Teaching Statement

Mukesh Tiwari

August 29, 2022

My teaching philosophy is to not immediately reach a solution but to develop a thinking process (problem solving mindset) that leads to the solution. I believe every student is different and has a unique style of learning, and my role is to help them find and hone their style.

1 Teaching Experience

When I started teaching as an assistant professor at the International Institute of Information Technology (IIIT), Bhubanesware, India[1], my single biggest challenge was keeping the students engaged in my class, especially the first year students in C programming course. At the IIIT, I taught C programming to first year students, Compiler Design and Java programming to third year students, and Cryptography to final year (4th year) students. In each course, every single problem, more or less, boiled down to keeping the students engaged in a topic. In order to keep them engaged in a class, I took a Coursera course on learning[2] and read many academic articles and non-academic articles about effective learning. I tried these techniques in my classes, and below I describe my experiences with teaching and efforts to engage my students.

1.1 Setting Clear Goals

In every course, I started with the end goal of the course. For example, in C programming course, I told my students that by the end of this course they should be able write simple C programs, e.g., calculator, time tracking system, student attendance system that automatically calculates their attendance percentage, validating a debit card, and other simple real-world problems. The rationale was to show a vision to excite them for learning and instill the feeling of empowerment, by a narrative from being consumer to being creator of software programs. In addition, in the beginning of a every single class I would tell my students explicitly the topics we were going to study and their importance. For example, when I taught pointers, I explained a very high level idea of an operating system and need to access memory in the course of booting the operating system. This demonstrated the need and importance of pointers. In addition, I told them Linux is written in C and encouraged them to check the source code. This two step process helped me on focusing the big picture, while at the same time tracking the progress towards the big picture.

[1] It is a small technical school, specilised for information technology.
[2] `https://www.coursera.org/learn/learning-how-to-learn`

1.2 Start with Why

One thing that I learnt by teaching for 3 years is that if you want a concept to stick in someone's mind, start with a *why*[3]. For example, when I introduced functions in C programming course, I wanted to convey the idea of *why do we need functions?* Therefore, I started the class by asking them to write a code, using pen and paper, to compute the **factorial of a number** n. Every one was comfortable with loops, so it was quick. I then asked them to write another program that computes the k^{th} **power of the factorial of** n. In this assignment, some added an outer loop to cover the factorial computation, and some added another loop after the factorial computation (stored the factorial computation result and used it in later computation). At this point, I asked the class how one could write a program that computes the **factorial of the** k^{th} **power of the factorial of** n. It slowly sinked to the class that they need to duplicate a lot of code, and once they had this revelation I introduced functions and showed them the solution of the previous problem by means of function composition.

1.3 Catching my Mistakes

To promote active participation, in the beginning of every class I would announce that there would be a few instances where I would write the solution of a problem wrong intentionally, and your goal is to catch me on the spot. If you succeed, you would be awarded class participation marks. If no one caught me, then I would tell the mistake, but no one would get any class participation marks. In every single class, especially in programming classes, I would first teach a topic and then solve problems on a blackboard related to the topic. It was during the problem solving where I made deliberate mistakes. It increased the class participation by order of magnitude[4]. In every single feedback that I got during my 3 years of teaching, almost everyone appreciated this idea of making deliberate mistakes.

1.4 Projects for Learning

For every course that I taught, I designed projects related to the concepts and ideas present in the course. It was a part of my teaching philosophy to impart critical thinking. Furthermore, I asked my students come up with their own ideas to promote idea-exploration, a key step to develop a critical thinking and problem solving mindset. In the process of idea-exploration, students came up with various interesting ideas, but one that still stands out is a group of third year students that wanted to understand the workings of a a DNS server by writing their own. Even though they could not write a DNS server from scratch, they managed to understand most of the workings of a DNS server by downloading a C code from the Internet. I consider it as a trophy for myself. In addition, because of my own competitive programming background, I redesigned the C programming Lab and introduced them to competitive programming[5], which led to first ACM-ICPC team participation from IIIT. More importantly, it made many of the students curious about programming, which was considered a difficult subject.

[3] I learnt this idea by reading the book *Start With Why* by Simon Sinek.
[4] I learnt this from advertising agencies where they write some ads wrong intentionally to grab the attention.
[5] `https://www.topcoder.com/community/arena`, `https://www.spoj.com/`

2 Potential Courses

I feel qualified to teach most of the computer science courses because of my background in computer science, but my natural preferences are courses close to my research area, e.g., theorem proving, cryptography, logic and its applications in computer science, discrete mathematics, algorithms and data structures, introductory programming course (C/C++/Java/Python/Haskell/OCaml), foundation of computing, etc. In addition, I would also like to design a course to teach various voting method used around the world and their pros and cons from a social choice theory perspective. Apart from these topics, I would be more than happy to teach other courses, given enough preparation time, at introductory and intermediate levels, e.g., type theory, theory of programming languages, networking, operating systems, databases, etc.