Teaching Statement

Darren Strash
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
strash@kit.edu

October 7, 2015

Teaching Philosophy

My teaching philosophy is that students should be *constantly challenged*, students should *learn by doing*, and their work and understanding should extend *beyond the classroom*. For my teaching, I make every effort to make content clear and to make the presentation as dynamic as possible. This means that I work on the whiteboard a lot. When I do present slides, I don't read from them; they are used as a guide. My intention is for the class to be one long conversation, including all the spontaneity and back-and-forth that is typical of conversation. I frequently ask questions (using the Socratic method), and give many asides to help build intuition. Most importantly, I make every effort to present a well-rounded view of the subjects I teach: why we choose to study it, its history, and its impact.

A challenge for everyone Having a dynamic classroom means changing the curriculum to meet the needs of the students. In particular, all students should have the opportunity to be challenged by the difficulty of their tasks. This means I actively watch for students who are excelling, and offer them work that will challenge and motivate them. This also means I offer extra support to those who are clearly struggling by providing office hours and by encouraging group study. When students are inclined toward either programming or research tracks, I offer projects that enhance these skills, so that students are prepared to meet their career goals. Ultimately, my goal is to challenge everyone so that they can achieve their potential, and build the skills that will carry them into the future.

Skills for the future While mastery of curriculum is necessary for students achieve their career goals, there are other skills that are equally important for a successful career. In particular, I work to give students the following skills for the future: critical thinking, effective communication, and the ability to work in groups. To enhance critical thinking skills, I give clear, coherent methods for solving problems, I illustrate the relationships between various problems and solutions, and I give exercises that require the creative use of knowledge. For communication and group skills, I give always give an assignment that includes a written and presentation requirement, and whenever possible I give at least one group project.

Impact outside the classroom Though in-class-only projects are necessary to build up students' skills, students can lose motivation and feel that they are wasting their time being bogged down in projects that lead nowhere. That is why I plan to offer projects that have real-world impact.

For those students who are inclined towards programming careers, there are many opportunities to learn from, and contribute to, open source projects. These are tools that I want to use in the classroom. For software engineering and programming classes, I plan to use open source projects as the basis for at least one project. For software engineering, I will encourage students to evaluate the code design and suggest further design improvements. For programming classes, I will have students pick open source projects, become experts in some (small) areas of the code and implement algorithms, design improvements, and additional features.

For those students on a research track, student projects should ideally include a *real* research component. Understanding and evaluating previous works are essential for starting research, but they are not research in themselves. Ideally, a student's work should have the potential to result in a publishable research article. This not only helps students to feel that they haven't wasted their time in a class (when they could be furthering their own research agenda), but I feel that students should have many early research experiences so they can evaluate if the research track is right for them.

Experience and Readiness

This semester I am the lead lecturer for the Master's level computational geometry course at KIT. This includes preparing the materials and giving a majority of the lectures for the course. During the previous semester, I developed and lead a Master's level research seminar entitled "Algorithms for Large Social Networks in Theory and Practice" at KIT. In addition to running the seminar, I also supervised several students in the seminar, helping them analyze existing state-of-the-art research and present their findings in a write-up and presentation. The feedback from my seminar was overwhelmingly positive: many students positively remarked that they had never received such detailed and helpful feedback from any other instructor before.

Aside from these two recent examples of my teaching experience, I also taught as a teaching assistant and as a guest lecturer during my doctorate program at University of California, Irvine. There, I served as a teaching assistant in lower-level algorithms and data structures courses for a total of one year. I treated each opportunity to teach as if I were the sole instructor for the course: I prepared all of the teaching materials, I created many homemade examples, and I actively sought out other materials to give variety and best illustrate the current topic. During one class in particular, I ran two simultaneous discussion sessions with over 90 students combined.

Given my current teaching experience, I am prepared to teach beginning to advanced algorithms and data structures (both theory and practice), computational complexity, discrete mathematics, security, programming in C++, C, Java, or Python, software engineering, and operating systems courses. With some further preparation, I could also teach cryptography, game development, distributed computing, algorithms for massive data, and scientific computing.