Teaching Statement

Neil Lutz

As computation becomes increasingly central to almost every aspect of our lives, the education of future computer scientists continues to grow in importance. To meet the challenges ahead, we need a generation of computational researchers and practitioners who are knowledgeable and conscientious, with a strong sense of purpose for their work. In my teaching, I prioritize clear exposition, active learning, building on students' varied backgrounds, and presenting material in the context of our dynamic field and its broader impacts.

Every course that I have taught has had students with a wide range of educational backgrounds and interests. I know from my years of tutoring in Chicago and Newark that a single unfamiliar term, concept, or notation can derail a student's learning in a major way, and that this phenomenon disproportionately affects students who already feel out of place. For this reason, I aim to make my courses as self-contained as possible: I review notation and terminology early in each semester, and I begin each class with a recap of relevant definitions and tools. Further, I ground abstract material in concrete examples, in both lectures and exercises. When I have taught algorithms courses, universal hashing has been one of the most challenging topics for students, especially those with less rigorous mathematical backgrounds. To help with this, I recently assigned the class to programmatically test a particular hash family for universality. This was not a difficult exercise, but it clarified the concept for my more application-oriented students.

Learning a technical subject as a passive observer is very difficult, so I always push students to engage with the material early and often. I encourage participation in lectures by calling on students by name and offering positive feedback. I have also learned the value, in addition to traditional long-form problem sets and projects, of straightforward practice exercises with much quicker turnaround. These let students internalize basic concepts and definitions before applying those concepts to more sophisticated synthesis and creative problem solving. In courses with labs or recitations, this usually involves having students work in small groups and then discussing solutions as a class, with an emphasis on the process by which they arrived at those solutions. In larger classes, I have implemented low-stakes, automatically graded electronic quizzes. These quizzes have helped me to quickly identify topics that the class might need to revisit, and students have found them useful in preparing for regular problem sets.

Following my first two semesters teaching undergraduate algorithms courses, I received some feedback from students about lack of motivation and difficulty hearing my voice, particularly when I was writing on the board. To improve student motivation, I have focused on outwardly expressing my enthusiasm during lectures, with positive results; my Fall 2019 teaching evaluations included the comment, "One key feature that distinguishes him is his enthusiasm for the material and why the material is enjoyable rather than tiresome. It was authentically interesting to be in class compared to when I normally need 4 shots of espresso to make it through all 9 am courses I have previously had." After my efforts to simply speak up did not adequately address the volume concerns, I have begun using microphones more often and posting lecture notes online. The latter change allows me to spend more time facing the class instead of writing every detail on the board. My fall 2019 rating on the statement, "The instructor speaks clearly and audibly when presenting information," was 4.63 out of 5.

For the past year, my teaching has been primarily remote. My spring data structures course at Iowa State was online after spring break, and my fall algorithms course at Penn was online from start to finish. I have used relatively short, pre-recorded lectures, supplemented by extra office hours, a very active presence on Piazza, and initiating direct contact with students who appear to be struggling. This has been a difficult transition, but I have still seen strong, positive outcomes for student learning. Students have been complimentary of the clarity of my lecture videos and the level of communication about logistics and expectations.

A computer science education is fundamentally empowering, and I communicate to my students that they are acquiring potent tools to understand and shape the world we live in. By tying the material to recent and ongoing research, and by telling them open problems related to course topics, I show students how close they already are to the cutting edge of our field. I show them that the gaps in our knowledge are exciting, and must be taken as direct invitations to try something new. I use the same approach in undergraduate research mentorship. Last year, I helped mentor three students as part of a Google exploreCSR program for promoting undergraduate research for women at the University of Pennsylvania. They studied algorithmic fairness as applied to the unintended learning-theoretic downstream consequences of college admissions policies, presenting their research in a well-attended talk at an end-of-semester workshop.

I hope to continue teaching primarily theoretical courses that complement my research, on topics such as algorithms, data structures, discrete math, and the theory of computing. I am comfortable teaching—and have taught—at both graduate and undergraduate levels, and I particularly enjoy helping computer science students understand the foundations of our field and apply their analytical skills to more abstract domains.