Teaching Statement: Sean Treichler

Teaching comes in many forms, and the right recipe is as individual as the teacher, the student, and the subject matter in question. While classroom lectures remain the most efficient way to initially introduce material to students, my preference is to balance it with interactive discussions, hands-on project work, and opportunities for students to teach some of the material they have learned through presentations and discussion groups (both online and face to face).

While at Stanford, I twice served as a teaching assistant for the undergraduate compilers class. My favorite part of this experience was office hours, in which I was repeatedly challenged to find alternative ways to explain difficult concepts or guide students towards their own way of understanding them. I often finished my office hours feeling like I understood the material better as well. More recently, I had the opportunity to teach the compilers class myself during a summer session. The significantly smaller enrollment allowed me to make lectures more interactive, adjusting to the needs and desires of the students. However, it also reminded me that learning how to teach is itself an ongoing process that I have only just begun.

The field of computer science is growing constantly broader and deeper, yet the instructional time available, especially at the undergraduate level, is relatively fixed. It is simply not possible to include all relevant material in a course syllabus. Many students turn to the large amount of teaching and reference material available online. This material should not be a substitute for coming to lectures, but at the same time, it is a valuable resource that students should learn how to take the most advantage of. I would like to explore explicitly incorporating such material into a course by having lectures focus more on the fundamental principles and the ways in which the concepts connect to each other. Starting from a suggested set of online resources, students would then be expected to dig into the concepts in depth on their own time and at their own pace, maintaining a critical eye to identify the inaccuracies that are unfortunately all too common in online material. A review of the key material in a subsequent lecture establishes the baseline of what students are expected to understand and allows discussion to cover areas the students are most interested in.

Based on my studies and experience, I am comfortable with teaching courses that cover compilers, operating systems, networks, graphics, and compute architecture at either the undergraduate or graduate level, as well as introductory courses in theory and algorithms. In line with my research interests, I would ideally like to teach a broader "compilers" course that explicitly includes the roles of libraries, runtimes, and even system architecture in the process of realizing a programmer's description of a desired computation on a given machine. The rise of interpreters, just-in-time compilation technologies, as well as hardware features such as transactional memory allow tasks traditionally done in a compiler to be moved up or down the software stack based on the needs of a particular problem domain. The fundamental principles are the same, and the ability to adapt ideas from one setting to solve problems in another is one of the most important skills that I think one can teach.