m.sabouri's programming blog

Saturday, October 25, 2014

My recipe to improve your programming skills

Hello everyone,

Sometimes I'm asked about how I learned competitive programming, and what do I suggest to people who are new to competitive programming. I'm obviously not one of the greats of TopCoder or ACM, but I'm going to share my opinions anyway in case someone's interested.

Initially when you start competitive programming, you need two very very important tools before even thinking of improving your problem solving skills. Without these two skills, I think there's no point in trying to solve harder problems in order to improve your problem solving skills.

1. Excellent coding skills. There's no point in solving problems which you cannot easily implement.
   Unless you only intend to participate in a team where someone else implements your ideas, there's really no point in solving the hardest problems with elegant solutions if you just cannot implement them. Even if there's someone in your team who can implement your solutions, I highly recommend against it because of the inefficiency of doing so.
2. More knowledge, being familiar with most algorithms and classical problems out there. In most 5-hour contests, I usually use the solutions and ideas I already know.

   Say, I solve 8 problems in a contest. Usually 7 or even 8 of them don't require anything besides those algorithms and ideas I've already used in earlier problems. Those ideas are hidden somewhere in the problem, and that's where problem solving skills become important. Unless you are familiar with those algorithms, there's no way you can come up with a solution for more than 2 or 3 such problems.

   I'm by no means recommending you to memorize the problems you solve and their solutions. My point is that if some problems requires something such as "meet in the middle" and you have never heard of it before, in most cases you cannot solve it. But if you have used it in some problems already, you have a much higher chance of recognizing it in the contest.

So, how do we improve these skills? We eliminate problem solving for now!

To improve your coding skills, solve as many problems as you can solve in 15 minutes or less. Find problems you already solve most of the time in around 15 minutes, like TopCoder's Div2 Medium, or Codeforces Div2 1000, and solve as many as them as you can.

It might seem easy and useless work to you. But it helps building your speed. Compare it to running! How many miles can you run in 15 minutes? How much is it going to improve after you practice running for 15 minutes 4 days a week for 3 months? The same goes for programming too! After you solve 100 easy problems in 15 minutes each, you can solve harder problems in that timeframe, maybe the ones that used to take 25 minutes to solve.

You should also invest some time in solving problems which don't need any special algorithms and are just a matter of implementing them. Like simulation or brute-force problems. These problems should take considerably more time to implement, maybe in 45-75 minutes range. (I am just making up these numbers, they are by no means the ultimate numbers you can choose)

For gathering more knowledge, you should solve A LOT of problems at a level which you cannot solve now, and don't spend more than 10 minutes on thinking on them. Then, you

About Me

Mohammad Reza Sabouri

G+ Follow 152

View my complete profile

Blog Archive

▼ 2014 (2)
- ► December (1)
- ▼ October (1)
  - My recipe to improve your programming skills

should read the editorial or judge solutions, and see how the problem is solved, and you MUST implement them.

When I was blue in TopCoder, I was unable to solve most Div2-Hard problems. But the problem was not my problem solving skills. The reasons was I not familiar with most algorithms, like Maximum Bipartite Matching, Minimum Spanning Trees, Dynamic Programming using bitmasks and so on. Obviously, spending 2 hours trying to solve such problems was not a good idea for me. So what did I do? I solved as many of them as I could with help of editorial.

Another great source of problems are regional contests. Go and find problems which many teams have solved, but you cannot find the solution in 10 minutes. Then read the solution, and implement it. Those problems are usually very easy if you are familiar with the algorithm needed. If you cannot solve it easily, you probably don't know the algorithm, or you have only solved a very few problems using it. NEERC subregionals, NWERC and SWERC are my favorite contests.

After practicing as I described for a few months, you will see that you can solve more problems which everyone solve, and solve them quite faster than before. After a while, you can feel that you are familiar with most solutions and ideas. By familiar I mean you have solved and implemented at least 7-8 such problems. After a while, you can spend more time thinking before looking at the solution, because you are more likely to be able to solve the problem.

I am assuming you usually participate in some contests. It really helps you evaluate yourself, and practice your problem solving skills a little. Participating in 5-6 SRMs and Codeforces rounds, and also 2-3 ICPC style contests is a good idea in my opinion.

So to wrap it up, I have these suggestions for new coders.
1. Solve problems which take at most 15 minutes of your time.
2. Solve problems which are just a matter of implementation, but they are quite hard and take around one hour to implement.
3. Solve algorithmic problems at a level which you cannot already solve. If you couldn't find the solution in 10 minutes, read the editorial, AND implement the solution.
4. Slowly start to add more problem which you spend more time thinking.
5. Make sure you regularly participate in live contests. (Or use Codeforces' Virtual Contest)

Suggestions are more than welcome to improve this article.

Thanks,
m.sabouri

Posted by Mohammad Reza Sabouri at 12:37 PM G+