Teaching Statement

Djellel Difallah (djellel@nyu.edu)

An important shift in classroom dynamics occurs when a question is asked, transforming students from passive listeners into active participants. This moment allows me to clarify concepts, connect seemingly disparate ideas, identify gaps in students' understanding, and observe new insights taking shape. I strive to foster an environment where asking questions is not just encouraged but integral to the learning process. To spark these moments in my classroom, I regularly present my students with vivid visual demonstrations and tangible examples drawn from relatable real-world scenarios and from my own experience. Outside the classroom, I support my students through a dedicated virtual discussion board (Slack Workspace), offering timely responses to their questions, expanding on some concepts, and engaging in discussions on open-ended problems and optional exercises. My assignments are intentionally designed to be challenging yet fun, encouraging students to think creatively while enjoying the learning process and embracing the challenge. I often hear from them that they feel compelled to complete their homework because it's engaging and, in doing so, they must stretch their abilities by improving their programming skills, finding synergies with their peers, or seeking help from me or by using online resources.

Ultimately, my mission as an educator is to ignite a spark of genuine curiosity so students not only grasp core concepts in a memorable way but also develop a sense of self-motivation, independence, and critical thinking that will serve them well beyond the classroom and throughout their careers. In the following sections, I will illustrate how I apply my teaching methodology in courses I have taught at NYU Abu Dhabi. These include two required computer science courses (Data Structures and Discrete Mathematics) and an elective (Big Data).

Inquiry-Driven Environment

Drawing inspiration from established active learning frameworks [2], I position students as active collaborators during lectures. By fostering an environment that encourages inquiry and questioning, students feel more empowered to engage with the material and less hesitant to ask questions. When students generate their own questions and hypotheses, it progressively deepens their conceptual understanding.

Student-Led Problem Solving In my Data Structures class, I introduce algorithms progressively, guiding students from simple scenarios they can easily reason about to more realistic cases. For example, when discussing queues (such as a line at a supermarket), we initially explore solutions assuming an *infinite* capacity table for storage, and students readily propose multiple approaches. As we transition to more realistic scenarios using a *fixed* capacity table, their initial propositions often become unusable. Rather than immediately offering answers, I prompt students to revisit and adapt their initial approaches. Through guided discussion and incremental hints, they discover solutions independently (such as looping around the table to reclaim unused space).

Through interactions like this, my inquiry-based approach can help demystify complex algorithms and enable students to internalize the general algorithmic ideas more effectively. I observe students actively identifying edge cases to challenge the algorithm – asking, for instance, *"What if the queue reaches its capacity?"* – which demonstrates that they have already grasped the algorithm. Student feedback often includes comments like: *"The professor explains concepts in an amazing way,"* although my role is primarily about guiding them toward discovering solutions themselves.

Class Participation A challenge I encounter when teaching introductory courses is that students come in with varying levels of mathematical background and programming skills. In the Discrete Mathematics course, for example, I employ tiered questioning strategies during lectures and use regular in-class polls to gauge understanding. I start with accessible, foundational questions to encourage broad participation, then gradually introduce more complex problems. These polls help me assess comprehension of the class in real-time but also prompts participation from students who are hesitant to speak up. I make an effort to constructively acknowledge all responses, even incorrect ones, by giving them a positive spin to build student confidence.

This approach has proven effective, as I observed an increase in participation across students of all ability levels when using it. I've also received personal notes from students specifically appreciating the poll-based participation format and how it helps them comprehend the material and not fall behind. Student feedback consistently highlights the interactive nature of my teaching, with comments such as, *"The class was very interactive and the professor explained concepts several times to make sure everyone was catching the concept."*

Engaging Learning Experience

Maintaining students' attention throughout an entire class session can be challenging and requires diverse teaching methods. During lectures, I insert checkpoints that allow students to reset their attention. These moments provide opportunities for students to catch up if they lose focus, as I offer a brief recap, restating the big picture and our learning goals. Additionally, when discussing complex problems, I incorporate engaging tools such as interactive visualizations and in-class activities.

Interactive Visualizations To create an immersive learning experience that engages students in a given topic, I use interactive visualizations to emphasize the importance of the underlying concepts we are studying. For example, in my Big Data course, the significance of graph partitioning algorithms becomes clear as students watch a vivid animation demonstrating how reorganizing the rows and columns of a matrix can dramatically impact its internal data organization. This visualization not only helps students grasp the algorithm but also leads them to independently propose the next logical step: split the array based on the dense regions.

Similarly, in my Data Structures course, I incorporate visualizations in nearly every session that involves tree-based algorithms, from exploring the self-balancing properties of red-black trees to constructing a heap in linear time. Students often note how these interactive tools enhance their understanding by allowing them to test a virtually infinite number of scenarios, reinforcing their grasp of key concepts through direct experimentation and immediate feedback.

Interactive Problem-Solving While visualizations can be highly engaging, they may have limitations for some learners. To broaden my teaching methods, I integrate interactive problem-solving in select instances where abstract concepts benefit from direct experience. In my Discrete Mathematics course, the Monty Hall problem [3] offers a perfect opportunity to make the concept of conditional probability intuitive. Instead of directly explaining the solution using formulas, I transform the classroom into a game show, where students face the classic dilemma: after choosing one of three doors, should they switch their selection when the host reveals that another door is empty? Using a simulation interface, students participate in multiple rounds and collect data on both switching and non-switching strategies, discovering that switching doors doubles their chances of winning.

This class exercise accomplishes multiple learning goals: it provides concrete evidence supporting theoretical results, sparks productive discussions as students contemplate the outcomes, and motivates deeper understanding of the underlying mathematical principles. I've observed that in subsequent probability lessons, students approach me with thoughtful questions rather than accepting formulas.

Group-based activities In my Big Data course, understanding the fundamentals of distributed systems and algorithms represents a significant paradigm shift for undergraduate students [1]. To facilitate this transition, I introduce carefully designed group-based activities that illustrate these concepts intuitively.

In one activity, students collaboratively approximate π using a Monte Carlo simulation method. Each student repeatedly generates sampled data points on their own computer, then submits them via a Google Form. These data points are collected in real time on my computer, where a dynamic visualization displays the running approximation. The students witness firsthand how distributed computation yields increasingly accurate approximations as more data is gathered. This activity demonstrates the efficiency gains achievable through parallel processing when paired with suitable algorithms.

In another activity, I divide students into small groups of five and assign each group a stack of playing cards to count as quickly as possible, under communication constraints that mirror real-world distributed systems. Through direct experience, students more effectively comprehend the challenges inherent in distributed environments and the origin of key principles: data locality (each student counting their own cards), parallel processing (simultaneous counting), and message-passing (limited communication). When we later formally introduce distributed computation frameworks, such as MapReduce, students can naturally relate each theoretical component to their earlier hands-on experience, making the transition from sequential to distributed computing paradigms more intuitive.

Students have responded positively to these activities, with one student remarking in their feedback: *"I loved his way of teaching and especially the interactive games."* Beyond engagement, these activities serve as anchors for follow-up discussions, as students frequently refer to them when reasoning through distributed computing scenarios.

Experiential Learning

Computer science is a rapidly evolving field, and many tools that students learn today may become obsolete within a few years. Recognizing this reality, I strive to design learning experiences that cultivate self-directed learning skills alongside theoretical knowledge, while also incorporating real-world use cases in the classroom.

Hands-on Projects To develop these skills, I structure my programming assignments into multiple manageable mini-projects throughout the course, designed to be both engaging and motivating. In one of my Big Data course projects, students build a *mini-Google*, a search engine capable of indexing millions of Wikipedia articles with sub-second query response times. Initially, students often express apprehension about the project's complexity; however, because it has a clear, unambiguous goal, they become curious and motivated. During support sessions, students decompose the problem into manageable components and identify which class concepts they can apply. The remaining work primarily involves technical implementation and researching external libraries—requiring them to read documentation independently. As one student reflected in the feedback, *"I loved the assignments, especially the inverted index, it was super fun and interesting."*

Real-world Experience I complement these hands-on challenges with contextualization drawn from my industry experience. For instance, I show students the system design behind Wikipedia and revisit in practice how several components discussed in class contribute to serving over 200 billion page views per day. However, the notion of *scale* remains abstract for most students. To address this, and with the generous support of NYU Abu Dhabi's Center for Research Computing, I have taken students on tours to observe our High-Performance Computing facility in operation. Seeing thousands of servers processing data, observing the complexity of the physical network, and experiencing the heat necessitating industrial cooling systems transforms their understanding of distributed computing from theoretical to tangible. This experiential learning sparks followup discussions with our guest speaker, Dr. Mouataz Albarwani, about current computing challenges, including the global competition for computational power and the sustainability concerns associated with it.

Through this combination of ambitious projects and real-world experiences, students develop a stronger grounding in the technologies we discuss in class and are better prepared to enter professional life. This aspect has been the highlight of my Big Data course in terms of student feedback, with one student commenting: *"The course was beautifully designed and went over so many important concepts directly from the industry."*

Mentorship

My teaching philosophy extends naturally beyond the classroom as I mentor students each year in their capstone projects, and I have supervised two directed studies on advanced computer science topics. In these small-group settings, I continue promoting an inquiry-driven environment where students develop the confidence and independence to explore new ideas. By providing selected readings, holding research group discussions, encouraging frequent hands-on experimentation, establishing structured milestones, and offering personalized support, I guide students to adopt methodologies that will help them solve complex, real-world problems. My goal is not just to teach technical concepts, but to instill a mindset of first-principle thinking, equipping students with the resilience needed in our rapidly evolving field.

References

[1] Nathaniel Kremer-Herman. 2022. Challenges and Triumphs Teaching Distributed Computing Topics at a Small Liberal Arts College. In *EduHPC@SC*. IEEE, 26–33.

[2] Eric Mazur. 2009. Confessions of a Converted Lecturer. Lecture/Talk. Available online at https://www.youtube.com/watch?v=WwslBPj8GgI.

[3] Allison Parshall. 2024. Why Almost Everyone Gets the Monty Hall Probability Puzzle Wrong. *Scientific American* (2024). https://www.scientificamerican.com/article/why-almost-everyone-gets-the-monty-hall-probability-puzzle-wrong/