Teaching Statement

Suyash Gupta

The Indian poet Kabir says, "My teacher and my god are standing in front of me, to whom should I bow? I bow to you my teacher as it is you who taught me about the god." This simple yet beautiful quote highlights the importance of a good teacher in the life of a student. A teacher not only plays the role of an educator but also adorns the robes of a mentor, idol, confidant, and friend. A teacher nurtures a student in the same manner as a gardener nurtures its plant. As a prospective candidate in the academic job market, I fully recognize this responsibility and I view the role of a computer science professor as a researcher, teacher, mentor, and contributor to the department.

I view computer science as a layered discipline with many facets that unravel new mysteries for students to learn and discover. Thus, I have devised the 3Ps model: *Participate*, *Practice*, and *Propose*, which I have been applying since my master's on every student with whom I have collaborated, and it has yielded fruitful results. Through the first P (*participate*), I encourage participation during class and group meetings by asking questions This can help students to be comfortable in their surroundings and visualize a class/meeting as a safe space. The second P (*practice*) encourages students to practice the concepts they learned in class or implement the ideas discussed during a meeting. This will propel the students up from the ground and eliminate their fear of starting something, for example, coding or debugging an assignment. The final P (*propose*) advocates free thinking. It aims to empower students with the confidence to propose new research problems and solutions.

Teaching. Throughout my academic journey, I have been fortunate enough to supervise students in various courses. I served as a teaching assistant at IIT Madras (for CS1100 - Introduction to Programming, CS3310 - Language Translators, CS6848 - Principles of Programming Languages) and Purdue University (CS565 - Programming Languages) and as a teaching fellow at UC Davis (ECS189F - Introduction to Distributed Ledgers). Additionally, during my postdoc at UC Berkeley, I taught Introduction to Databases course to underprivileged students associated with Mentors without Borders. During all of these courses, I aimed at teaching students through my 3Ps model.

1. Participate. For me, the first step to teaching students any CS course is to ensure that they are comfortable in the class and are eager to ask questions. As a result, I devote the first five minutes of each session to talk about non-academic topics. My opening statement is often a joke that aims to welcome students. I have found "dad jokes" as my ugly strength. For instance, in CS1100, I often made statements like "Let us see (C), or To C or To not C", and in ECS189F, I often called a projector failure or equipment failure as a Byzantine failure. Once the stage is set, although I always have slides at hand, I go for *blackboard style teaching*; I find students more interactive when I write illustrations and analysis on board. For instance, in one of the guest lectures (ECS265: Distributed Database Systems, Fall 2019), I introduced students to state machine replication, safety and liveness guarantees, the need for $3f + 1$ replicas in a Byzantine setting, and the complete PBFT protocol on a blackboard. This motivated students to bombard me with different types of failures and attacks. To further encourage participation, I ask students to reach out to me via email, slack and schedule one-to-one meetings with me beyond office hours, which has proved useful for several undergraduate students with time conflicts. For instance, CS3310 students used to often request meetings between 5-7pm.

2. Practice. Just encouraging participation is insufficient to help students understand the finer details of a subject. Through practice, students learn to appreciate the expressed ideas and concepts. My teaching strategy envisions practice as a two-step process: (i) give examples connected to real-world software, and (ii) pose brain simulating questions. For example, for a majority of CS1100 students, it is their first shot at coding, and these students are often surprised to know that games, such as Counter-Strike and Warcraft were written in C++. I show them how it is easy to write a menu-driven program using if-statement and switch statement, which is also how Counter-Strike allows a player to select its choice of ammunition. Similarly, I showed CS6848 and CS565 students how Jane Street Capital uses strict typing to capture bugs in input data and can build liquid types in its OCaml codebase to annotate the data at compile time to efficiently filter data at runtime. The next step requires assigning students with thoughtful assignments, which promote thinking rather than rote learning. For instance, one assignment I gave to ECS265 students required them to explore the impact of an optimization on PBFT; what will happen if we remove one phase of PBFT protocol? This assignment required students to show that simply eliminating a phase diverges the states of the replicas. One way to guarantee safety is by increasing the number of matching responses needed by the clients from $f + 1$ to $2f + 1$ and allowing transactions to rollback.

Propose. The eventual goal of my 3Ps model is to allow students to propose new research problems. This goal is comparatively easier to meet in a research-based course than in a standard course. Hence, I always work with the course instructor to offer students opportunities to work on a project. The goal of each project is to allow students to propose an idea that excites them and help them explore that idea. This process requires having weekly meetings with students. Often students cannot decide on a feasible project in a couple of weeks. In such a case, I give them several options. For example, for the CS1100 course, I asked students to design games like Tic-Tac-Toe and Pac-Man, and for the ECS289F, I asked students to build decentralized applications like secure NFT auction platforms and blockchain explorers like EtherScan. For an introductory programming course, games like Tic-Tac-Toe provide sufficient coding challenges to the students and force them to use different programming patterns like conditionals, loops, and recursion. Building decentralized applications requires students to design a replicated client-server application and ensure their application is consistent (prevents ledger forks and attacks like double spending) and available (despite failures). To interested students, I also proposed research problems or invited them to join ongoing research projects, which in several cases has led to publications in top research venues.

Courses. I can teach standard database, distributed system, operating system, blockchain, and programming courses. Additionally, I would like to introduce two new graduate courses: (1) Distributed System Design and (2) Principles of Fault-Tolerant Consensus. These courses would also be available to undergraduate students who have taken the introductory database and OS courses. Through *Distributed System Design*, I want to introduce students to challenges faced while designing real-world distributed applications. During my mentoring and teaching sessions, I have observed that often students do not know how to measure system metrics using tools like htop, iotop, and nload. Further, students are unfamiliar with profiling the system using perf tools and analyzing bottlenecks through flamegraphs. Similarly, although students know the importance of parallelism and need to judiciously use locks, I have observed them (i) creating embarrassingly parallel codes where the number of spawned threads is much more than the available cores, (ii) using lock-free queues without anticipating the bottlenecks they create if accessed through synchronous queuing operations, or the delay they introduce for subsequent events if accessed asynchronously, and (iii) forgetting the impact of reduced bandwidth and high latency when machines are geo-distributed.

In the *Principles of Fault-Tolerant Consensus* course, I want to help students understand consensus protocols, because they are at the core of any distributed system. In the first half of the course, I want to work with the Paxos protocol and show students the impact of different optimizations on Paxos. In the second half of the course, I will start focussing on the PBFT protocol. Both of these protocols can be subject to numerous optimizations, each of which renders the guarantees expected from the system. Some such optimizations include reducing phases, reducing the quorum size, burdening the client with transaction ordering, parallelizing and pipelining consensus phases, and clustering replicas into groups. My interest in unwrapping the finer details of consensus protocols to students motivated me to co-author the book, titled *Fault-Tolerant Distributed Transactions in Blockchain* for Morgan Stanley and Springer.

Mentoring. Apart from teaching, each professor plays the role of a mentor. As a mentor, a professor advises undergraduate, master, and doctoral students. This role is more critical in the case of doctoral students as each professor helps to shape the career of its students; like parents teach their children to walk, professors guide their students to traverse through the academic landscape. I envision the role of mentoring undergraduate and master students as different from that of mentoring doctoral students; the former requires concise short-term planning, while the latter requires elaborate long-term planning. For either set, I intend to apply my 3Ps model: (1) promote participation by scheduling weekly 1:1 meetings, (2) promote practice by suggesting the related work to read or test, and (3) encourage propose by offering them with a non-judgemental space to express their ideas.

In the case of undergraduate and master students, often applying the 3Ps model requires giving them a precise project and setting up weekly goals/tasks to complete. Additionally, I have observed that students work better in groups as it allows them to discuss problems with each other. Hence, I try to pair students whenever possible. In the case of doctoral students, my goal is to assign them an initial research problem to work on. The initial research problem aims to guide students towards quick preliminary results, which can act as a moral booster and help them discover the broad research problem. To my doctoral students, I want to emphasize that a Ph.D. is like running a marathon where you start slow and sprint the last couple of laps. No two Ph.D. students have the same journey; most have bad days where their papers get rejected, and they hit a dead-end, but through persistence, each paper eventually finds a good home.

Fortunately, during my Ph.D. at Davis and Postdoc at Berkeley, I have worked with some amazing undergraduate and master students. Several of these collaborations have resulted in new findings and subsequent publications. The most noteworthy of these is with Shubham Pandey, who worked with me for around 8 months on our Eurosys'23 paper. During this period, Shubham was instrumental in implementing MinBFT and MinZZ protocols and exploring safety violation due to rollbacks in existing TrustBFT protocol. This work went on to receive a Best Paper Award. Several other collaborations have led to fruitful results: (i) Dhruv and Rohan worked with us to analyze and implement the AHL protocol, which served as a baseline for our RingBFT protocol [EDBT'22], and (ii) Erik Linsemayer helped in linking serverless Google Cloud functions, which helped to implement our Serverless-Edge codesign [ICDE'23]. Recently, Ethan Xu has worked with me to integrate etcd's Raft with our upcoming Scrooge protocol. Finally, Chun Deng and Shubham Misra are working with me to reduce the storage consumed by existing blockchain systems. Chun and Shubham have explored codebases of Algorand, Ethereum, and Bitcoin and shown that most of the existing blockchains have an ever growing storage needs and their nodes require at least 700Gb disk space. They are trying to formulate cryptographic algorithms that can split the storage required at each machine.

Service. The final goal of a professor during its academic tenure is to serve the department and the larger community at hand. In past, I have been actively involved with several organizations for the betterment of the community. During my masters at IIT Madras, I was part of the *Mitr* group where our aim was to identify fellow students who are under mental distress and prevent any potential cases of suicide. Our duties involved cautiously reaching out to a student under stress and guide them to the on campus resources. We were trained to immediately notify the Mitr counsellors and to not attempt counselling the concerned student.

At UC Davis, I served as a member on the Graduate Student Association's Treasurer Advisory Committee (TAC). Our goal was to allocate available funds to graduate students for their conference travel. Due to limited funds and extreme competition, it was my duty to remain impartial and select the students who will receive the travel fund. I ensured that we follow the principles of Diversity and Inclusion and reward students of various backgrounds.

At UC Berkeley, I was actively managing the Sky Seminar from April 2022 till August 2023, which provides Berkeley students with a unique opportunity to learn from and meet with renowned professionals from industry and academia. Due to the pandemic, the Sky Seminar had almost died down, and I was tasked with the job of scheduling it regularly and finding a good set of speakers for this seminar. Additionally, I focused on having a diverse set of speakers, which would allow students of different backgrounds to feel represented.

Finally, I have served the computer science community as a reviewer for a large set of conferences, including ICDE and SIGMOD, and artifact evaluator for SOSP, OSDI, and ATC. I am also working as an assistant editor for Journal of Systems (JSys) Research and have served as a web chair for Middleware and FAB conferences.