Teaching Statement

Leah Perlmutter

The culture of Computer Science (CS) education at many institutions is individualistic [15], and show-offy [3], features a defensive climate [8] and hegemonic gates to knowledge [4], and produces graduates who care less about public welfare the longer they participate [5]. Students judge their fit in computer science based on perceived values match [10]. Thus, mainstream CS culture, which arises from structural forces such as racism, sexism, gender bias, and ableism, privileges White men and keeps other groups (Black, Indigenous, and People of Color, women and non-binary people, disabled people) out of tech, reinforcing a hierarchical society that marginalizes members of those groups.

Key to diversity and inclusion efforts is to engage students studying CS for the first time, as first year experiences can attract or repel [12]. BIPOC students are less likely than their White peers to have CS opportunities before college [11], so it is reasonable to hypothesize an over-representation of BIPOC students among those new to CS when they reach college. Women are more likely than men to be studying CS for the first time in college [16, 1], and may be less likely to find it interesting because of this [1]. Against this backdrop, how can educators promote inclusion by offering engaging first-time CS experiences that scale to meet the demands of booming CS enrollment?

Culture and inclusion are enormous interlocking issues that researchers and practitioners across CS education work hard to address. As I begin my career as a CS education practitioner and researcher, my teaching outlook responds to the twin crises of culture and inclusion by educating those around me about DEI issues and applying evidence-based practices from research to foster a more supportive and less defensive climate and offer engaging and culturally relevant paths for students to approach CS.

Engagement through Active Learning

I regularly practice active learning to engage students and help them build their own knowledge. My theory of knowledge draws on Friere [7]: I can't inject knowledge into students, I can only present concepts and ideas to them and support them in building their own knowledge. Teaching in person, I consistently turn to active learning practices that normalize collaboration and the social nature of learning, such as think-pair-share and explaining a concept to a neighbor. In a course evaluation, one of my students showed appreciation for this active learning technique saying, "Leah's enthusiasm was awesome. I also really liked how she had us talk to each other about a problem." Teaching on Zoom, small group collaboration was harder, and I found that breakout rooms had the potential to disengage students, so I preferred other active learning practices, such as polls and think-share. Both remotely and in person, I value free-answer polls that allow students to connect their identities to computing and bring their humanity to the classroom. Examples I have used in class include "What is your motivation for learning CS," "What do you want to change in the world," and "What are you grateful for."

From my course evaluations, I learned about students' boundaries regarding class participation, including the extent to which active learning can make some students uncomfortable. In future teaching, I will take steps to make sure that students don't feel coerced to speak in class by offering more choices for participation and designing my active learning activities to help quieter students to acclimate to them.

More Supportive and Less Defensive Climate

I promote a more supportive and less defensive climate, informed by research on defensive climate in CS [2, 3, 8]. For example, reducing student performances of academic prowess that can alienate and intimidate peers [3, 1]. In my own research, when I interviewed students about belonging in CS, I heard firsthand from several students about how observing peers show off their CS knowledge detracted from the observer's sense of belonging. When I instructed CSE 331 (Software Design and Implementation) at UW, I announced my policy for calling on students by emphasizing that the purpose of posing a question is to give everyone an opportunity to think, not to produce an answer as fast as possible, and my intention to prioritize calling on students who had spoken less in class. This policy, informed by the work of Barker, O'Neil, and Kazim [3], resulted in outspoken students often lowering their hands to give others a chance. One student expressed appreciation in a course evaluation, commending "how she gave a lot of people the opportunity to ask questions or answer her question." Another key supportive practice is to empathize with students. I do this by validating their struggles, both in front of the whole class and in individual communication. Each student is unique -- when helping students one-on-one, it's important to figure out where they're at and meet them there.

When teaching larger courses, it's important to scale empathic and supportive practices through TA teams because TAs have the majority of contact hours with students. I have experience as an instructor coordinating a team of 6 TAs, and as a TA operating within teams of various sizes, including a team of 37 TAs, teaching 630 students. I've observed that undergraduate TA culture can reinforce a defensive climate, especially when training is led by fellow TAs. Thus, even though delegation and TA autonomy are vital for scale, it's important for instructors to be actively involved in certain things such as TA training. When it comes to hiring TAs who will have lots of student contact hours, I believe TAs with the most CS experience are not necessarily better. For example, TAs who started learning CS later may be better able to empathize with novice students.

Creative Work and Sociotechnical Curriculum for Inclusion

I believe in applying culturally relevant pedagogy to address the twin crises of CS culture and inclusion. That is, teaching computing as a way for students to learn about their own identities and create technology that matters to them [13]. I am currently a fellow in the Duke University Cultural Competence in Computing (3C) program, where I am working to increase my cultural competence and figure out ways I can address systemic racism in the world and in CS through undergraduate CS education. Offering open-ended creative work and teaching sociotechnical topics in the core CS curriculum are two strategies that I have practiced so far.

When I studied CS for the first time as an undergraduate at Colby College, I appreciated that every programming assignment in my CS1 class was open-ended and invited creativity. They channeled my interest in art toward experimenting with code, and I invested more time learning CS than I would have with bounded assignments. Researchers have found that creative work leads to a higher sense of identity/belonging [14], improves retention [9], and lowers cultural barriers to academic success [6]. For inclusion and enjoyment, I wish for my future students to share an engaging CS experience like mine. In my curriculum development role at UW, I designed creative assignments for CS1 students. I believe every CS class should expose students to CS applications and offer ways to engage in creative work, and I plan to offer them in my future teaching.

As a CS educator, I consistently take the initiative to bring in sociotechnical topics. I frame this for students by explaining that in order to really understand CS, the bits and bytes and lines of code are important, but it's at least as important to understand the social, economic, and political interactions surrounding technology. As a curriculum development TA, I advocated to my team for the adoption of and helped to write reflection questions for a homework assignment that gets students to critically reflect on race, gender, and AI bias[1] into the new CS2 curriculum. During my job as a section TA in CSE 143 at UW (a CS2 course), I regularly added discussions of race and gender, inclusion, access, and ethics in tech which were not included in the prescribed lesson plan. Some of my topics included censorship, racism, the potential values mismatch between an individual and their employer, and challenging students to use technology for social good.

My most memorable ethics micro-lesson happened when students were working on a programming assignment to simulate a game involving murderers and victims. I started with an anonymous poll to elicit student feelings. When over half reported discomfort with the assignment, I related this to how they might feel in a future job if asked to write code for an ethically ambiguous or questionable application, challenged them to think about their social responsibility in such a situation, and introduced them to the Tech Won't Build It movement. After teaching this spontaneously developed lesson, I presented it to the 40 other course staff, where it was well received by TAs and the professor (Brett Wortzman), who later hired me for a curriculum development role. My teaching was also well received by students. After one section, a student approached me to say that she hadn't thought about AI ethics before this class, but now it's something she gets excited about in CS.

Educating Colleagues about DEI Issues

Beyond teaching CS to students, I use my skills as an educator to teach my colleagues about DEI issues and bring them together in community to improve CS culture. In autumn 2020, I collaborated to compile, organize, and facilitate a series of six student-led workshops on anti-racism for members of my lab and the other robotics labs in my department. Topics included "What is Race," mass incarceration, police brutality, abolition, and racism in AI. 41 out of 55 lab members, including grad students, postdocs, and faculty, attended at least one workshop. I also developed and taught a workshop for members of my union on *calling in*, a strategy of responding to microaggressions by approaching the person responsible in a caring and supportive way, attended by 35 participants. And, I organized a community meeting for graduate students in my department to share their own labs' anti-racism strategies and learn strategies from members of other labs.

Summary

My teaching outlook prioritizes active learning and centers on culturally relevant computing and fostering a more supportive and less defensive climate. Through my time as a TA and an Instructor, I've gained experience engaging students with active learning and encouraging them to bring their whole selves to the classroom, and developed and taught embedded lessons in sociotechnical topics. My work as an educator interleaves with my research in CS identity and belonging. I continue to learn and develop as an educator by reading CS Education literature, studying cultural competence in the 3C program, and listening to my students' feedback.

[1] https://ethicalcs.github.io/#manipulators

References

[1] Alvarado, C., & Dodds, Z. (2010). Women in CS: An evaluation of three promising practices. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education - SIGCSE '10*, 57. https://doi.org/10.1145/1734263.1734281

[2] Barker, L. J., Garvin-Doxas, K., & Jackson, M. (2002). Defensive climate in the computer science classroom. *ACM SIGCSE Bulletin*, *34*(1), 43–47. https://doi.org/10.1145/563517.563354

[3] Barker, L. J., O'Neill, M., & Kazim, N. (2014). Framing classroom climate for student learning and retention in computer science. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 319–324. https://doi.org/10.1145/2538862.2538959

[4] Björkman, C. (2005). Crossing Boundaries, Focusing Foundations, Trying Translations: Feminist Technoscience Strategies in Computer Science. http://urn.kb.se/resolve?urn=urn:nbn:se:bth-00296

[5] Cech, E. A. (2014). Culture of Disengagement in Engineering Education? *Science, Technology, & Human Values*, *39*(1), 42–72. https://doi.org/10.1177/0162243913504305

[6] Eglash, R., Bennett, A., O'donnell, C., Jennings, S., & Cintorino, M. (2006). Culturally Situated Design Tools: Ethnocomputing from Field Site to Classroom. *American Anthropologist*, *108*(2), 347–362. https://doi.org/10.1525/aa.2006.108.2.347

[7] Freire, P. (1996). Pedagogy of the oppressed. *New York: Continuum*.

[8] Garvin-Doxas, K., & Barker, L. J. (2004). Communication in computer science classrooms: Understanding defensive climates as a means of creating supportive behaviors. *Journal on Educational Resources in Computing*, *4*(1), 2-es. https://doi.org/10.1145/1060071.1060073

[9] Haungs, M., Clark, C., Clements, J., & Janzen, D. (2012). Improving first-year success and retention through interest-based CS0 courses. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education - SIGCSE '12*, 589. https://doi.org/10.1145/2157136.2157307

[10] Lewis, C., Bruno, P., Raygoza, J., & Wang, J. (2019). Alignment of Goals and Perceptions of Computing Predicts Students' Sense of Belonging in Computing. *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 11–19. https://doi.org/10.1145/3291279.3339426

[11] Margolis, J. (2017). *Stuck in the Shallow End, updated edition: Education, Race, and Computing*. MIT Press.

[12] Schulte, C., & Knobelsdorf, M. (2007). Attitudes towards computer science-computing experiences as a starting point and barrier to computer science. *Proceedings of the Third International Workshop on Computing Education Research*, 27–38. https://doi.org/10.1145/1288580.1288585

[13] Scott, K. A., Sheridan, K. M., & Clark, K. (2015). Culturally responsive computing: A theory revisited. *Learning, Media and Technology*, *40*(4), 412–436. https://doi.org/10.1080/17439884.2014.924966

[14] Siva, S., Im, T., McKlin, T., Freeman, J., & Magerko, B. (2018). Using Music to Engage Students in an Introductory Undergraduate Programming Course for Non-Majors. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 975–980. https://doi.org/10.1145/3159450.3159468

[15] Waite, W. M., Jackson, M. H., Diwan, A., & Leonardi, P. M. (2004). Student culture vs group work in computer science. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, 12–16. https://doi.org/10.1145/971300.971308

[16] Wilcox, C., & Lionelle, A. (2018). Quantifying the Benefits of Prior Programming Experience in an Introductory Computer Science Course. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 80–85. https://doi.org/10.1145/3159450.3159480