Han Dong — Boston University
Teaching Statement — Fall 2024

INTRODUCTION

Henry David Thoreau's writings have always reinforced my belief in the ideals of a liberal arts education. I believe this is even more important in an era of rapidly changing technologies, such as advanced AI models. In that spirit, my teaching goals center on helping students 1) become adept at integrating computing across disciplines while being conscientious of their societal impact and, 2) develop a solid scientific foundation that allows them to combine their theoretical understanding of a topic with the practical tools necessary to embark on a career of lifelong learning. Thoreau said, "*the more slowly trees grow at first, the sounder they are at the core, and I think the same is true of human beings*." My goal as a teacher is to encourage this growth in my students.

TEACHING AND MENTORSHIP EXPERIENCE

As a teaching assistant for *CS 210: Introduction to Computer Systems*, I led weekly discussion sections, creating programming exercises for hundreds of students that emphasized the foundational knowledge needed to advance in the course. As a bonus, I often included an exercise to demonstrate the application of their knowledge in different contexts, from disciplines such as scientific computing to the arts. One of the things that I learned from teaching this course is that new undergraduates are often so focused on completing a specific problem, they have difficulty adapting to unexpected information. To better prepare them for the unpredictable I often lectured on how the bits and bytes they had learned in class have real-world impacts on many of the complex systems they interact with daily.

As a teaching assistant for *CS 451/651: Distributed Systems*, I led and designed discussion sections for over sixty students who had to complete a semester-long project on consensus protocols. As these were often upper-level students, I prepared material that focused on helping them develop practical debugging and logging tools to tackle complicated distributed systems. In hindsight, though the material was challenging, I learned that imparting students with the tools to dive into a sophisticated problem empowered them to develop the individual skills and confidence needed to take on future engineering work.

Throughout my academic career, I have mentored four undergraduate and two graduate students, advised a student on their master's thesis, and provided support to many others for class projects in advanced courses. As my students and mentees came from diverse and international backgrounds, I found it was important to dedicate time to pair programming sessions to gauge their progress and solicit feedback on areas where I can improve and on ways to better accommodate their unique learning style and needs. Thanks to these conversations and experiences, I learned to anticipate and be adaptive to various needs, such as providing flexible in-person and remote mentorship.

TEACHING PHILOSOPHY

As I've reflected on my experiences and the role of computing in our world, the following goals and guiding principles have become central to my teaching:

1. **Fostering curiosity and deep understanding through interactivity:** I plan to instill an understanding that there is a difference between using advanced AI models to solve a problem and deeply engaging with someone to thoroughly understand the problem and solution - an endeavor that is often messy and even recursive. To emphasize this, I aim to foster a sense of curiosity in my students, and help them cultivate the desire to delve into the inner workings of computing. One way I strive to achieve this goal is by broadening students' understanding of the role of computing across various disciplines. Beside the standard methods of teaching, such as lectures and programming exercises, I plan to leverage advances in embedded devices to create assignments and projects that deepen students' interactions with computing and underscore its pervasiveness. Overall, my aim is to both cater to students with different learning styles and lay the foundation for how the theoretical and practical can come together in a physical setting.

2. **Increasing confidence and independent problem-solving through experimentation:** As the scale of computing is ever-growing, it is important that courses train students to be adaptive and confident in their abilities to unravel the complexities of modern computer systems. In the course of my work in the Open Education Project, a partnership between Boston University and Red Hat, I found that leveraging open source tools to access server-class hardware can provide a practical environment in which students can experiment with and deepen their understanding of complex systems.

3. **Individualizing learning through one-on-one interactions:** The ability to communicate directly with students through office hours or pair programming sessions is one of the most effective ways to enhance a student's learning experience. I find this not only helps students who are hesitant in a group setting, but also allows us to discuss our thinking process and learn from one another. I believe building this rapport is especially important in a liberal arts institution with a smaller student-to-faculty ratio.

4. **Promoting diversity, equity, and inclusion:** I strive to facilitate and increase student access and opportunities in computer science. From my mentorship experiences, I learnt the importance of this by building confidence in my mentee's in a personalized way. My hope is to foster an environment that will encourage my future students, especially those from underrepresented groups, to advance and thrive in this field.

STUDENT ASSESSMENT METHODS

To assess my students' performance, I typically use programming exercises, exams, and class projects. In the past, I built automated grading systems for programming exercises, which enabled students to receive prompt feedback on their progress. As a result, students and I were able to catch problems early on and clarify their understanding of the material.

Though I plan to use written exams to test students' theoretical knowledge, they must be written with clarity to ensure students who might struggle with this format are still able to demonstrate their knowledge. Class projects are another important component, as they allow students to engage with the material in a creative way. Additionally, working collaboratively on a project not only prepares them for the real world, but also allows the sharing of ideas between students from diverse backgrounds. These projects also help students build up their public repositories, which they can show to future employers.