

CS Education for the Socially-Just Worlds We Need

The Case for Justice-Centered Approaches to CS in Higher Education

Kevin Lin
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA, USA
kevinl@cs.uw.edu

ABSTRACT

Justice-centered approaches to equitable computer science (CS) education frame CS learning as a means for advancing peace, antiracism, and social justice rather than war, empire, and corporations. However, most research in justice-centered approaches in CS education focus on K–12 learning environments. In this position paper, we review justice-centered approaches to CS education, problematize the lack of justice-centered approaches to CS in higher education in particular, and describe a justice-centered approach for undergraduate Data Structures and Algorithms. Our approach emphasizes three components: (1) ethics: critiques the sociopolitical values of data structure and algorithm design as well as the underlying logics of dominant computing culture; (2) identity: draws on culturally responsive-sustaining pedagogies to emphasize student identity as rooted in resistance to the dominant computing culture; and (3) political vision: ensures the rightful presence of political struggles by reauthoring rights to frame CS learning as a force for social justice. Through a case study of this *Critical Comparative Data Structures and Algorithms* pedagogy, we argue that justice-centered approaches to higher CS education can help all computing students not only learn about the ethical implications of nominally technical concepts, but also develop greater respect for diverse epistemologies, cultures, and experiences surrounding computing that are essential to creating the socially-just worlds we need.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

critical pedagogy; cultural competence; culturally responsive pedagogy; data structures; ethics; identity; political vision; social justice

ACM Reference Format:
Kevin Lin. 2022. CS Education for the Socially-Just Worlds We Need: The Case for Justice-Centered Approaches to CS in Higher Education. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499291

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCSE 2022, March 3–5, 2022, Providence, RI, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9070-5/22/03…$15.00
https://doi.org/10.1145/3478431.3499291

1 INTRODUCTION

Dominant approaches to CS education frame "equity as inclusion" [4], emphasizing (for example) *capacity*, *access*, and *participation* in CS education for all students [12]. This assumes that all students want to learn CS the way we currently teach it. "However, are we satisfied with everyone learning to code, if the end game is to produce (admittedly more 'diverse') coders who will primarily work to ensure the continued profitability of capitalist start-ups and technology giants?" [6]. Are the more 'diverse' students we wish to recruit and retain satisfied with this vision for CS?

Beyond *capacity*, *access*, and *participation*, Fletcher and Warner describe student *experience* as the fourth component of the CAPE framework for assessing equity in CS education. While student performance measures such as course grades provide one means of assessing equity in outcomes, they argue that "truly equitable experiences must go beyond these simple outcome measures" in order to "create an environment where all students feel they belong, instruction is inclusive, and diverse perspectives are valued explicitly" [12]. Proponents of critical CS education [18, 31] argue that teaching computer science content knowledge alone is not enough "because they do not consider the perspective of the group being served" [15]. Dominant approaches to CS education marginalize diverse students' identities and political values by positioning CS education as a force for reproducing social inequity—more interested in maximizing efficiency and profit than doing good [4, 7, 28, 31, 32, 34, 36–38].

Informed by critical theory, justice-centered approaches to CS education complicate and challenge prior research efforts that "explain Black (and other minoritized) students' motivations about what to learn or not to learn as tied to their perceptions of what is 'geeky' (or inversely what is 'cool')," instead emphasizing "more complex dynamics underlying student resistance or interest" toward learning CS such as students' identities and political values [37]. Justice-centered approaches are currently emphasized in K–12 CS education through curricula such as Exploring Computer Science [31], but they are relatively absent in higher CS education research and practice. In this position paper, we argue for justice-centered approaches to CS in higher education. Section 2 reviews the literature on justice-centered approaches that center ethics, identity, and political vision. Section 3 makes the case for attention to justice-centered approaches in higher CS education. Section 4 problematizes dominant approaches to teaching undergraduate Data Structures and Algorithms (CS2). Section 5 proposes a novel, justice-centered approach for CS2 by applying critical pedagogy. Our justice-centered CS2 presents a case study of how a course that has traditionally marginalized all three of ethics, identity, and political vision can be reauthored to center justice.

2 JUSTICE-CENTERED CS EDUCATION

Vakil defines justice-centered approaches to CS education as attending to three features: the content of curriculum (centering ethics), the design of learning environments (centering identity), and the politics and purposes of CS education reform [36].

2.1 Ethics in the computing curriculum

Recent work in undergraduate computing ethics include standalone ethics courses [9, 30]; integrated ethics across the curriculum [5, 14]; and integrated ethics modules or lessons in courses such as machine learning [33], human-centered computing [35], and introductory CS [8, 10]. Fiesler et al. analyzed 115 syllabi from tech ethics courses and found that "many topics within tech ethics are high level and conceptual when it comes to the impact of technology on society—e.g., how human decisions are built into code, how technology can reproduce and augment existing social inequalities, how data is created by and directly impacts people, and how choices made at both the level of companies and in small bits of code combine to create large-scale social consequences" [11].

Critical approaches to tech ethics extend these ideas by emphasizing computing's political power to reshape social structures and hierarchies in the eye of the designer [9, 18, 24, 27, 32, 36, 37, 39]. Although Fiesler et al. argue that tech ethics "could be part of every computing course" [11], the inclusion of tech ethics is challenged by the hierarchy of knowledge in computing that prioritizes "technical" skills over "social" skills [27]. An analysis of 200 "technical" artificial intelligence and machine learning courses by Saltz et al. revealed only 12% of courses included some mention of ethics; and of these 12% of courses, ethics-related topics were relegated to the last two classes in the schedule and, in one course, left as a discussion topic only "if time allows" [13].

The exclusion of ethics in computing courses produce a kind of epistemic violence where dominant, "technical" ways of knowing CS are reinforced while marginalized, "social" ways of knowing CS are diminished. Malazita and Resetar observe this tension embedded in the emphasis on abstraction: the fundamental concept that some details or information are more relevant to a computational solution than others. The absence of ethics combined with the emphasis on abstraction defines CS education as not just apolitical (disregarding political or ethical values) but rather *anti-political*: computing is framed as a means of "solving" complex social problems by defining abstractions to manage complexity [23].

But critical educators argue that this framing is problematic: "quick ethics fixes, like modules largely developed for and within computer science, are not a sufficient intervention to actually teach CS students of how ethical challenges get resolved in real world contexts" [27]. Rather, more interdisciplinary efforts are needed to engage students with diverse stakeholders and collaborators [27, 39]. In *Design Justice*, Costanza-Chock calls on us to "seek more than 'freedom from bias.' For example, feminist and antiracist currents within science and technology studies have gone beyond a bias frame to unpack the ways that intersecting forms of oppression, including patriarchy, white supremacy, ableism, and capitalism, are constantly hard-coded into designed objects, platforms, and systems" [6]. Justice-centered approaches must teach students about the relationships between computing, power, and identity.

2.2 Identity in the learning environment

Students have not only social identities such as race, gender, or ethnicity, but also disciplinary identities that represent what they might be able to do with computer science [37]. Although data on social identity is often prominent in CS education research (e.g. racial demographics), empirical or theoretical data on disciplinary identity is often lacking [36] despite the importance of disciplinary identity in shaping students' sensemaking around the values of CS [37]. For marginalized students who experience firsthand the inequities in the social structures of schooling, their disciplinary identity is inextricably linked to their political identity and their commitment to issues of power and social justice [4, 36, 37]. Yet dominant approaches to CS education that frame learning as anti-political (where only the "technical" ideas count) emphasize that CS has no space for students' political identities.

Disciplinary identity is not only political, but also intersectional. Just as Costanza-Chock emphasizes the intersecting oppressions hard-coded in designed objects, students' disciplinary identities are also shaped by the "power dynamics that exemplifies racism, sexism, socioeconomic status, homophobia, ableism, xenophobia, etc." [29]. To create more inclusive and anti-oppressive learning environments, Washington argues for cultural competence in computing (3C): greater awareness, attitudes, knowledge, and skills toward working effectively in cross-cultural situations [40]. To realize this, Davis et al. define a framework for culturally responsive-sustaining CS pedagogy to ensure that "students' interests, identities and cultures are embraced and validated, students develop knowledge of computing content and its utility in the world, strong CS identities are developed, and students engage in larger socio-political critiques about technology's purpose, potential, and impact" [7].

2.3 Political vision for CS education

Given the importance of political identity toward students' disciplinary identity and their understanding of tech ethics, justice-centered approaches engage political identity by "collectively, clearly, and unequivocally articulat[ing] a political vision for CS learning anchored in principles of peace, antiracism, and justice" that "challenges the corporate technology sector on moral, epistemological, and political grounds" [36]. Justice-centered approaches seek to develop students' sociopolitical consciousness: "the recognition and desire to act upon societal inequities" through computing [22].

Integrating disciplinary identity and sociopolitical consciousness, Calabrese Barton and Tan describe a framework of *rightful presence* "towards *making present* the intersections of contemporary (in)justices, while orienting towards new, just social futures" [4]. Ko et al. argue for making injustices visible in CS education; rightful presence does so by foregrounding social and political narratives as a core pedagogical practice. CS education's commitment to justice in the classroom and beyond can be enacted through political activism in the classroom, such as "calls to action, practitioner reflections, legislative engagement, and direct action" [24]. This political activism moves beyond narrow ethical critiques or modules [27] and supports student agency to create the socially-just worlds we need [6]. By articulating a just political vision for CS education, ethics and identity "do not need to be 'included' in the curriculum" because they will already be the center of inquiry [36].

3 WHY JUSTICE IN HIGHER CS EDUCATION

Recent research calls attention to the limitations of dominant and epistemologically-exclusionary approaches to ethics in higher CS education [5, 23, 27]. These critiques reveal problems arising from the lack of attention to identity and political vision: CS education that ultimately produces students whose anti-political values perpetuate injustice [4, 6, 23]; educators who feel the need to avoid rather than address "political overtones" in ethics education [5]; internship experiences that marginalize and oppress Black women in computing [29]; and "well-intentioned" but culturally-blind CS faculty [23, 40] whose resistance to critical interventions and identity-centered instruction marginalizes students' political identities [25, 32, 36, 37, 39]. Ko et al. urge CS educators to make injustices visible "through the problems we focus on in our classrooms; through who we choose to teach; in how we shape students' career choices; and in how we conceptualize computing to journalists, social scientists, and society" because "[t]he world has critical questions about computing and it is time we started teaching more critical answers" [18]. But much of the research on justice-centered approaches to CS education focuses on K–12 learning environments [4, 22, 31, 32, 34, 36, 38]; much less research focuses on higher education. Ethics without identity or political vision is problematic.

We need justice-centered approaches to higher CS education to support justice-centered K–12 CS education; to help programs make progress toward diversity, equity, inclusion, and access (DEIA) goals and broaden participation in computing (BPC); and to develop student interest toward creating more socially-just worlds.

3.1 Support justice-centered K–12 CS education

Justice-centered approaches to higher CS education support the parallel and ongoing efforts in K–12 CS education across curricula such as Exploring Computer Science [31], books such as *Critically Conscious Computing: Methods for Secondary Education*,[1] and teacher education programs such as the University of Washington's STEP CS.[2] The current dominant approaches to higher CS education place critically-conscious secondary educators in a difficult (and potentially untenable) position. Given the need to *make present* contemporary injustices, critical secondary educators must make clear to their students that what awaits them in higher CS education is an epistemologically-exclusive and anti-political experience [23, 37]—one that often rejects ethics, marginalizes students' social and political identities, and envisions computing as a force for corporate profit. Higher CS education risks not only undoing efforts in K–12 CS education, but also subjecting undergraduate CS students to epistemic, material, and physical harm [18, 25, 29, 36].

By making present these injustices, students are less likely to put up with oppressive education systems. As justice-centered approaches become increasingly common, critically-conscious students unsatisfied with dominant approaches to higher CS education know to vote with their feet and enroll in programs that support their CS identity. Institutions unresponsive to justice-centered approaches risk backsliding on recruiting and retaining diverse students by failing to center marginalized students' identities and political values in computing.

[1] https://criticallyconsciouscomputing.org/
[2] https://criticalcsed.org/program/

3.2 Make progress toward DEIA goals and BPC

Justice-centered approaches to higher CS education can help undergraduate computing programs realize diversity, equity, inclusion, and access (DEIA) goals and broaden participation in computing (BPC). These goals are not only initiated within institutions of higher CS education, but also mandated by funding agencies for certain research grants: the National Science Foundation Computer and Information Science and Engineering directorate, for example, recently began requiring principal investigators of proposals submitted to selected programs to include a plan for broadening participation in computing at the time of award.[3] Fletcher and Warner describe "CAPE: A Framework for Assessing Equity throughout the Computer Science Education Ecosystem" that addresses *capacity for*, *access to*, *participation in*, and *experience of* equitable CS education [12]. Justice-centered approaches to CS education directly affects student experience in the way they "explicitly address issues of equity" and help "all students feel included and accepted" [12].

Attention to all three features of justice-centered approaches—ethics, identity, and political vision—can help higher CS education move "beyond equity as inclusion" [4] and address disparities in student participation and experience by ensuring the rightful presence of marginalized students' interests in learning CS [4, 32]. Justice-centered approaches can improve DEIA in programs and support BPC efforts by centering the values, experiences, and purposes of marginalized students through emphasis on computing's social responsibility [5, 18, 36], the disparate experiences of students with dominant versus marginalized identities [28, 29, 34, 40], and the political vision of computing toward realizing more socially-just futures [6, 18, 36].

3.3 Reauthor CS for more just futures

Despite differences in curricula between institutions, higher CS education shares common authors: the 1960s and 1970s academic computer scientists whose dominant, European scientific values [2] inspired a vision for computing that centered cognition and mathematics [16, 27, 36] to the exclusion and marginalization of identity and political vision [25, 32, 36, 37, 39]. Justice-centered approaches to higher CS education enable teachers to engage students in a process of *reauthoring rights* to value more diverse epistemologies rather than expecting assimilation to dominant epistemologies [4].

The implications of *reauthoring* extend beyond the classroom because they involve communication and learning not only with students, but also with teachers as they navigate tensions between values and purposes for CS education. Reauthoring considers students' cultural values, experiences, and ways of knowing as "integral to disciplinary learning" [4]. To address today's critical questions surrounding social computation, dominant approaches that emphasize an epistemological wall between the "technical" and the "social" are not equipped to provide critical answers [18]. Reauthoring rights through a process of political struggle and critical engagement—involving both students and teachers—can move computing as a discipline toward more critical perspectives that better appreciate and understand the sociopolitical implications of computation for all. In doing so, we develop student interest toward creating more socially-just worlds.

[3] https://www.nsf.gov/cise/bpc/

4 DATA STRUCTURES AND ALGORITHMS

In this section, we propose a justice-centered approach to teaching undergraduate Data Structures and Algorithms, or "CS2" at many institutions. While CS2 is a broad label representing the second course in computer science, many CS2 courses emphasize the design, implementation, and application of data structures and algorithms. Through conversation with experienced instructors, Porter et al. identified "two largely disjoint courses that are referred to in the CS education community as CS2"—Basic Data Structures and Advanced Data Structures [26]. They argue that, "At the end of a course on Basic Data Structures, students should be able to:

(1) Analyze runtime efficiency of algorithms related to data structure design.
(2) Select appropriate abstract data types for use in a given application.
(3) Compare data structure tradeoffs to select the appropriate implementation for an abstract data type.
(4) Design and modify data structures capable of insertion, deletion, search, and related operations.
(5) Trace through and predict the behavior of algorithms (including code) designed to implement data structure operations.
(6) Identify and remedy flaws in a data structure implementation that may cause its behavior to differ from the intended design.

Advanced Data Structures extend these learning objectives to emphasize "topics that rely on earlier data structures (e.g., balanced trees rely on BSTs, heaps rely on arrays)" as well as graph representations and graph algorithms [26].

4.1 Dominant approaches to teaching CS2

Dominant approaches to teaching CS2 center the development of cognitive skills toward relatively standardized data structures and algorithms content knowledge: CS learning is ultimately about increasing "individual comprehension of CS concepts and competent programming performance" [16]. Although there is much to critique about dominant approaches, this *cognitive framing* for CS learning does not necessarily imply that learning is decontextualized or irrelevant to students. The persistent popularity of the "Nifty Assignments" session at the annual ACM Technical Symposium on Computer Science Education (SIGCSE) is a testament to CS1 and CS2 educators' desire to design and adapt assignments that connect to students' diverse interests for learning CS.

However, without justice-centered approaches, CS2 instructors risk reproducing present-day oppressions. By teaching data structures as implementations for abstract data types, undergraduate computing programs emphasize the dominant programming practice that uses abstract data types to free "a programmer from concern about irrelevant details in his use of data abstractions" [21]—the kind of content knowledge reinforcing the *infrastructures of abstraction* that produce anti-political values [23]. By centering runtime or space complexity analysis [26] as the primary (and sometimes only) method for evaluating data structure and algorithm tradeoffs or design decisions, undergraduate computing programs emphasize efficiency as the primary metric for determining the quality of a computational solution. A more critical reading of the CS2 learning goals reveals the limits of cognitive approaches that frame algorithm design and implementation as "a means of realizing a specification or abstract data type without critically questioning the design of the abstraction" [6, 20, 23]. CS2's content knowledge emphasis on implementations over abstractions presents a unique challenge for critically-conscious and justice-centered approaches. Sociopolitical values are encoded in the design of abstractions [6, 20], but CS2's emphasis on implementations avoids discussion of the difficult social contexts and instead focuses on purely cognitive and mathematical analyses.

Justice-centered approaches are not necessarily mutually exclusive to the dominant cognitive approach [16]. In the context of CS2, undergraduate Data Structures and Algorithms content knowledge is uniquely powerful: completion of a "Data Structures and Algorithms" course offers a kind of limited certification to corporations that a student meets some of the qualifications for software engineering jobs. Completion of CS2 offers students easily-realizable power afforded by social mobility because data structures and algorithms content knowledge can prepare students for internships and full-time work in computing industries. The financial opportunities afforded by access to high-paying jobs in computing research and industry offers significant value to oppressed students who might otherwise have few options to generate wealth, sustain their communities, and escape poverty.

Where dominant approaches fail is in their inattention to the participation and experience of marginalized students in computing. Dominant approaches to CS2 risk exacerbating inequity in computing by creating and sustaining wealth for dominant students and corporations overwhelmingly benefiting from higher CS education's production of anti-political programmers [6, 23, 36]. After all, anti-political programmers are good for maximizing profit: they're easy to control and unlikely to resist because they see their design and engineering work as separate from the world. A justice-centered approach to Data Structures and Algorithms not only equips students with cognitive skills that unlock high-paying computing jobs, but also teaches students how they might navigate the tension around "selling out" their political commitments just to be "a part of a huge unfeeling oppressive corporation that makes you money sure, but never does something good" [37].

5 CRITICAL COMPARATIVE DATA STRUCTURES AND ALGORITHMS

Critical Comparative Data Structures and Algorithms (CCDSA) is a novel, justice-centered approach for teaching undergraduate Advanced Data Structures.

Ethics Critiques sociopolitical values of data structure and algorithm design and dominant computing epistemologies that approach social good without design justice.

Identity Centers students in culturally responsive-sustaining pedagogies to resist dominant computing culture and value Indigenous ways of living in nature.

Political vision Ensures the rightful presence of political struggles through reauthoring rights and emphasizes the political power of computing as a force for social justice in contrast to dominant narratives around corporate profit and hegemony.

The approach relies on *critical comparison* as the primary method of inquiry for centering ethics, identity, and political vision. The

method draws on traditions in critical pedagogy wherein structural critiques are foregrounded in students' education and injustices are made present. Kafai et al. coined a more researcher-centered view of the critical comparison approach: *theory dialogue*, which is designed to engage the diversity of cognitive, situated, and critical framings for CS education by emphasizing "understanding of key computational concepts, practices, and perspectives" (cognitive framing); "stress[ing] personal creative expression and social engagement" (situated framing); and respecting "the values, practices, and infrastructure underlying computation as part of a broader goal of education for justice" (critical framing) [16].

CCDSA does not strictly necessitate removing traditional, cognitive content from CS2. Instead, the critical comparison method enables learning of dominant knowledge to be framed from several diverse epistemological perspectives—the situated and critical framings often marginalized by dominant approaches. Although foregrounding the critical framing does not completely erase the tensions around "selling out," it offers students "a new and exciting possibility to *be* political while engaging in creating technology within the context of [a] CS class" [37].

5.1 Ethics via epistemological comparison

The epistemological values of dominant approaches result in "ethical and social interventions in CS education becom[ing] framed as valuable in application-centered classes, like data visualization or applied machine learning, but not in 'core' technical classes" like CS2 [23]. CCDSA proposes countering the dominant narrative by incorporating ethics as a type of algorithm analysis on equal epistemological footing as runtime or space complexity analysis.

CCDSA engages ethics in CS2 with an affordance analysis of data structures and algorithms [20]: a more critical algorithm analysis that draws on critical methods from science and technology studies, philosophy of technology, and human-computer interaction in order to evaluate the political consequences of data structures and algorithms in social contexts and applications [20]. Unlike dominant approaches to "algorithm analysis" that emphasize the internal implementation of data structures and algorithms [26], affordance analysis emphasizes the external interface of data structures and algorithms as they are applied in real-world applications. While dominant approaches to teaching abstraction risk producing "CS students as knowers who organize the world through excision" and abstraction [23], affordance analysis specifically problematizes the affordances of abstract data types (e.g. priority queues) as they are applied to problems (e.g. content moderation) by emphasizing how the design of an abstraction encodes affordances that can have political values [20]. While comparison between data structures and algorithms on the basis of efficiency are important, CCDSA emphasizes *critical comparison between abstractions on the basis of their sociopolitical implications*.

However, CCDSA is not satisfied with affordance analysis alone. Although affordance analysis makes space for sociotechnical critique of the design of computational solutions, "it is inherently limited to the algorithmic components of a sociotechnical system" and provides less instruction toward redressing harmful design values [20], engaging epistemological limits [23], or reauthoring the political purposes of CS [4, 37]. Costanza-Chock argues for practices that move beyond the "universalizing assumptions behind affordance theory" and instead attend to design justice that pushes students to "think more critically about software, technology, and design [...] in service of human liberation and ecological sustainability" [6]. Echoing the centrality of ethics, identity, and political vision in justice-centered CS education, design justice emphasizes the design practices, design narratives, design sites, and design pedagogies that create social conditions. Ethical analysis is not just limited to the harmful computer technology itself, but also all of the processes, infrastructures, and cultures that informed and enabled its design, implementation, and deployment [6].

CCDSA engages design justice to move beyond affordance analysis toward *critical comparison of computing epistemologies and design narratives*. Dominant approaches that center the software designers/engineers are compared to justice-centered approaches that position community members as the rightful designers. Design justice highlights the need to reauthor relationships between "designer" and "user" in order to give power back to people, rather than corporations or governments that often design technologies with little structural input from the people that they would most greatly affect. "Design justice is interested in telling stories that amplify, lift up, and make visible existing community-based design solutions, practices, and practitioners" [6]. Design justice often manifests in justice-centered approaches to K–12 CS education through projects enable students to design for their own communities, needs, and interests [7, 16, 32, 37]. In CCDSA, design justice counters the dominant approach that emphasizes software design as elite, private, and exclusionary; and instead centers community goals and values.

5.2 Identity via cultural comparison

CCDSA implements 5 of the 6 core components for culturally responsive-sustaining CS pedagogy [7] to center student identity. Although the framework emphasizes all three of ethics, identity, and political vision, in this section we'll focus on how it specifically enables *critical comparison* of cultural approaches to computation.

5.2.1 Acknowledge racism in CS and enact antiracist practices. The instructor leads fireside chats with teaching assistants (TAs) or students to explore their social identities (e.g. race, gender, ethnicity), their power and privileges, and how lived experiences have shaped their worldviews. By calling-out sites of inequity in the CS classroom and beyond, the teaching staff makes a commitment to dismantling structural oppression and decenter whiteness [7, 29, 34]. *Critical comparison* is made between dominant European scientific culture and Indigenous ways of living in nature [2] to engage the limits of European scientific epistemology and knowledge.[4]

5.2.2 Create inclusive and equitable classroom cultures. Antiracist practices are reinforced through course structure, assignments, and policies. Instruction and assessment applies universal design for learning to meet students' diverse means of engagement, representation, and action/expression [3] by valuing many reasons to learn data structures and algorithms. A specifications-based grading system enables creative assessments (e.g. self-reflections and student-submitted video explanations) without assigning a demeaning, trivial percentage weight. The creative assessments give

[4] https://courses.cs.washington.edu/courses/cse373/22wi/

students full opportunity to collaborate on most of the work in the course. Students are not only encouraged to collaborate but also taught how to recognize, confront, and dispel stereotypes and power imbalances that occur during teamwork [34].

5.2.3 Pedagogy and curriculum are rigorous, relevant, and encourage sociopolitical critiques. CCDSA recognizes the power afforded to students by the cognitive framing (because CS2 offers social mobility) but problematizes the applications of data structures and algorithms toward social problems embedded in complex historical and sociopolitical context. Dominant approaches emphasize programming implementation and asymptotic analysis, which admit little creativity and often leave students "subservient to the micro-demands of the autograder" [23]. *Critical comparison* engages students beyond programming and asymptotic analysis toward sociopolitical critiques of technology in relation to people, places, values, and hierarchies.[5]

5.2.4 Student voice, agency, and self-determination are prioritized in CS classrooms. Evidence-based pedagogies such as POGIL [19] offer a more structured approach to teamwork in order to ensure equity and teach students process skills such as communication and team management. POGIL classroom instruction is designed to amplify student voices rather than instructor voices. In particular, the instructor never presents ideas on their own. Instead, they listen to student teamwork on problems during class and amplify ideas generated during teamwork. Teaching assistants are specifically taught to resist the dominant pedagogical culture of teacher-centered instruction. Problem sets previously handed out during recitations are integrated into the main class and recitations are redesigned around self-reflection, team reflection, and whole-recitation reflection. Cognitive ideas, process skills, and emotional experiences are foregrounded through discussion with peers during recitations.

5.2.5 Family and community cultural assets are incorporated into CS classrooms. Community and cultural engagement beyond the scope of individual student experiences are not easily engaged by the dominant approaches to higher STEM education—particularly in large, research-oriented universities where impact is more largely determined by national prestige than service to the local community. Future work could examine the ways that higher CS education can partner undergraduate students with local community organizations in ways that are synergistic with the CS2 curriculum.

5.2.6 Diverse professionals and role models provide exposure to a range of CS/tech careers. The teaching staff are often the most visible people in the classroom, so effort is made to diversify instructors and TAs. TA recruitment and selection processes not only emphasize "clarity, technical proficiency, use of whiteboard, and responsiveness to student questions and needs" [17], but also cultural competency [7, 40] and readiness to engage with culturally responsive-sustaining CS pedagogy. Classroom and career advice decenters dominant narratives around software engineering jobs in order to highlight opportunities in academic research inside and outside of CS, professional as well non-profit or volunteer opportunities to teach CS in local communities, and roles for computing in social change beyond critique [1].

[5]https://github.com/kevinlin1/huskymaps

5.3 Political vision via narrative comparison

The *critical comparative* approach to ethics and identity orients CCDSA toward a political vision of CS for social justice: design justice [6] and culturally responsive-sustaining pedagogy [7] go a long way to making space for students to engage with sociopolitical values in the computing classroom [32]. Implicit in all of this is a vision of CS for social justice. CCDSA expands on this by explicitly articulating a political vision of CS for social justice through *critical comparison* of narratives and purposes for learning CS2.

From day one of class, teachers emphasize diverse end goals for learning data structures and algorithms: not only the dominant narrative around high-paying jobs in the tech industry, but also marginalized narratives around how data structures and algorithms content knowledge can support research, non-profit, and activist work in revealing social inequities or enabling solutions for local communities. By teaching identity via cultural comparison, teachers break down normative barriers between students and teachers that limit emotional communication—the idea of being distanced and neutral STEM learners is rationalized by the dominant European scientific culture. The establishment of this mutual trust and common understanding between students and teachers enables critique of the "fraught histories" [4] of computing by connecting a thread between the history of CS and present-day inequities. One such thread spans (1) the invention of data structures and algorithms for expressly military purposes, e.g. dynamic programming was invented as a political cover for mathematics research given the uncertainty around post-war US military research funding; into (2) the authoring of computer science as an academic discipline defined by post-war mathematics researchers seeking a new brand for their cognitive mathematics work; finally leading to the (3) the marginalization of many Black women computers who were pushed out of programming once computing became perceived as a profitable field for white men. By engaging this dialogue with students through collaborative teamwork and creative assignments, CCDSA develops students' political vision through understanding data structures and algorithms content knowledge not as a purely "technical" and apolitical knowledge, but rather as a knowledge situated in the political vision of the authors of the field.

6 CONCLUSION

The articulation of a political vision for CS centered in social justice requires work not just in a single CS2 course but rather across all of CS education. This is an ambitious project, one that will require us to reprogram the defaults of CS education. But it's also a project that today's CS educators do not need to solve alone: though today's students may be newcomers to CS, they are also precisely the people who will be the authors of our future worlds. When we center ethics, identity, and political vision in our culturally responsive-sustaining pedagogy, we treat students as collaborators and co-creators of education that works for them. Rather than seek assimilation of diverse students into the dominant CS culture, justice-centered approaches ensure the rightful presence of students' political struggles for co-authorship in designing and imagining a new, more just CS education. To create more socially-just worlds, we need to do more than just teach students computer ethics. We need justice-centered approaches to CS in higher education.

REFERENCES

[1] Rediet Abebe, Solon Barocas, Jon Kleinberg, Karen Levy, Manish Raghavan, and David G. Robinson. 2020. Roles for Computing in Social Change. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. ACM, New York, NY, USA. https://doi.org/10.1145/3351095.3372871

[2] Glen S. Aikenhead and Masakata Ogawa. 2007. Indigenous knowledge and science revisited. *Cultural Studies of Science Education* 2, 3 (9 2007), 539–620. https://doi.org/10.1007/s11422-007-9067-8

[3] Sheryl Burgstahler. 2011. Universal Design: Implications for Computing Education. *ACM Transactions on Computing Education* 11, 3 (10 2011). https://doi.org/10.1145/2037276.2037283

[4] Angela Calabrese Barton and Edna Tan. 2020. Beyond Equity as Inclusion: A Framework of "Rightful Presence" for Guiding Justice-Oriented Studies in Teaching and Learning. *Educational Researcher* 49, 6 (8 2020), 433–440. https://doi.org/10.3102/0013189X20927363

[5] Lena Cohen, Heila Precel, Harold Triedman, and Kathi Fisler. 2021. A New Model for Weaving Responsible Computing Into Courses Across the CS Curriculum. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 858–864. https://doi.org/10.1145/3408877.3432456

[6] Sasha Costanza-Chock. 2020. *Design Justice: Community-Led Practices to Build the Worlds We Need*. The MIT Press, Cambridge, MA, USA.

[7] Kalisha Davis, Shana V White, Becton-Consuegra Dinah, and Allison Scott. 2021. *Culturally Responsive-Sustaining Computer Science Education: A Framework*. Technical Report. Kapor Center. https://www.kaporcenter.org/equitablecs/

[8] Stacy A. Doore, Casey Fiesler, Michael S. Kirkpatrick, Evan Peck, and Mehran Sahami. 2020. Assignments that Blend Ethics and Technology. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 475–476. https://doi.org/10.1145/3328778.3366994

[9] Rodrigo Ferreira and Moshe Y. Vardi. 2021. Deep Tech Ethics: An Approach to Teaching Social Justice in Computer Science. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 1041–1047. https://doi.org/10.1145/3408877.3432449

[10] Casey Fiesler, Mikhaila Friske, Natalie Garrett, Felix Muzny, Jessie J. Smith, and Jason Zietz. 2021. Integrating Ethics into Introductory Programming Classes. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 1027–1033. https://doi.org/10.1145/3408877.3432510

[11] Casey Fiesler, Natalie Garrett, and Nathan Beard. 2020. What Do We Teach When We Teach Tech Ethics?. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 289–295. https://doi.org/10.1145/3328778.3366825

[12] Carol L. Fletcher and Jayce R. Warner. 2021. CAPE. *Commun. ACM* 64, 2 (1 2021), 23–25. https://doi.org/10.1145/3442373

[13] Natalie Garrett, Nathan Beard, and Casey Fiesler. 2020. More Than "If Time Allows": The Role of Ethics in AI Education. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. ACM, New York, NY, USA. https://doi.org/10.1145/3375627.3375868

[14] Barbara J. Grosz, David Gray Grant, Kate Vredenburgh, Jeff Behrends, Lily Hu, Alison Simmons, and Jim Waldo. 2019. Embedded EthiCS. *Commun. ACM* 62, 8 (7 2019), 54–61. https://doi.org/10.1145/3330794

[15] Aleata Hubbard Cheuoua. 2021. Confronting Inequities in Computer Science Education: A Case for Critical Theory. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA. https://doi.org/10.1145/3408877.3432453

[16] Yasmin Kafai, Chris Proctor, and Debora Lui. 2019. From Theory Bias to Theory Dialogue. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ACM, New York, NY, USA, 101–109. https://doi.org/10.1145/3291279.3339400

[17] Amir Kamil, James Juett, and Andrew DeOrio. 2019. Gender-balanced TAs from an Unbalanced Student Body. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA. https://doi.org/10.1145/3287324.3287404

[18] Amy J. Ko, Alannah Oleson, Mara Kirdani-Ryan, Yim Register, Benjamin Xie, Mina Tari, Matthew Davidson, Stefania Druga, and Dastyni Loksa. 2020. It is time for more critical CS education. *Commun. ACM* 63, 11 (10 2020), 31–33. https://doi.org/10.1145/3424000

[19] Clifton Kussmaul. 2012. Process Oriented Guided Inquiry Learning (POGIL) for Computer Science. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. ACM Press, New York, New York, USA. https://doi.org/10.1145/2157136.2157246

[20] Kevin Lin. 2021. Do Abstractions Have Politics? Toward a More Critical Algorithm Analysis. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. IEEE, 1–5. https://doi.org/10.1109/RESPECT51740.2021.9620635

[21] Barbara Liskov and Stephen Zilles. 1974. Programming with abstract data types. *ACM SIGPLAN Notices* 9, 4 (4 1974), 50–59. https://doi.org/10.1145/942572.807045

[22] Tia C. Madkins and Maxine McKinney de Royston. 2019. Illuminating political clarity in culturally relevant science instruction. *Science Education* 103, 6 (11 2019), 1319–1346. https://doi.org/10.1002/sce.21542

[23] James W. Malazita and Korryn Resetar. 2019. Infrastructures of abstraction: how computer science education produces anti-political subjects. *Digital Creativity* 30, 4 (10 2019), 300–312. https://doi.org/10.1080/14626268.2019.1682616

[24] Jared Moore. 2020. Towards a more representative politics in the ethics of computer science. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. ACM, New York, NY, USA, 414–424. https://doi.org/10.1145/3351095.3372854

[25] Thomas M. Philip and Pratim Sengupta. 2021. Theories of learning as theories of society: A contrapuntal approach to expanding disciplinary authenticity in computing. *Journal of the Learning Sciences* 30, 2 (3 2021), 330–349. https://doi.org/10.1080/10508406.2020.1828089

[26] Leo Porter, Daniel Zingaro, Cynthia Lee, Cynthia Taylor, Kevin C. Webb, and Michael Clancy. 2018. Developing Course-Level Learning Goals for Basic Data Structures in CS2. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 858–863. https://doi.org/10.1145/3159450.3159457

[27] Inioluwa Deborah Raji, Morgan Klaus Scheuerman, and Razvan Amironesei. 2021. You Can't Sit With Us. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. ACM, New York, NY, USA, 515–525. https://doi.org/10.1145/3442188.3445914

[28] Yolanda A. Rankin and Jakita O. Thomas. 2020. The Intersectional Experiences of Black Women in Computing. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 199–205. https://doi.org/10.1145/3328778.3366873

[29] Yolanda A. Rankin, Jakita O. Thomas, and Sheena Erete. 2021. Real Talk: Saturated Sites of Violence in CS Education. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, Vol. 12. ACM, New York, NY, USA, 802–808. https://doi.org/10.1145/3408877.3432432

[30] Rob Reich, Mehran Sahami, Jeremy M. Weinstein, and Hilary Cohen. 2020. Teaching Computer Ethics. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 296–302. https://doi.org/10.1145/3328778.3366951

[31] Jean J. Ryoo. 2019. Pedagogy that Supports Computer Science for All. *ACM Transactions on Computing Education* 19, 4 (11 2019), 1–23. https://doi.org/10.1145/3322210

[32] Jean J. Ryoo, Tiera Tanksley, Cynthia Estrada, and Jane Margolis. 2020. Take space, make space: how students use computer science to disrupt and resist marginalization in schools. *Computer Science Education* 30, 3 (7 2020), 337–361. https://doi.org/10.1080/08993408.2020.1805284

[33] Jeffrey Saltz, Michael Skirpan, Casey Fiesler, Micha Gorelick, Tom Yeh, Robert Heckman, Neil Dewar, and Nathan Beard. 2019. Integrating Ethics within Machine Learning Courses. *ACM Transactions on Computing Education* 19, 4 (11 2019), 1–26. https://doi.org/10.1145/3341164

[34] Niral Shah, Julie A. Christensen, Nickolaus A. Ortiz, Ai-Khanh Nguyen, Sunghwan Byun, David Stroupe, and Daniel L. Reinholz. 2020. Racial hierarchy and masculine space: Participatory in/equity in computational physics classrooms. *Computer Science Education* 30, 3 (7 2020), 254–278. https://doi.org/10.1080/08993408.2020.1805285

[35] Michael Skirpan, Nathan Beard, Srinjita Bhaduri, Casey Fiesler, and Tom Yeh. 2018. Ethics Education in Context. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 940–945. https://doi.org/10.1145/3159450.3159573

[36] Sepehr Vakil. 2018. Ethics, Identity, and Political Vision: Toward a Justice-Centered Approach to Equity in Computer Science Education. *Harvard Educational Review* 88, 1 (3 2018), 26–52. https://doi.org/10.17763/1943-5045-88.1.26

[37] Sepehr Vakil. 2020. "I've Always Been Scared That Someday I'm Going to Sell Out": Exploring the relationship between Political Identity and Learning in Computer Science Education. *Cognition and Instruction* 38, 2 (4 2020), 87–115. https://doi.org/10.1080/07370008.2020.1730374

[38] Sepehr Vakil and Maxine McKinney de Royston. 2019. Exploring Politicized Trust in a Racially Diverse Computer Science Classroom. *Race Ethnicity and Education* 22, 4 (7 2019), 545–567. https://doi.org/10.1080/13613324.2019.1592846

[39] Sepehr Vakil and Jennifer Higgs. 2019. It's about power. *Commun. ACM* 62, 3 (2 2019), 31–33. https://doi.org/10.1145/3306617

[40] Alicia Nicki Washington. 2020. When Twice as Good Isn't Enough. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 213–219. https://doi.org/10.1145/3328778.3366792