Mentoring young researchers and teaching students are among the most rewarding parts of being an academic. The feelings of seeing a student's eyes light up as they grasp a difficult concept, or seeing a young researcher start to take off and develop their own ideas are special and are core to my goal of being a professor. The mentors and teachers who guided me have had an outstanding impact on my life; as a faculty member, I intend to pay this debt forward and continue my current trajectory of mentoring young researchers and teaching students.

Being a Research Mentor

While I am still early in my journey as a mentor and advisor, I have developed a set of principles that guide how I approach my interactions with students. First, I try to be a fluid combination of a teacher, collaborator, and advisor to students — at every stage of a project, a student needs something different. At the inception of a project, I try to be a teacher to get the student knowledgeable enough to make progress, while in the later stages, I transition to collaborating with the student on the design and implementation, as well as advising on strategy and dissemination of the work. Second, I pay extra attention to understanding and staying up-to-date with the lower-level details of student projects whenever possible so that the students feel that there is someone who is able to discuss and work through key technical challenges in their work. Finally, my long-term goal is for students to become independent researchers and engineers themselves, which I approach by leading students to ask the right questions about their work and instilling confidence in them to proceed without me.

I have had the opportunity to apply these principles in advising undergraduates as well as younger graduate students during my PhD. Some of these undergraduate researchers have made substantial contributions to the Legion project and my overarching research vision. Joseph Guman worked on understanding the sources of overhead within task-based runtime systems; his initial exploration and prototypes have led to a larger research undertaking that is now under submission for publication. Rohan Chanani investigated developing GPU-parallel algorithms for partitioning operations within the Legion runtime system. His algorithms have already led to order-of-magnitude speedups and are being requested for use in NVIDIA's cuPyNumeric library.

Teaching In The Classroom

Within the classroom, I have developed a philosophy for teaching that is influenced by my experience as an eight-time teaching assistant at Carnegie Mellon (functional programming and parallel algorithms) and at Stanford (compilers and programming languages). My teaching philosophy is centered around properly motivating students and building an understanding of the course material. To motivate students, I believe in grounding coursework with a focus on why the coursework matters and how it may be applied to real problems students may face in their future. While this motivation naturally differs for specific subject material, I believe establishing this connection early on and repeatedly is important to begin the learning process within students. I am a proponent of hands-on project work whenever applicable (particularly for programming courses), where students build real systems as part of courses. Project work motivates students by showing that classroom concepts translate into real artifacts; seeing a compiler generate real code, or watching a parallel program speed up with more processors, are tangible results that students are excited to see. Additionally, building non-trivial projects reinforces classroom concepts and requires students to learn course material beyond a surface level. I approach teaching in a bottom-up manner, where I build up understanding of core concepts from first principles. I start with examples and generalize to the overarching concepts that students can apply beyond the classroom. When interacting with students in one-on-one settings, I work to meet students where they are in their understanding of the material and aim to give them the necessary tools to make progress. I believe in separate strategies for teaching undergraduate- and graduate-level material. The goal of undergraduate courses is to expose students to material that has withstood the test of time and has emerged as our community's accepted approach to problems. In contrast, I believe the goal of graduate-level seminar courses is to expose students to the boundaries of knowledge. That means understanding the messiness at the frontier, where even we as researchers are not sure of the answers, or even the best ways to think about the problems we face. For seminar courses, I hope to instill in students the intellectual tools for properly evaluating, critiquing, and being skeptical of research papers.

Beyond the classroom, I have also worked to make course-management software that simplifies course administration while enabling more integrated student interaction with course materials. At Carnegie Mellon, I contributed to the inception and development of the Diderot course management platform. Diderot combined common utilities such as lecture note hosting, discussion boards, and code autograding into a single piece of software, simplifying course management and enabling student interaction with and discussion within course lecture notes. At its peak, Diderot was used by roughly 1500 students across multiple courses within Carnegie Mellon's computer science department.

Courses

Based on my background and experience, I would be excited to teach both introductory and higher-level courses on parallel computing (algorithms and systems), programming languages, compilers (construction and optimization), and performance engineering. I would also be happy to teach introductory courses on programming, computer systems, algorithms, distributed systems (the cloud computing, fault-tolerance, and consensus side of the field), and machine learning compilers. I plan on developing and teaching seminar courses aligned with my research: supercomputer programming techniques and the design of domain-specific languages (DSLs). The goal of the supercomputing course would be to expose students to recent developments in high-performance computing technology and alternatives to bulk-synchronous programming techniques. In the DSL course, I aim to teach students the principles behind DSL design and implementation, and to explore successful DSL designs in recent years. The underlying theme of both courses would be confronting the ever-growing complexity of modern machines with advances in programming and runtime systems.