

Few human activities are as difficult as teaching. Whether you ask a carpenter or a computer programmer, very likely they will agree on this issue. It seems so much easier for us to do what we do best, than to explain to a total novice in simple and precise terms how we do it, why our way is best, or what the novice should do to become proficient most easily and quickly. Whether the specific occupation is very practical, or involves manipulating abstract concepts, teaching it requires patience, empathy, responsibility, method... Did I mention patience? Yet, this difficult job lies at the basis of our very survival and evolution, both individually, and as a species. Each of us gets to do quite a fair share in a lifetime, and some even seem to excel at it. Is there a secret?

I hadn't thought about this until my first such experience, while being a TA for *Introduction to Computer Science*, at University of Bucharest. Leading computer labs for groups of students with very diverse backgrounds, but much alike in their eagerness to learn and progress, is a big challenge. Especially in the beginning. In the lecture hall, and even in the recitation section, the (physical) distance tends to blur differences. Labs, on the other hand, involve a direct and much more informal interaction, and there you can immediately see whether "they got it" or not. There you can also be so flexible to explain in more comfortable terms some abstract theoretical concept, or an algorithm, or even some erroneous result. There, in the lab, I realized how remarkably different the learning experience is for everyone. I myself learned a big lesson: knowing your students, and adapting to them by meeting them half-way whatever their specific needs may be, will earn you their trust. This is an essential ingredient of a good collaboration.

This principle generalizes across all forms of teaching, course topics, or schools. I applied it just as well at Bucharest (where I was also TA for *Algorithms and Programming Techniques*), as I did later at Carnegie Mellon University. Here, in Spring 2004 I had the opportunity to TA for *Graduate Algorithms* (taught by Manuel Blum). As during the previous year I had taken the class myself, I was quite familiar with the mechanics. And yet, getting involved in the organization of a graduate course assumes, more than anything, a good sense of vision and outright maturity. When proposing new candidate topics for addition to the syllabus, or when designing homework and exam problems (and even when I decided to announce the most flexible office hours), I was only trying to find the answer to one question, for myself: What would I like (or need!) to take away from this course if I were a student? This question was particularly important given that quite a few students in the class came from engineering, biology, or even economical backgrounds, and were interested in learning potentially useful things to their own research. Changing the perspective made me understand their needs, and in many cases – anticipate their "next question"; consequently, I was able to serve them much better.

Next Fall I was again a TA for an *Algorithms* class: this time it was the undergraduate course taught by Avrim Blum and Manuel Blum. I could probably rank this as the most instructive of my teaching experiences. Big class. Demanding crowd. Excellent lecturers. Great teamwork. Recitations prepared down to minutiae. Oral grading. (Actually, all possible kinds of grading). I improved my public-speaking skills *a lot*. I improved my understanding of native- and non-native-spoken English *a great deal*. But the most important lesson for me was to observe (in fact, to realize for the first time) how people's minds literally change, how they grow as a result of one course. Most students are receptive, inquisitive, and alert to begin with, but getting used to a new type of thinking can be tricky at times. It is the teacher's responsibility to trigger the appropriate thought mechanisms that enable the students to discover a new way of seeing the world. Probably the most successful strategy is to provide them with the *right questions*, as opposed to the *correct answers*; thus, instead of storing information students are trained to explore, reason, debate, and create their own answers (sometimes - of rare elegance and beauty).

I think that the sole principal indicator of success in teaching is how much do students themselves feel that their view on the world and on their place in it, has changed when they leave the course. The teacher being extremely rigorous, or more indulgent seems not to matter that much, as long as he *inspires* his students to learn. The thorough instructor who does not offer high-level guidance will have less impact than a colleague who neglects some of the details, but favors the big picture. In my opinion, a good teacher will always know his students, and thus will be able to adapt the course plan to their necessities and even to their interests. Guessing what these are is, if not impossible, at least very hard in the absence of direct and open communication, and just as importantly, without seeing things from their perspective. Therefore, I think that a good teacher should himself be open-minded, eager to absorb the new and the unusual, and always remain conscious of his higher goal – to educate.

Is there a secret to teaching? No, not really. I think that these rather common sense principles I mentioned here should enable anyone to become better at sharing their experiences (and expertise!) with others. In my case, I intend to rely on them whether I will be teaching theoretically oriented topics (like *Matrix Theory*, *Wavelets*, or *Algorithms*), as well as more applied ones (such as *Information Theory and Coding*, *Pattern Recognition*, or *Signal Processing*). I am eager to advise my own students, to help them get a good and healthy sense of what research is, and to teach them what I myself have learned about science, collaboration, independence, publishing, responsibility, and most importantly, about what it means to find your own way.