Training of the Carnegie Mellon Teams for the ACM Programming Competition

Gregory Kesden **Eugene Fink** **Daniel Sleator**
gkesden@cs.cmu.edu **e.fink@cs.cmu.edu** **sleator@cs.cmu.edu**
www.cs.cmu.edu/~gkesden **www.cs.cmu.edu/~eugene** **www.cs.cmu.edu/~sleator**
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, United States

We will describe our approach to training the Carnegie Mellon teams for the regional and international programming competitions. We have developed this approach in the last five years, and it has helped improve the team performance and gradually move from the bottom 30% to the top 30% in the ACM international competitions. All training materials that we have used are available on the web at www.cs.cmu.edu/~eugene/teach.

Since Carnegie Mellon is a home to a large number of leading research projects in Computer Science, its traditional culture strongly encourages undergraduate students to participate in cutting-edge research, and most members of the competition teams are involved in research and related advanced coursework in addition to their competition training. To follow this culture, we have made a long-term decision to limit the training of programming teams to about 200 hours per year, so that the team members have time for research work and other extra-curricula activities. We therefore focus on the optimal use of this fairly limited training time.

We have organized the training as a "club" with weekly three-hour meetings, open to all interested students; the attendance is usually between thirty and sixty people. The main activity during these meetings is competition-style problem solving, sometime individually and sometimes in teams. We use the PC^2 environment for these mini-competitions and announce the winners in the end of each club meeting. We allow students to get elective-course credit for their participation; they can earn 3 credits (which is equivalent to 1/3 of a standard full course) for participating in all meetings during one semester, and additional 3 credits for problem-solving homeworks. If a student participates for several semesters, she can earn this credit every semester. We do *not* give any "required" work and present all tasks as optional; the more tasks a student completes, the more course credit she earns. We provide free pizza during all meetings, which has proved to be a powerful incentive for voluntary hard work.

In fall, we select the most talented students for participation in the regional competition; identify potential superstars among freshmen, who are not yet ready to be competitors but may become team members in future years; experiment with different team compositions until constructing optimal teams for the regional competition; and work on basic teamwork and competition skills. In spring, we shift the focus to training the international team and ask the members of that team to work on problems of international-competition difficulty, whereas the other students are still solving regional-level problems.

During these club meetings, we maintain the focus on practical problem solving and do not spend much time on basic theory. We have observed over the years that motivated students usually learn all basic algorithms on their own, whereas less motivated participants ignore algorithms even if we spoon-feed them. We however run an *additional* two-hour "discussion session" every week, focused on advanced theory, which is open only to the team members and sometimes a few other especially strong students. The number of attendees at these additional meetings is between six and eight, and the focus is on discussing a wide range of hard competition problems. The meetings are organized as brainstorming sessions, where all attendees collectively analyze problems until finding solutions. We use problems from past international competitions and TopCoder, as well as our own "home-made" problems. We go over all major problem classes from the past international competitions; identify and discuss a wide range of advanced algorithms; and build a "team notebook" with critical algorithms. We encourage students to implement the discussed solutions and empirically evaluate their performance as part of their homework; for each problem, one or two students usually write a program for solving it.