

I have enjoyed my teaching experiences, and I look forward to future teaching. Starting as an undergraduate student, I have taught sections, graded assignments, and held office hours for a variety of courses. My experiences, together with conversations with professors at Harvard and Tufts, have refined my understanding of what it means to teach effectively.

The most effective teachers I know have designed courses not by focusing on the subject matter implied by the name of a course, but by thinking about and prioritizing the ideas and skills that students need to practice and learn. For example, in a course on data structures, the teacher may present several representations of balanced binary trees, but the goal is not for students to memorize the details of each representation—the goal is for students to learn how an invariant on the representation of a tree can limit its height, guaranteeing efficient insertion and lookup. An effective homework assignment might ask students to practice using invariants by building an implementation of red-black trees and showing that the implementation maintains the invariants of a red-black tree.

In my early teaching experiences, I saw that when a course is not focused on ideas and skills, it is hard to predict what students will learn. For example, I assisted in a course which, as a capstone, required students to implement a garbage collector. For the students, the challenge was to build and debug a large program. The students had previously written only small programs, and they had not been taught how to build and debug a large program incrementally. Fewer than 25% of the students built working garbage collectors, and it was not clear what they learned.

If I were using the same assignment today, I would focus not on the garbage collector but on skills used to build and debug large programs. In particular, I would teach students how to use an invariant to debug a program compositionally. In a garbage collector, the central invariant states that every live value is reachable from a global variable or the stack. Because any function that violates the invariant is wrong no matter what other functions do, students could find most bugs by checking each function individually to see that it maintains the invariant. By teaching the students how to debug with invariants, I would expect not only that more students would complete the assignment, but also that students would practice and learn a skill which would help them throughout their careers.

In a graduate course, it is also important to focus on ideas and skills, but the ideas and skills are different: graduate students must learn to understand and build upon published research. In my experience, a common model of graduate seminars, in which one student presents a research paper to the rest of the class, is highly inefficient: only the presenter practices the skills required to understand and discuss the paper. I have observed that these skills can be practiced more efficiently among small groups of students, where the instructor has given each group discussion questions that are crafted to highlight interesting features or difficulties in the paper. This model allows each student to practice the essential skills of distilling information from a research paper, thinking critically about research problems, and communicating with colleagues.

I can teach a broad range of courses covering both theory and implementation. Through my research, I have extensive experience with programming languages and compilers. I am eager to teach not only courses in those topics, but also advanced courses in run-time systems, concurrency, optimization, and static analysis. I am also prepared to teach courses in computer architecture, systems programming, data structures, and algorithms. When I have more experience, I will also be very interested in teaching introductory courses, although I feel that beginning students are best served by more experienced teachers.