Teaching Statement

Fan Long

November 2016

Teaching is one of the most important activities of academia. As a growing and ever-changing field, I believe teaching in computer science is not only about transmitting knowledge to students but also about inspiring the students to inquire and learn independently. It is a process of collective improvement for both the students and the teacher. I want to become a professor because I am enthusiastic to engage in this exciting process with the best young talents of our next generation.

Teaching Experience My first teaching experience in computer science started when I was a senior competitive programmer in my high school. My coach asked me to give weekly lectures to junior students about standard algorithms that are often used in programming contests, such as dynamic programming and max flow algorithms. Two of the junior students that I taught became as successful as I was in programming contests – they both went on to win gold medals in the International Olympiad in Informatics (IOI), the most prestigious programming contest for high school students.

During my undergraduate studies at Tsinghua University, I worked as a teaching assistant (TA) for the course *Introduction to Programming.* I prepared programming problem sets and held office hours to help students debug their submitted programs. During my graduate studies at MIT, I worked as a TA for the course 6.035 (*Computer Language Engineering*). I organized and graded a large compiler project, as well as prepared and graded two mid-term and one final quizzes. I am always eager to teach and to practice my lecturing skills. I volunteered to serve as a supplement lecturer for courses 6.035 and 6.033 (*Computer Systems Engineering*) at MIT.

Teaching Methodology I believe that a central goal of teaching a subject is to nurture the ability of students to inquire and learn the subject themselves. Achieving this goal is particularly important in computer science, because computer science is one of the most rapidly changing fields. Many programming languages and software techniques that we are using today did not exist twenty years ago and most likely will be obsolete or superseded twenty years from now. Nevertheless, the ability to learn can serve a lifetime. From my personal experience, I learnt all of Java, Python, C#, and Objective C programming skills by myself from online tutorials and empirical projects. I believe the success of a career in computer science largely depends on one's ability to keep up with the rapid innovations happening everyday in the field.

To realize this goal, I will design courses that encourage and nurture "knowledge exploration" for students rather than "knowledge transmission". From my past teaching experience, interactive components such as open-ended problem sets and large projects serve this purpose better than traditional lectures and quizzes. I am a strong proponent of using large open-ended team projects for teaching computer science subjects. There are at least three major benefits. First, working on a large project encourages active learning. When I worked as a TA for the MIT compiler course 6.035, many students told me that they learnt more from finishing the large compiler project than from listening to the lectures. Secondly, there is always a gap between the theory and the practice. There is no better way to close this gap than actually to implement the learned theory. Lastly, doing projects is a more attractive way for students to learn. It gives the students the satisfying feeling that they invented and created something on their own.

To ensure the success of a large project assignment, the project should be broken into many step-by-step incremental milestones so that a steady learning curve can be achieved and the course staff can monitor the progress of each team. The project should have a unified framework for fair grading but the framework should impose as little technical restrictions as possible to realize the full potential of each team. More importantly, assistance and guidance from the course staff should be accessible for every student along the way so that students do not stray too far during the "knowledge exploration" process. I followed these guidelines when I organized the large compiler project for MIT 6.035. The results were extremely successful – all participating teams in the course finished with at least working compilers and all except one teams implemented advanced dataflow optimization features in their final compilers.

Teaching Interests My past teaching and research experience has covered a wide range of topics in computer science including software engineering, programming languages, security, algorithms, machine learning, and operating systems. Given the need, I am qualified and ready to effectively teach undergraduate courses in any of these subjects. I am particularly eager to teach undergraduate and graduate level courses that are directly related to my research projects. Examples of such courses include *Software Engineering*, *Software Reliability and Security*, *Compilers*, *Program Analysis*, and *Systems Programming.*

I would be exciting to explore the opportunity to teach Massive Open Online Courses (MOOC) in computer science given the need. In fact, I believe that many program analysis techniques used in my past research projects could be applied to address important challenges of organizing MOOCs for computer science. Examples of these challenges include how to effectively and scalably grade programming assignments and how to automatically provide feedbacks to the submitted programs from students.

Advising Approach I always enjoy helping and mentoring junior students. At MIT, I have helped to advise one master thesis. I formally or informally advised one high school student, two undergraduate students, and four junior graduate students. One important challenge of advising is to establish an effective communication pattern with the students. For example, having regular weekly meetings might be effective for some students but unproductive for others. Another important challenge is to decide how much I should get myself involved into the technical details of student projects. When advising a student, I prefer to play a guiding role from the start and actively help the student to determine high level technical directions. Once the project is on the right track I switch to a consultant role and allow the student to take responsibilities of making all technical decisions. The goal is to nurture the ability of the students to productively and independently do research on their own.