Teaching Philosophy: Computer Science is part of Everyday Life

Justin Li

The hardest part of writing a computer science teaching philosophy is to not write a life philosophy.

Computer science and teaching have always been part of the backdrop of my adulthood. I wrote my first computer program in ninth grade, and peer-tutored my first students in my sophomore year of college. Since then, I have mentored other peer tutors, taught four sessions at the Center for Talented Youth (CTY), served as a teaching assistant for four college courses, and consulted for other teaching assistants in their courses – all in computer science related subjects. Teaching computer science has been a continuous influence on my life and has critically shaped me as a person. If I had to summarize my past decade in a teaching philosophy of computer science, it would be that neither teaching nor computer science exists in a vacuum. Computer science concepts can be seen outside the digital world in such mundane phenomena as the organization of library stacks and the strategies of board games. To teach and to study computer science is to direct students' natural curiosity towards these patterns, and in turn let this knowledge provide a foundation for their other activities. These core beliefs drive all my interactions with students, from first capturing their interest to guiding them towards independent research.

Of all the students I have had, the most enthusiastic by far were the middle-schoolers I taught at CTY who, despite being tasked with building and programming Lego robots every day for three weeks, would still want to go back to the classroom after dinner. A veteran staff member once masterfully described the program as an "academic Disneyland," and that atmosphere of joyful discovery remains the ideal I want to cultivate. While most of my students are older and have diversified their interests beyond playing with Lego's, I continue to rely on their passion to introduce them to computer science. By showing how data structures and algorithms may help students understand the real world, I want computer science to be relevant to the students' goals, whether it is to get a job, to master a tool for a different field, or simply for recreation.

One of my favorite in-class examples is how artificial intelligence helped me plan a road trip for nineteen people – by using constraint satisfaction to figure out who should be in which of three cars, while balancing the number of drivers and the social relationships. This whimsical-but-true story drew my students in while providing the context for my explanation of the algorithm. It not only showed a real-world application of an algorithm the students have only learned theoretically, but it also demonstrated the fluidity that I would like my students to develop in seeing how computer science might be a solution. Perhaps just as importantly, I like to use the story because it is an implicit example of how learning and teaching is not always dry, but can be thought-provoking and entertaining.

The final reason for telling the road trip story is because I believe that intellectual curiosity is infectious, a belief I also use to structure the classroom. This is why I often ask students to work together or to explain concepts to each other: in addition to the gains in conceptual understanding cited by education literature, I have personally seen how excited students can be when they are making progress on a problem. One of my most successful problem solving moments was when, after briefly reviewing the necessary linear algebra concepts, I asked a student to take the blackboard and lead the next several minutes of class, rotating students as necessary. I made sure the first student was enthusiastic and had a strong basic understanding of the material, and the other students followed her lead beautifully, learning from each other's mistakes and building on each other's partial solutions.

Even when talking to students one-on-one, I aim to be a model for my students by asking questions about their other interests, and sharing stories of my own intellectual digressions. Without this rapport, I would not have discovered that one of my students was thinking of quitting the engineering program, and I would not have had the opportunity to ultimately convince him otherwise. These casual conversations also allow me to introduce students to extensions of the material – such as how a binary tree generalizes to a B-tree – that I believe every computer scientist should know. Having had similar conversations with my own professors, I want to help students relate the topics of the course to the bigger picture of computer science, while conveying that learning should not be restricted to the classroom, but should be interwoven with their everyday behavior.

After the group problem solving session above, the first student told me that she was proud of herself for leading the class, and I replied that I was proud of her as well. Just as homeworks, exams, and discussions of

code provide students with feedback on their technical ability, this praise is feedback for my student's broader intellectual development. Even in more formal feedback, I like to supplement the standard questions about the rote implementation and application of algorithms by asking students to consider whether and how the algorithms are merely taking garbage in and giving garbage out. I wrote a homework question that asked students to calculate the probability that someone will receive an A for a course within a given model, then asked if the model was missing components that would render it unrealistic. I want to remind my students that their technical knowledge is situated in the larger context of the real world, and that keeping this context in mind is necessary as a computer scientist. Although students often find these open-ended questions challenging, the difficulty of seeing alternate explanations is also part of the feedback I want my students to receive.

Experience with open-ended questions, the ability to work with and build off of teams, a thirst for relevant observations in unrelated areas – these are not only traits of my ideal student, but also traits that prepare students to be independent discoverers of knowledge. Despite only currently mentoring my first undergraduate, I have already seen the struggle of transitioning from classes to research, from questions with definite answers to questions that may not be answerable. As much as helping students tackle the research question, my goal as a mentor is to guide them towards the mindset of a successful researcher, to ease that transition by demonstrating the perseverance necessary, as well as sharing more practical habits such as keeping a journal of research-related observations. Especially for my topic of research, everyday mental life is a constant source of inspiration. Taking advantage of this connection requires both noticing and remembering the insight as well as translating the idea into data structures and algorithms. I want to prepare students not just for a possible future in research, if that is their goal, but for their learning to be self-sufficient when teachers are no longer available.

Self-sufficiency applies not only to my students, but to my own growth as a teacher as well. I have learned to review my students' difficulties after a class, to adapt our next interaction to their backgrounds and personalities. I have also sought help from others: being a teaching consultant has allowed me to learn from other instructors, while I ask my students for anonymous feedback. Comments describing my discussions as "concise and incredibly helpful in understanding the material" encourages me to continue with my approach. At the same time, a teaching consultant pointed out that although I have "many students engaged much of the time," I should "brainstorm ways to move toward engaging everybody all of the time." I have since learned the value of preparing as many questions as there are students and requiring a different student to answer every question. Just as I track my research, these lessons on teaching are kept in my personal journal, both as suggestions for my own consultations and as markers of my progress as a teacher.

One sign of my maturation as a teacher is the increasing awareness that computer science needs to have a more diverse community of scholars. My experience in this broader domain is limited to leading review sessions for SHPE and NSBE, but they have made me more cognizant of the need to support gender and racial minorities in STEM. I have also sought out conversations with friends in my and other computer science programs, to better understand their perspectives and the cultural biases that exist. These brief exposures made me more attentive as a teaching consultant, making me notice a student who was slipping through the cracks of an introductory computer science class, and I was able to notify the instructor to provide support for the student. However, larger changes are needed to reverse the norm of homogeneity, and as I gain the influence to more directly shape the computer science culture, I hope to create a more supportive environment for students and scholars of all backgrounds.

I have grown up over the last decade of teaching computer science. Through these experiences, I have become comfortable incorporating active learning into the classroom, have built up a body of pedagogical content knowledge, and have become interested in the larger problems of the computer science community. But to conclude on that note would be to ignore how teaching computer science has led me to value traits like intellectual curiosity, openness to alternate explanations, and a desire for self-sufficient improvement. I want to help my students see the patterns of computer science in the real world, and in this process guide their development of the same traits. This is why neither teaching nor computer science exists in a vacuum: they exist as part of my everyday life.