How to use codeforces Effectively

By Nishant Chahar & Ek bhaut acha dost uska!

Channel Link: https://www.youtube.com/c/NishantChahar11

How to use CF and other online judges in order to get good at competitive programming?

1. **Which OJs should I use?**

   CF and Atcoder for practice.
   CF, Atcoder and topcoder for contests.
   //Codechef short contests are also good.

   Regarding Codechef Long:
   I personally don't like codechef long contests. It's quite subjective and would vary from person to person. That being said, it is quite good for beginners who are learning programming.

2. **Which problems to solve?**

   CF:

   Problems rated approximately 200-400 more than your current rating. Keep in mind that this is not a rigid window.

   Say your current rating is 1850, using the filter in the problemset page select problems in the rating range 2000-2200. If you are able to solve many problems with rating 2000 easily, you can shift the window to 2100-2300. If 2000 rated problems are too tough, do problems in the range 1900-2100.

   In my opinion the window size of 200 is good as there would be a good mix of easy and tough problems.

   Atcoder:

   Use this tool to find problems of suitable difficulty. Atcoder generally has good quality problems (the tougher ones like last one or two problems of ABCs).

3. **Should I keep tags visible while practising (general practice and not any specific topic)?**

   Strict no.

   Many problems require us to observe some properties and formulate the problem as a graph / binary or ternary search problem / DP problem and then code it. If you keep the tags visible you get a big hint.

   Say the problem requires you to identify some states and transitions and the code a DP solution. But thinking of some (wrong) greedy solution is very intuitive. In such a case just knowing that this problem is to be solved by using DP is a huge hint. While you would solve the problem and add one to your count of problems solved, you might not be able to get to the solution in a similar problem in future.

4. **I just learnt some topic. Should I search problems by tags and solve them to get a good grip over what I've just learnt?**

   Not advisable. As said previously. Many problems on CF require deducing some logic and then formulating the problem as a graph/DP/range query problem. As you've JUST learnt the topics it might happen that you won't be able to deduce how to apply this topic to this problem.

   A better way would be to search for some blogs on the topic. These blogs would have links to problems and their solutions. Try them first and improve your understanding of the topic. After this practising as mentioned in Q-1 should be fine.

5. **When to read the editorial?**

   Depends on a lot of factors. I personally first see the problem tags for a hint after being clueless for 20-25 minutes. If I'm still not able to get to the solution I read the editorial.

   Ideally, you should read the editorial after solving every problem. The editorial might have used a better approach / might have provided formal proofs to their greedy solution / might have given a bonus problem - eg: can you do it in O(1) space?

6. **What all contests should I give?**

   Depends on your rating. If you're a beginner then you can give all contests and solve the few easy problems. Give the codechef long contests. They are good for beginners.

If you have some experience, say 1700-1800+ on CF you can skip CF div-3 contests as they would be too easy. They won't be rated for you anyway. You can also leave the Codechef long contests if you want.

Codechef short contests, some atcoder contests and majorly CF contests. Topcoder is also a good platform. Problems are of high quality. Definitely worth trying.

7. **My rating is not increasing. What should I do?**

   In decreasing order of importance:

   - Practice using the strategy in points 1 and 5.
   - Participate in contests.
   - Upsolve.

8. **I am a beginner. How should I start?**

   Learn programming. Codechef contests. Long also.
   CF div 3
   ABCs

   Once you are comfortable with programming. Say you can solve the implementation based problems during the contest time. Then you can learn new topics. Binary search, greedy, DP, etc.

   For more details refer to
   https://drive.google.com/file/d/1J2x8pIYQ3MXANgvzOgBciWd3d79j_Exa/view

9. **Are virtual contests helpful?**

   They are less effective when compared to real contests. The pressure is not there. There are no tangible rewards/punishments either... Gain or loss of rating points

10. **In what order should I learn to get good at competitive programming?**

    https://drive.google.com/file/d/1J2x8pIYQ3MXANgvzOgBciWd3d79j_Exa/view

    This PDF is what you need.

    A high level plan might look like this.
    Implementation (STL data structures) => Greedy, binary search, DP => Segment

tree, sparse table, DSU => Graphs => Advanced topics like DP on trees, bitmask DP, 2-SAT, flows, string algorithms, etc.

11. **What are some good resources to study algorithms?**

    https://cp-algorithms.com/
    Some CF blogs
    Some hackerearth blogs

    Use google search for whichever topic you want to learn. Eg: "DP on trees Codeforces"

    Of late there has been a lot of content on youtube and some other websites. But I haven't checked them out yet. Therefore, I'm not recommending those. You can try them out if you feel like.

12. **How to get a good rank in Kickstart or other big contests?**

    Be good at CF type contests. With 2000+ rating, getting a sub 150 rank in Kickstart shouldn't be very hard. (As of April 2020. This may change as more and more people get a 2000+ rating on CF)

13. **How to prepare for ICPC?**

    It depends on how well you want to do. If your goal is to just get to the regionals then you need to be among the top few teams from your college in the preliminary contest.

    To get a good rank in the regionals and to have a chance to qualify for the next rounds the following should be enough:

    1. Have a good team:
        1. Decent ratings on CF for all three members. Say 2000-2100+
        2. The team members should complement each others skills. Everyone would be having some strong topics. In these topics they can solve tougher problems. If their rating is 2000, they might be able to solve problems rated 2300 if the problem is related to their strong topic.

           If the team members have diverse strong topics, then as a team, they would have their bases covered.
    2. Have many team practice sessions. You can give virtual contests of previous regionals on the CF Gym.