UCF Programming Team: Practice Tips Guide

by Arup Guha

(originally written for Coursera Course: Learning How to Learn)

Abstract

UCF runs a very successful ACM-ICPC (http://icpc.baylor.edu/) team. Over the course of the last 30 years, our team has always finished first, second or third in the South East Regional. In the last ten years, we have qualified for the ACM-ICPC World Finals all but one year. This past year (2014), we were 21st at World Finals, the 3rd highest finish amongst the North American teams. Most of this success can be attributed to the amount of practice that our students put forth, independently. The goal of this paper is to help our students maximize the benefit of those many hours they put in, so they see the best contest results possible, given the amount of time they are willing to practice.

Introduction

Each year UCF runs a Local Programming Contest (http://lpc.ucfprogrammingteam.org/). The results of the contest, coupled with the knowledge that the coaches have about the participants from classes and past years of competition are used in choosing UCF's teams. Once the teams are chosen, each Saturday includes a lecture for all students followed by a five hour mock contest for each of the teams. In addition, each team is assigned a designated coach. That coach closely monitors his team, meets with them weekly, and gives his team resources, guidance and extra practice problems to help the team get better. Other coaches post problems for all team members to work on during the week, in addition to the unsolved problems from the Saturday practice. Largely, team members are free to choose the manner in which they conduct this additional training during the week, with limited monitoring from their individual coach.

In this paper, I will aim to incorporate the ideas from the Coursera course, "Learning How to Learn: Powerful mental tools to help you master tough subjects" to give suggestions to our team members about how to best spend their independent practice time, during the week.

Summary of Relevant Ideas from "Learning How to Learn"

We have two learning modes: focused and diffuse. The former is good for tasks you already know how to do, but that need concentration. The latter is better when you are acquiring new skills or are attempting something more creative. Real learning requires moving back and forth between these two modes.

Sleep is critical for learning. The unconscious mind strengthens important neural connections, weakens unimportant ones and clears toxins from the brain. In addition, occasionally, if primed properly, dreaming can lead to more creative ideas that solve problems you hadn't previously been able to solve.

Spaced repetition is better than cramming. Since your brain has to iteratively build a solid infrastructure of new material going between the focused and diffuse modes of learning, exceedingly long cramming sessions aren't best for overall learning. At some point, no more can be learned in a single session. Instead, practicing several times over the course of days or weeks, allowing the natural process of sleep to iteratively strengthen the appropriate neural connections into long-term memory tends to be more successful for learning.

Exercising allows for the growth of new neurons and naturally allows for the diffuse mode of thinking. Though we don't fully understand the full effects of exercise on learning, much of what has been observed is positive.

Recalling newly learned information is actually more effective for learning than just rereading that information. The former is more active. Furthermore, just reading solutions, rarely leads to learning and often gives the student a false sense of understanding.

Interleaving studying between different types of problems or different subject areas helps promote deeper learning. Truly understanding a concept means being able to recognize its use out of context. If you can only solve a system of two equations in two variables when you're presented that section of the textbook, you haven't fully learned the concept. Rather, interleaving your studies allows your brain to make more creative leaps and get better and seeing disguised versions of problems you've previously learned about.

Occasionally, a simple pattern that one has learned well prevents her from having a more complicated thought needed to solve a new problem. This phenomenon is called "einstellung", the German word for installation. In these situations, we have to unlearn the old way of doing things to make room for learning new ideas.

Habits are very powerful things. The habit loop involves the following four steps in sequence: a cue, a response, a reward, and a belief. We tend to go into our habit mode only when certain cues trigger us. Once we see that cue, then we respond in a trained, automatic way. We do this because we've learned from the past that this specific response will lead to some reward. Completing these steps reinforces beliefs that we have about our behaviors. The key here is that if we identify negative habits, the most effective way to change them are to either remove the cue, OR work on changing our response to the cue, if the cue is something inevitable in our environment.

Making lists of items to complete is important. It helps frame the mind for what has to be done. Completing the least savory task early tends to be best approach for time management.

Finally, when working on difficult material, it's critical to take breaks periodically and reward ourselves for accomplishing subgoals.

List of Proposed Techniques for Programming Team Training

Each student has a different personality, so it's unlikely that all of the following tips will be appealing to any of them. However, each student should find that they can incorporate some subset of these tips into their training during the week.

1. "Primed Exercise" – routinely exercise (any exercise will do), but before you set out to exercise, read a problem statement of a problem you haven't solved yet. This will prime your brain with something to think about while you are exercising. The natural diffuse mode of thinking that ensues during exercise may help you think of new approaches to solving the problem that you hadn't previously considered. In my experience, this also makes the exercise seem a bit less strenuous on the body, since your mind is wandering, thinking about the problem vaguely, instead of focusing on the fact that there's a little pain in your foot.

2. "Primed Sleep" – Nearly identical to the idea above, before you go to sleep, read a problem description and think about it for a bit. Once you sleep, you'll find that occasionally, you'll dream up ideas you hadn't previously thought of to attack the problem.

3. "Problem List" – Always maintain a list of unsolved problems you want to work on, in the order you want to attack the problems. Keep this list in a place that you frequently see (perhaps on the clipboard you take to all of your classes) and cross off problems as you solve them. Make sure this problem list has problems that are properly calibrated for difficulty. These problems should be ones that appear to be just out of your reach. Save them from past practices and contests. You must not put ALL unsolved problems on this list. Then it will become too unwieldy and unfocused. I recommend about five problems on the list at a given time, with a secondary list of 10 to 20 problems. When the primary list whittles down to one or two problems, move up some of the problems from the secondary list.

4. "Daily Practice" – Set aside 1 or 2 hours every week day to practice for programming team, instead of trying to do most of your practice on one day. Though you'll spend the same amount of time with either technique, the former will lead to better long term learning. Also, many times, if you get stuck on a problem on Monday after working on it in a focused mode, the time away from the problem until Tuesday may help the diffuse mode kick in and give you new ideas when you attack the problem again the next day.

5. "Sleep and Food" – before every competition, make sure you sleep close to 8 hours and have a good nutritious breakfast. This one's a no brainer that everyone should do, regardless of personality.

6. "Changing Contest Habits" – Some students don't react optimally in contest when they get a "Wrong Answer" response. Other students have some difficulty in the last two hours of the contest. Pay attention to your suboptimal habits and see if you can figure out what triggers them. Then, attempt to put a plan forward to break that particular habit loop. For example, some students will work on the same problem for hours, even when it might be better for them to give up on that problem and solve a different problem in the set. In this case, we must identify what causes that habit. Perhaps after two "Wrong Answers", we might actively force ourselves to read two other problems in the set before we allow ourselves to think about the previous problem. This tip is the most difficult to implement because different students have vastly different habits and there is debate about which habits may or may not be good for contests.

Conclusion

Our team members have already mastered many difficult topics. Yet, there remains room for them to make their practice time more efficient. Incorporating the ideas from “Learning How to Learn” into their training regimen may improve their performance in contests by helping them better understand the algorithms learn and practice and better bring out combination of diffuse and focused mode necessary to do well in programming contests.