[Saurabh Khanduja](https://www.quora.com/profile/Saurabh-Khanduja), B.Tech. Computer Science and Engineering, Indian Institute of Technology , Dhanbad (2012)

[Updated Jul 16, 2017](https://www.quora.com/What-is-the-best-coding-competition-site-for-beginners/answer/Saurabh-Khanduja)

This is the answer, I wish I had when I started.

For a beginner, it’s difficult to know where to start when there are so many choices. So, I would suggest this plan:-

**First 2–3 months**:- Start with [HackerRank](http://www.hackerrank.com/" \t "_blank). Why?

HackerRank has some pretty good ui with boilerplate code pre-written. So, the beginner gets to focus on writing the logic part only. Being easy to start means you are less likely to give up. Next, it has pretty high inflation, i.e. your rank increases the more you practice, in spite of solving less complex problems . Don’t take this negatively, and think of it as a reward system for keeping you happy in initial stage. Finally, HackerRank practice section divides problems well into proper domains, thus you know how well you perform on various sub-domains such as disjoint set/Trie/etc. Complete all practice section problems and +points if you participate in few competitions(at HackerRank). And don’t get distracted with anything else. When some big competition is coming, simply say “I don’t care”. For the same competition will repeat next year and unless you are in top 100(arbitrarily high number) already at HackerRank, the chances of doing very well is really low.

**Don’t forget** to focus on your studies academically and build a well grounded foundation. See below examples where you get to implement knowledge learned from architecture or data mining into coding competitions. **Bonus:**HackerRank also has other domains such as on AI, Distributed Systems/etc. which will help you in academics.

**Examples:-**

1. Clustering can reduce run-time easily in some problems. Optimize for a million query, find the closest point to some arbitrary (x,y) in a set of a fixed million points.
2. Tricks, like modulo should be rarely used for its high operation time.
3. Discrete Mathematics, Graph Theory, etc. are simply at the core of many problems.

Stick to solving one problem a day at least. It doesn’t take that long.

**What next**:- As of now, you should have strong background in academics(DS/Algo at least), with good experience in all variety of problems. Now, its time to attenuate your love for HackerRank. Maybe, you have reached a high reputation(in top 200 or 500), and don’t really want to leave and start from bottom again. But do it anyways. Its time to contemplate on a bigger picture. Maybe you want something big and bold in resume, something that will matter when applying for a job or research opportunity. Or maybe it’s for your intellect thirst. In any such cases, you have even more reason to leave HackerRank.

First, let’s start with getting to know major competitions around the world, and stop focusing on the major competition sites. Some examples : Codechef’s SnackDown, Google’s CodeJam, ACM ICPC, Facebook Hacker Cup, etc. If you think it’s too much to track, there are Apps which provide updates about future coding competition event. Another approach is to subscribe on all these annual events site and you will have the announcement mail for the events once the time comes.

When you participate in these competitions, there is a good chance you will get through only 1–2 rounds. You might feel down, but don’t. Even if you could qualify only first round, that’s good. It means early days of HackerRank practice paid its dividend.

Now, comes the really hard part, improving yourself. Continuous improvement is hard. If you think you can do it working non-stop, you are highly likely going to be horrible on some important parts of your life such as school. And worse, you might exhaust yourself and never return later. Be realistic and make a time schedule of different competitions happening on codechef, hackerrank(yes going back to it as well) and topcoder. When there are too many events happening, pick one and stick to it. There is no dearth of these competitions and you will have another event a week later at most.

At the end of each competition, do two things.

1. Read editorial of the problem which you could only solve partially. Analyze why(not which) you made that coding/logic error which gave wrong answer or study thoroughly the concept which you didn’t know about. Regarding **why(not which)**, I wrote so because it’s not the missed rule or constraint that prevented you from solving the question, but it was the hurry and lack of focus on reading the problem.
2. Check how top 3 coders solved the same problem as you did. See how their code differs. This will help you not only understand how the best minds in the competition works as well as some cool programming tricks which takes years to learn/collect. These tricks can provide performance improvement to reducing the amount of time it takes for you to code.

**Time for the Annual Competition:** It’s time. There is only one month left. It’s time to start practicing daily for a longer period. Improve your speed, implement DFS,BFS, etc. in 10 minutes or even less. Practice problems from past contests, like [Google CodeJam 2008](https://code.google.com/codejam/contest/32013/dashboard#s=p0). And on final day, give your best.

Before I end, be very strong in mathematics. In computer science and other sciences, mathematics play a crucial role. It not only will help you in Coding competitions, but will play a major role later in life, whether in career - machine learning, data science, image processing - or in research.