Teaching Statement

Teaching computer science requires both disciplinary stability and pedagogical adaptation. The field's core ideas (abstraction, mathematical reasoning, logic, algorithmic thinking, design, and validation) remain essential. At the same time, the contexts in which students learn and practice computing are changing rapidly. My teaching philosophy is grounded in the belief that effective instruction must respond to these changes without abandoning the intellectual foundations that define computer science as a discipline. In particular, the emergence of generative artificial intelligence has necessitated a reconsideration of how students learn programming, data structures, algorithms, and computational problem-solving. My goal as an instructor is to help students use new tools productively while developing the judgment, independence, and conceptual understanding needed to evaluate and design programmable solutions.

This philosophy has shaped my teaching in foundational undergraduate courses, especially CS251 - Data Structures and Algorithms. Since joining Purdue, I have taught CS251 across multiple offerings and have treated the course as a central site for helping students move from programming practice toward deeper computational reasoning. In my approach, data structures are not merely implementation exercises; they are mathematical and conceptual tools for organizing information, analyzing tradeoffs, and designing efficient solutions. I have therefore emphasized formal analysis, abstraction, correctness, and the conditions under which one data structure or algorithm is appropriate over another. This emphasis is especially important in today's technological environment, where students can often generate code quickly but may not be able to justify, debug, adapt, or validate it without a strong conceptual foundation.

My teaching also reflects the view that assessment should support learning rather than simply measure performance at a few high-stakes moments. In CS251, I have revised assessment structures incrementally in response to the course's demands and students' experiences. Beginning in Summer 2025, I increased the number of examinations while narrowing their scope to encourage more continuous study and reduce the cognitive burden of infrequent, comprehensive exams. I later introduced shorter in-class checkpoints (equivalent to midterms) administered approximately every two weeks. These checkpoints provide regular feedback, help students identify misunderstandings earlier, and distribute evaluation more evenly across the semester. This structure aligns with my broader goal of helping students develop durable habits of reasoning, preparation, and self-correction.

Generative AI has become an important part of my teaching, but not as a replacement for student thinking. From the moment tools like ChatGPT became publicly available, I

recognized that computer science education could not respond only with prohibition. Students would encounter these tools in their coursework, internships, and careers. The educational task, therefore, is to teach students to engage with AI critically and responsibly while preserving the discipline's core cognitive work. In my courses, I have worked to distinguish between using AI as a tool and relying on it as a crutch. Students should learn to ask better questions, evaluate generated responses, identify corner cases, validate outputs, and explain the reasoning behind a solution. They should also understand when AI-generated code is incomplete, misleading, inefficient, or inconsistent with a problem's requirements.

This perspective informed my work with the AI-Lab framework, which integrates generative AI into programming and algorithms courses through scaffolded, bounded applications. Rather than encouraging students to ask AI for final answers, the framework emphasizes uses such as receiving alternative explanations, exploring logic and corner cases, and improving student-authored code after the central design work is complete. In CS251 and CS253, this work has enabled me to study student AI use while adjusting course policies to reflect reality. For take-home work, I have permitted AI use under explicit conditions and placed greater weight on in-person assessments that measure individual mastery. This approach acknowledges that AI assistance is now part of the learning environment while maintaining the integrity of the course’s learning outcomes.

I have also created new instructional spaces in response to AI’s influence on computing practice. CS290 - Vibe Coding was designed to help students deliberately explore AI-mediated software development rather than do so informally. In this course, students complete programming challenges under the constraint that they must rely on AI systems to generate and revise code. This constraint forces students to confront the practical challenges of prompt-based development: specifying requirements precisely, evaluating generated code, diagnosing failures, and iterating toward correctness. The course is not intended to suggest that manual programming is obsolete. Instead, it helps students see that even when code is generated by AI, human expertise remains necessary for design, validation, evaluation, and responsible decision-making.

My teaching philosophy also extends to curriculum development. The creation of CS253 - Data Structures and Algorithms for Data Science and Artificial Intelligence addressed a curricular mismatch I observed among Data Science students taking CS251. These students often had different preparation and professional goals than Computer Science majors, particularly in their familiarity with dynamic memory, pointer-based structures, and data-oriented applications. CS253 enabled the department to better serve Data Science and Artificial Intelligence students while preserving the rigor of data structures and

algorithms. I continue to view this course as an opportunity to connect algorithmic thinking with data-intensive and AI-oriented problems, including probabilistic data structures, large-scale search, sparse representations, and applications that reflect the computational needs of those majors.

Beyond individual courses, I view teaching as a shared instructional ecosystem. In large-enrollment courses, undergraduate and graduate teaching assistants play a central role in students' learning experiences. I therefore treat TA mentoring as part of my teaching responsibility. I give TAs meaningful instructional roles in office hours, discussion forums, assessment development, and student support, and I meet with them regularly to discuss student difficulties and instructional strategies. A recurring focus of this mentoring is helping TAs support students without removing the productive struggle necessary for learning. This requires judgment: students need guidance, but they also need to practice reasoning through uncertainty. My work with AI-simulated student platforms and structured GTA preparation further reflects this commitment to developing instructors who can combine technical accuracy with empathy, scaffolding, and professional communication.

Undergraduate research mentoring is another important extension of my teaching. I have involved students in projects on GenAI in education, algorithmic runtime analysis, plagiarism detection, geometric modeling, and instructional tools. These experiences enable students to engage with open-ended problems, collaborate on technical development, communicate results, and, in several cases, contribute to scholarly publications or research presentations. I view these opportunities as a way to move students beyond course completion and into participation in the practices of the discipline.

Across my teaching, curriculum development, and mentoring, my central commitment is to prepare students for a version of computer science that is changing yet remains grounded in rigorous thought. AI tools will continue to improve, and computing practice will continue to evolve. For that reason, students need more than familiarity with current tools. They need the ability to design, analyze, validate, communicate, and reflect. My teaching aims to help students develop the capacity to use emerging technologies with confidence, skepticism, and responsibility while retaining the computational thinking that defines computer science as a field.