Teaching

Why I Teach
I teach because I love helping people learn; because I can be effective in helping students overcome obstacles they thought they could not surpass; and because people feel good about themselves when they learn something difficult.

I teach computer science because I have developed deep understanding in certain areas that I would like to share; because I find abstract structures and algorithms to be fascinating and beautiful; because computer programming is a useful skill; and because the precision and clarity-of-thought necessary for computer science are important in all academic domains.

What I Teach
I have taught undergraduate computer science courses at UCSD and at the University of the Western Cape (UWC). These include lower-division courses on data structures (CSE 12 at UCSD) and upper-division courses on operating systems (COS 315 at UWC) and theory of computation (COS 345 at UWC).

Undergraduate level: My overarching goals when teaching undergraduate computer science are to convey (1) the skill and thrill of computer programming and (2) the joy of scientific discovery:

Computer programming requires that the programmer express her-/himself unambiguously to a machine. Learning to program can be useful to students of all disciplines because it teaches how to communicate more precisely. In my view, good programmers must thoroughly understand the basic principles of control flow, data representation, and memory management. They must be fluent in the canonical data structures and algorithms, and in meta-algorithmic techniques such as recursion and dynamic programming. Finally, they must know how to structure their code to be efficient, elegant, and easily understood by others.

Computer programming is not only extremely useful for building software. It also constitutes the set of “laboratory skills” through which computer scientists conduct empirical research, e.g., by writing a software simulation of influenza infection, or creating a test program to measure the accuracy of a face detector. Starting at the introductory level, I strive to show my students the science in computer science. For example, in CSE 12, I designed a programming project in which students wrote code to infer the identity of a set of “mystery data structures” (e.g., hash table, binary search tree) whose identities were hidden. By conducting experiments to measure the asymptotic time cost of basic operations (add, find, and remove) and using simple statistics to filter out noise, the identity of each structure could be ascertained. For many students, this modest project was an enlightening experience on the purpose of programming.

Graduate level: As a PhD student I had the pleasure of mentoring 6 Masters students at the University of the Western Cape on automated face analysis and gesture recognition projects. I also co-advised an excellent UCSD research assistant, Karmen, on her research on automated teaching systems. When mentoring students in research, I strive to teach two skills: (1) how to choose a research question that is scientifically interesting, realistically attainable, and personally fulfilling; and (2) how to tackle a
specific research question by devising and conducting experiments, and interpreting their results. At the undergraduate and early graduate levels, I typically provide strong guidelines for the research topic itself, and focus my students’ efforts on skill #2. For example, when I suggested to Karmen that she use logistic regression to model teachers’ eye gaze patterns during math tutoring sessions, she made rapid progress: not only did she learn some particular machine learning techniques, but she learned about experimental design in general. After students have demonstrated some success in tackling a research problem, I begin to encourage them to formulate their own research questions.

**How I Teach**

My teaching method derives from the principle of *trust*. To lead my students successfully through a course, I must build and maintain their *trust* that I know the material deeply, that I know how to guide them through it, and that I have their best educational interests at heart. Students must know that I will respect them equally whether they are star A+ students or they struggle to earn a C. While working with brilliant students is exhilarating, I often find the pedagogical challenges of teaching weaker students to be more satisfying. One student commented in his/her evaluation of my teaching performance (CSE 12 in 2010) that “the TA is amazing. He explains to the level that you’re at and doesn’t make you feel stupid.” I believe my respect for every student regardless of performance is what engendered this sentiment.

Building trust starts on day 1: While every lecture and classroom activity must be carefully constructed, the first lecture must be especially clear, illuminating, and confidently presented because it sets the tone and expectations for the rest of the course. I also begin the first class of every course by asking all students to introduce themselves briefly and answer a question such as “What was your favorite computer science assignment ever?” These introductions give me the chance to look at each student in the eyes and show her/him that I am listening.

Outside of the classroom, I strive to create trust from the “bottom up” by working with students 1-on-1. As an instructor I always spend some time walking through the lab or the department hallways, asking my students how they are doing and whether they want help. I enjoy it, and the students seem to appreciate it. I was gratified and deeply honored when one student, J.C., from CSE 12 in 2012, wrote me in an email that “You’re honestly one of the best teachers I’ve had, you put more effort into the class more than any other teacher.”

When giving a lecture or talking with students individually, my objective is always to provide *clarity*, i.e., precise, unambiguous, illuminating explanations. In my experience, both as a university instructor and also as a volunteer tutor working with troubled youth, receiving a clear explanation and finally understanding something is one of the strongest motivators to learn more.

While teaching I find two pedagogical techniques to be useful: One is to establish that there is a problem before offering a solution. For instance, in COS315, I first tried to convince my students that memory fragmentation was a real problem before describing how operating system page tables can alleviate that problem. Another technique is to “force” students to reflect on the curriculum by issuing unannounced anonymous quizzes during class. Every few class periods, I give students a ~10 minute quiz in which they answer questions similar in style to the final exam. After they submit their responses, we discuss the solutions interactively. Students appreciate the engagement and opportunity
to assess their current understanding. The anonymity helps them to feel comfortable submitting the quiz, and I can still gain valuable feedback on the class’ overall progress. The value of “testing” versus “studying” is also supported by a growing body of empirical evidence [1].

Summary
I am a passionate, dedicated, experienced teacher who cares deeply for his students’ educational well-being. I would be thrilled to be a professor and teach students not just for a semester, but throughout their undergraduate and graduate school careers.

References