Teaching Statement

Teaching Philosophy

I subscribe to a constructivist concept of learning, believing students from diverse backgrounds build on their prior knowledge through active learning, discussion, and reflection. Especially because computer science and related fields evolve as technologies, programming languages, and societal priorities change, learning how to learn effectively is one of the most pragmatic skills computer science students should develop. My approach to teaching is to first spark students' motivation to learn by connecting computer science to other areas of interest and emphasizing the social impact of computing and second to foster an environment in which students build a solid foundation of computing concepts while simultaneously practicing and developing strategies for effective learning.

Connecting Computer Science to Other Disciplines

Ada Lovelace's description of computing as a "poetical science" resonates with me as someone whose introduction to computing felt like learning a language. My high school computer science teacher taught Java 101: we made flashcards defining reserved words. I came to think – simplistically – of objects as nouns, method calls as verbs, and parameters as adjectives, adverbs, or other modifiers. I liked puzzles and did well in math classes, but being able to make parallels between computer science and other areas that interested me played a fundamental role in my decision to double major in computer science and psychology and pursue research in Human-Computer Interaction.

I have seen the importance of helping students connect computing to their other interests as a teaching assistant at a small liberal arts college and a public R1 university, especially in introductory courses. Regardless of the class I am teaching, I want students to leave my classroom thinking differently about other courses. For example, students should grasp how theoretical understanding of the limits of computing and real-world resource constraints inform ethics of applied engineering projects.

Learning from Mistakes and by Doing

Part of what leads to that growth in thinking is the opportunity to recover from mistakes. During an introductory computer science lab, I overheard one pair of students discussing their plan to convert a dictionary to a string and parse it using string methods. When they got stuck using this approach, I reminded them they had just learned about dictionary methods in class, so it might be worth checking their notes on those methods. One student said, "I guess we could try it her way," before looking up the dictionary methods and realizing for himself how much easier the problem would be if he did not limit himself to string methods. I was tempted to intervene as soon as I realized what the students were trying to do but knew by first trying and failing to get by with a limited set of methods, they would begin to gain an appreciation for abstraction and data structures.

Building opportunities to learn from mistakes into a course can be difficult. I have found providing low-pressure, anonymous opportunities to indicate misunderstood concepts are useful: short knowledge checks built into a lecture, an anonymous contact form, etc. These items can make students feel more comfortable making mistakes.

In project-based courses, frequent opportunities for feedback are especially important. My senior year of college, I worked with a psychology professor to develop and teach a course called Robotic Animals. The course required teams of computer science and psychology majors to design and build a robot with

three distinct animal-like traits the other teams should be able to infer through structured observation (e.g., aggressive toward red stimuli, attracted to light, and afraid of loud noises). Because options for traits were limited by sensor capabilities, and we did not want to bias the groups toward similar ideas, we tried to avoid giving too many specific example traits. Before students could start building their robot, each group turned in a design plan listing the group's specific plans. One group was overly ambitious, outlining traits that would be complex to implement and unlikely to come across in a way that could be interpreted by the other students (e.g., "hates light" represented both by seeking out dark areas and charging toward light areas at high speeds). We recommended this group simplify their representations of these traits or focus on one of the complex traits and break it up into three discrete behaviors. If I were to teach this course again, I would add more iteration in the design drafting process to help overly ambitious groups distill their ideas into something more achievable.

Breaking a large project up into manageable tasks is a skill I developed in graduate school. In my dissertation work, I work with and help mentor undergraduate students in computer science, informatics, psychology, creative writing, and art. This experience has helped me practice giving the big-picture view of a project, identifying goals along with students for what experience they want to get out of the project, breaking off a digestible chunk of the project that gives the student some freedom to explore a problem on their own, and checking in regularly to identify stumbling blocks or complications. This experience has been helpful for my research goals, but it also helped me fully realize the power of learning by doing in building students' confidence.

I aim to continue to improve my skills as a teacher and mentor, which has motivated me to seek out institutions with more experienced faculty who are also passionate about computer science education. I plan to work closely with other faculty to incorporate lessons learned through prior versions of courses and continuing teaching evaluations.

Teaching Plans

I am comfortable teaching any course in the core computer science curriculum, with a specific interest in introductory courses. I have experience as a teaching assistant for intro computer science, data structures, algorithms, and principles of computer organization. I also have experience as a teaching assistant for social psychology and psychology research methods and statistics, which lend themselves well to computing ethics and data science courses.

I would be especially interested in developing and teaching a course on research methods in Human-Computer Interaction. The research methods course would emphasize learning to read research papers critically, which would especially benefit students who are interested in pursuing graduate school. After an initial introduction to research methods, most of the class would focus on student-led discussions of recent papers published in the field. Students would select the paper they would like to present to the class on the day they lead discussion and use a set of prompts to guide discussion of ethical issues raised by the paper, assumptions made in the paper, and the paper's credibility, novelty, and generalizability. The final portion of the course would require students to work independently or in small groups to design their own small-scale studies, with the other students in the class serving as their participants. For this portion of the course, I would work closely with other faculty to connect this project to other courses students are taking. For example, students could use this project to conduct user tests of an app they developed in a software engineering course.