmer, the coding round of every interview will just be like another CP contest for you.

- **International Olympiads and Competitions:** If you are a high school or a college student, you can target international computing Olympiads like the International Olympiad in Informatics (IOI) for high school students and the International Collegiate Programming Contest (ICPC). Apart from these, several other prestigious contests with onsite finals and large cash awards are also organized by major companies. These include Google Code Jam and Facebook Hacker Cup by Google and Facebook respectively.

- **Improving Problem-solving skills:** Practicing Competitive Programming significantly improves your problem-solving skills in general and leads to the overall development of your brain and thinking capacity.

- **An interesting hobby:** A majority of competitive programmers do not practice competitive programming to train for Olympiads or to seek jobs but just as a challenging and mind-capturing hobby. You really do not need a reason to practice CP!

3. How to begin with Competitive Programming?

The only prerequisite for starting with competitive programming is to know a programming language.

In case you are already learning or using a programming language in your school or college, do not hesitate to go ahead with that as there are no restrictions on the language you use to write your code in. CP contests usually offer participants a wide range of languages to choose from.

However, if you do not know any programming language, I would recommend you to learn any among C++, Java, and Python. These are the three most widely used languages by competitive programmers today. They are usually preferred because their standard libraries contain a large collection of data structures and algorithms.

There are several good videos and resources on the internet where you can learn a programming language from. YouTube, in particular, has many such integrated tutorials.

An important point to be noted here is that you needn't devote a lot of your time to learning everything about a language or trying to master every single one of its libraries. You can start with CP as soon as you are familiar with the basics like loops, conditional statements, and basic data structures, and are comfortable in implementing simple programs in that language.

A common myth among absolute beginners is that one needs to be good at math to do well in competitive programming. Although being good at math can be advantageous at times, as long as you are clear with the basic concepts of math taught at grade 10 level, you are good to go.

Once you are ready with your language, you can safely begin with practicing from and participating in contests hosted by three of the most popular competitive programming websites, namely, CodeChef, AtCoder, and Codeforces. Each of these will allot you a rating that will reflect your standing and performance in their regular rated contests.

4. Common Mistakes

- **Spending too much time on easy problems:** One of the most important rules about practicing CP is that you must always try to stay out of your comfort zone. Many beginners spend hours solving hundreds of easy problems while being under the false impression that they are practicing when in reality they are just wasting their time. Remember, the end result is not going to be the number of problems you have solved in a day or the number of hours you spent practicing, but rather how much you learned and improved that day. If at any time you feel confident solving the current level of problems, go ahead and slightly increase the difficulty. Make sure that you also do not increase the level by a huge amount. Otherwise, you will just get stuck and end up demotivating yourself. Knowing what to practice is thus an extremely important aspect of CP.

- **Reading too much:** Beginners in CP are usually under a false impression that they must first learn all the algorithms and data structures before starting. However, more important than learning new algorithms is to practice the ones you already know. Do not be like that person who knows everything about swimming but has never stepped inside a swimming pool! Knowledge of a topic is of little value unless you can code it during a live contest. For instance, binary search is one of the first algorithms one reads but still a lot of people fail to realize when a problem is based on binary search.

- **Fear of a drop in rating:** Rating is a big source of motivation for all of us but we must always remember is that rating is nothing but a reflection of our current level at competitive programming. A drop in rating after a bad contest must actually be taken as a motivation to work harder and perform better in the next one rather than a source of demotivation. Some, for instance, stop taking part in contests just because of the fear of losing their rating further. My advice to them is that the path to success is never straight and so is the rating curve to being a successful programmer. It is extremely important to keep participating in contests from time to time as

they help you keep track of your progress and ensure that you stay on the right track. In case you did have a significant drop in your rating after a seemingly bad contest, one possible point of motivation for you might be the fact that there is a higher probability of getting a positive delta in the next contest.

- **Getting stuck on one problem:** While practicing for CP, it is natural for anyone to find himself unable to solve a particular problem. In some cases, however, when people get stuck, they spend hours and days trying to figure out a solution to that problem. Finally, after wasting a week, when they are still not able to solve it, they get highly demotivated and lose all interest. In such cases, it is important to know how much time one must spend on a problem. At the same time, he must also not hurry through problems. Ideally, one must give every problem enough time in which he can give his best shot at solving it. Although this ideal time might vary from problem to problem, some prefer to set a time limit of around 20-30 minutes for every problem after which they go ahead and check the editorial.

- **Not participating in contests regularly:** Many beginners refrain from participating in contests, thinking that they may not yet be ready to compete in one or take a long break from competing at times. However, participating in contests from time to time is extremely important as contests not only reflect one's current skills but also highlight his weak areas which require attention. Participating in contests is also important as it teaches you how to think and act under a time constraint.

5. From Division 3 to Division 2

You do not need to know any algorithm or advanced data structure to become a Division 2 coder. This fact may surprise most readers but it is 100% true. How do I know that it is true? Because that is how I become a Division 2 coder myself.

Once you are familiar with the basics of your programming language, if you can:

1. Find a brute force solution to a problem quickly, and
2. Implement the brute force solution quickly,

Nobody can stop you from becoming a Division 2 coder.

As mentioned before, a lot of beginners tend to learn a bunch of obscure algorithms and fail to realize that they must focus more on solving problems than learning theory. This does not mean that those algorithms are unimportant but that they are simply not useful to them at the moment. In fact, most advanced programmers are good at competitive programming not because they have memorized hundreds of algorithms but because they know when and where they have to apply those algorithms they already know.

Although it is encouraged to participate in all upcoming official contests, a few that are conducted especially for absolute beginners are:

- CodeChef Long Challenge and Starters
- AtCoder Beginner Contest
- Codeforces Educational Rounds and Division 3 contests

Among these, I personally like the CodeChef Long Challenges the most because of their longer time limits. This way, they allow participants to think and act without any pressure and perhaps enable those to reach their true potential who face difficulties in timed contests. Moreover, Long Challenges also give participants the time and opportunity to learn new concepts and apply them during the contest. Participating in long challenges only, however, is also a bad practice as it might weaken your ability to work under a time constraint.

6. From Division 2 to Division 1

Once you become a Division 2 programmer, this when you need to start learning some basic concepts and algorithms, namely:

- Binary Search
- Dynamic programming (DP)
- Breadth-first Search (BFS)
- Depth-first Search (DFS)
- Dijkstra's Algorithm
- Binary Indexed Tree or Fenwick Tree
- Permutations (nPr) and Combinations (nCr)
- Mod Inverse
- Bitmasks
- Segment Trees

Implementing and debugging faster:

Many a time, there are contests in which the difference in difficulty between some trivial problem and the next easiest problem is substantial, leading to the rank list largely being a reflection of speed or the time taken to solve the easy problems. This makes the quick implementation and debugging of a code extremely important.

Targeting your weak areas:

Do not keep practicing a single type of problem. Instead, start targeting your weak areas (you can find out those by participating in contests regularly) and practice problems based on those topics. As mentioned before, to improve, you need to constantly push yourself to work outside of your comfort zone.

7. Where to learn from?

A major problem with beginners is that they spend a lot of time wandering around on the internet in search of the best resources to learn from. However, instead of wasting time on such things, they must just go ahead and start learning from any one resource they trust. Based on my experience, The 'Competitive Programmer's Handbook' by Antti Laaksonen is the single best guide that is available on the internet for free and covers pretty much all major algorithms and data structures. Another point to be noted is that learning the same topic from multiple places is just a waste of time as eventually each one of them will be teaching the same concepts. Instead, one must focus more on solving problems related to that topic to get a good hold on it.

8. How to practice?

CodeChef and Codeforces are two of the most popular communities that host contests regularly and have excellent discussion forums. If you are looking to practicing problems from random topics, head on to their practice sections and start solving problems that are rated slightly above your current rating. This way, you can keep yourself from wasting your time on trivial problems and can make sure that you always remain out of that comfort zone I have talked about before. You can also filter out problems under specific tags if you are seeking to solve problems that are based on a particular topic. HackerRank is another good platform that I have used personally to practice problems on different topics.

Virtual Contests:

For those who wish to participate in contests without any pressure or fear of losing their rating (although they must not!) or just wish to participate in more contests than those that are scheduled, virtual contests are the best option. They help you by providing you with an experience similar to that of an ongoing contest through a real-time simulation of a past contest.

Upsolving contest problems:

In my opinion, participation in a contest is a waste of time unless you upsolve the problems you weren't able to solve during the contest. After all, the very reason you took part in the contest was to learn something new and if you do not

upsolve problems, there was no point in appearing in the contest anyways. This advice may seem fairly counter-intuitive to many since what is the probability the same problem will be asked again in a future contest? The answer is close to zero but that is not the purpose of upsolving. Trying to solve problems you weren't able to solve before will give you an idea of how to approach such problems in the future and what way of thinking you must employ to solve such problems that are of a higher difficulty.

9. Some more valuable tips and tricks

- Whenever you come across a new algorithm or concept during practice or in contests, do not hesitate to learn more about it from the internet. GeeksforGeeks is one such website that has an exhaustive list of tutorials on various algorithms. In fact, I even maintain a diary where I note down all these new concepts for future reference.

- Try to learn from codes and solutions written by higher-rated participants. This way, you are bound to come across some valuable tricks and techniques along with some efficient and varied approaches that one could use to solve that problem.

- Always use a pen and paper during contests. This is an extremely useful habit as it allows you to pen down your thoughts and approach during the contest, apart from providing easier visualization of the problem statement and various test cases.

- Never give up mid-contest. Even if you think feel that you have a fairly small chance of being able to solve the next problem, stick with it until the end. You never know when the solution might strike you. Simply walking away is the worst thing you can do during a contest. If you are a true programmer, then you must have the courage to fight till the very end!

- Make a routine for practicing CP problems every day or every alternate day. This way you can avoid irregular practice sessions with large gaps, and ensure that you do not lose touch with CP at any point in time.

- Take up a positive attitude and most importantly, have fun!

10. Conclusion

As of August 2021, there are over 290,000 registered Division 3 users on CodeChef but barely over 5,000 Division 1 coders. The main reason behind this surprising statistic is that almost 99% of users who begin with Competitive Programming do not receive the right guidance. Instead of practicing, they spend hours on the internet in search of that ultimate secret formula that can make them a Division 1 programmer overnight. As a result, when they do not see any quick improvement in their performance and rating, they lose all motivation and simply give up.

This handbook is a cumulation of all the tips, tricks, and lessons I learn on my journey to becoming a Division 1 programmer. Like many, I learned most of these lessons “the hard way.” Therefore, it would be more than my pleasure if I could help out any beginner by trying to make his CP journey smoother through this book. Thank you for reading and I wish you good luck and a high rating. If I can do it, so can you!