On Teaching Computer Science: Thoughts and Advice for TAs

by David Chiu

When I sat down in my TA training session, I was already on my second cup of coffee. As I perused the day's handouts, about halfway down the page was something that would not slip my observation: "A good teacher teaches students, not content." Maybe I graduated to a new intellectual level that morning (perhaps it was either the coffee or the fact that the quote appeared in large bold-faced type), but for whatever reason, I couldn't help mulling over it. For the next hour, instead of focusing on my training, I considered its meaning and how I would develop the philosophy in my own classroom. The following are a few of my thoughts and lessons-learned from my experiences as a TA.

Getting Students Involved

It's our nature—we love to categorize things. When we're at the grocery, we choose juicy-looking peaches over the rotten, beat up ones. At the bookstore, we purchase books from our favorite authors as quickly as we dismiss those from writers for whom we don't particularly care. At all levels of education, the most prolific student project is probably the ever-evolving clandestine database of good and bad teachers. If not for the sake of good teaching, then certainly for the sake of saving face, it should be in every teacher's best interest to leave behind a mark of positive legacy.

When I think about those whom I tossed in my own "bad teacher" basket, I realize that they all share common aspects. In particular, they tend to think that teaching is best conducted using a simplistic model: *I talk, you learn*. For many students, this unilateral model doesn't make a whole lot of sense. From my own class in the past: Ross, my most vocal student, learned well through class discussions. Michelle, who sat in the front, best learned through visualizing concepts, and Pat, in the back, from hands-on activities. The point is, students are a heterogeneous bunch. Teachers must adapt to the different styles of learning to accomodate students, not the reverse. It would be irresponsible to think that the students collectively should change the way they learn to fit the teacher. With this in mind, one of the most poignant questions a teaching philosophy must address is: *How do we better engage all of our students?*

Having the Usual? No Thanks!

I begin each new lesson by first justifying what I'm about to teach. It's every math teacher's nightmare, and computer science education is similar enough to make such a justification challenging at times. Here's why I do it: before investing their time in a particular subject, students need to understand just why they have to learn what they are about to learn. In response, we should leave no doubt in their minds that the lesson at hand is relevant.

Particularly for those teaching introductory computer science courses, we need to pull in ideas from various disciplines. It takes no time at all, for instance, to browse through the class roster to get a gist of your students' majors. So be prepared to tell your scientists and statisticians exactly how loops are used for implementing prediction models. Show your artists the importance of matrices for realizing digital images. Students crave to know, as they are naturally inquisitive, exactly what they are getting out of each lesson.

Every instructor also needs to explore ways to include a wider variety of deliveries, even if it takes time. For example, as opposed to projecting a piece of code on the screen and blandly discussing each line, superimpose a control flow diagram and work through the algorithm visually. Rather than assuming that students automatically absorb abstractions, we should provide nuanced and concrete examples. For instance, we could facilitate this by running an algorithm with a test input, tracking each variable and explaining how/why conditions are met. When teaching recursion, illustrate the call stack and trace return values. Time consuming? Absolutely. But I think time is better invested here than later if future lectures start crumbling as a result of their dependencies on key topics built on shaky foundations.

Lastly, we should try to find ways to make lectures more interesting. Every time I ripped apart a phonebook to demonstrate binary search, my class filled with laughter. I'm sure their enthusiasm stemmed not from the skinny kid struggling to tear a phonebook in half, but from the excitement of seeing the search space exponentially diminish. All jokes aside, though, this has got to be more exciting than any binary search routine you can draw on a chalkboard. Not to mention, can you think of a better use for phonebooks nowadays? Another time, we demonstrated various sorting methods by splitting the class into groups and having each group stand in front of class to act out their assigned sorting algorithm on the basis of their height. This not only encouraged them to actually understand the procedures, but it also stimulated group work. By making my classes more interesting, it promoted attendance, attentiveness, interaction, and critical thinking.

Use Discussions when Possible

An underutilized but effective way of engaging students in computer science classes is through classroom discussion. Students interested in topics best facilitated with varying thoughts and opinions, especially those with social/ethical connotations, benefit greatly from discussions. However, let's be reasonable: Getting students to talk openly about computer science can oftentimes be painful, and depending on the lecture, discussions are not always possible or even desirable.

In the past, several strategies have worked for me for facilitating effective discussions. First, ground rules must be established. Although it is sometimes difficult, it's important for the instructor to remain unbiased on controversial issues, e.g., computer security and privacy, by arguing for both sides. Also, you can avoid dominance and intransigence by volunteering students to talk rather than having an open discourse. Most importantly, remember that to a student, there is nothing worse than the fear of being humiliated in the presence of their peers. This is why most of them won't talk. So make it clear to them that they (and you) must only be critical of arguments and ideas, and not people. Finally, make sure that the right questions are asked. For instance, instead of asking yes/no questions, which are often terminal, try to use more why's and how's.

Caring to Care

I honestly believe that the most important factor for encouraging student engagement is to involve yourself. I've learned that the sooner a mutually respectful relationship can be established with your students, the easier the classroom is managed. This doesn't mean you need to go out of your way to be their best friend (in fact, I strongly oppose this). I find it's easier than that. Treat the class as a whole like you'd treat a new colleague or acquaintance. Be casual and respectful. Care enough to be interested and attached. For example, when reading the news, it takes little effort to digress and think if it might relate to any of your lectures.

A personal rule I make for myself is to always read the day's events, particularly tech articles from **slashdot.org**, as they might relate to future lessons. Above all else, it might simply add anecdotal value to your lectures so they don't seem canned and monotonous. It also has helped me to arrive a few minutes early just to chat with my students. Letting them know that you are human (at least most of the time) makes it less intimidating for them to approach you. The rapport you earn pays off dividends when you realize the rewards it brings: ease of classroom management, a livelier classroom environment, and kids actually showing up to class—just to name a few.

"A Good Teacher Teaches Students, Not Content"

As I emptied my second cup of coffee and I packed up to leave, I once more questioned why I found the teaching philosophy so compelling: "A good teacher teaches students, not content." Maybe it was the memories of my own poor teaching in the start of my graduate program that invoked this introspection. I've always thought that the most auspicious moment in my teaching was taking a moment to realize, had I been a student in my own class at the time I was a budding TA, that I probably wouldn't have enjoyed it much. In response, I conceived a new ambition and ultimately was empowered to improve. In other words, I learned to start caring about my students and about my teaching.

To be honest, I still don't know what it takes to be a good computer science teacher. But I do know what it takes to be a bad one. So I figure, if I avoid bad teaching habits, care about my students, and just use some common sense, then I will be on the right track to being an effective instructor and TA.

Biography

*David Chiu (**chiud@cse.ohio-state.edu**) is a PhD student in the Department of Computer Science and Engineering at the Ohio State University. Over the last four years, he has taught several introductory level computer science courses at Kent State University and Ohio State University.*