Teaching Statement

Eric Price

December 14, 2012

Philosophy. In my experience, the most important component of a course is the exercises, both theoretical (a.k.a. problem sets) and empirical (a.k.a. labs). With lectures and readings, most students only passively accept knowledge; exercises make the students take the active role necessary for true understanding and retention of the material. Undergraduates have many competing demands on their time, and regular problem sets are necessary to keep them engaged.

Labs make abstract theory much more visceral. While this is especially so for physical labs (I still vividly remember the smell of "magic smoke" from undergraduate EE), it remains true for labs on a computer. A lab demonstrating the differences among linked lists, binary search trees, and hash tables lets the student themself tinker with the parameters, getting much more intuition than a demonstration in lecture can instill. Labs even remain important for graduate students: pure theoreticians can all too easily ignore constants and polylogarithmic factors.

It's also important for a course to accomodate a range of student quality. My senior year, I was a teaching assistant for 6.02, MIT's "Introduction to EECS II". 6.02 was a fairly new course teaching a broad overview of communication systems, from analog radios to digital radios with coding for error correction and compression to networks with packet routing. It's a great vision that ties together the EE and CS parts of the department, and as a TA I learned a lot. Unfortunately, the students did not learn nearly so much. The course was paced for the bottom third of the students, and bored the rest because it avoided exercises that more than a few students could not solve. As a result, it seemed like very few of the students bothered to learn the overarching vision of the course.

The solution, I think, is to include exercises that challenge students at all levels. Good differentiation of the difficulty level both makes the exercises more rewarding to the students and gives the most information for grading purposes. One complication is that the standard grading system is designed for situations where the vast majority of students are capable of solving almost all the problems, not situations with frequent tough problems that you expect most of the class to miss. There are a number of methods for alleviating this, including optional problems, optional hints on problems, and very coarse grading on problem sets.

Courses. The introductory course I am most interested in teaching is algorithms. I fell in love with the content of the course as a freshman in high school doing the USA Computer Olympiad. As a high school upperclassman, most weeks I spent an hour teaching the material to younger students on our computer team. In undergrad, when MIT was revamping its computer science curriculum, I helped design the new Introduction to Algorithms class 6.006. In comparison to the old class, 6.006 features an emphasis on coding—hence it is both more similar to the computer olympiad that taught me so well and has more of the labs that I value in a class. The last couple years I have been a coach for the computer olympiad, teaching algorithms to bright high school students.

For graduate classes, I would be happy teaching algorithms, data structures, coding theory, information theory, and of course streaming algorithms and sparse recovery.