Experiences Teaching Computer Science

Vincent La
CS 190J

March 16, 2018

Over the past quarter, I had the experience of mentoring students taking lower division computer science courses. Having been a tutor for math courses, and routinely being the study buddy who ends up explaining concepts to everybody else later, I was initially very confident in my abilities. I thought it was going to be an easy, breezy quarter where I spread the good word about C++. But contrary to my expectations, teaching computer science was not an automatic reflex for me, and I had quite a bit to learn. In this paper I outline both the challenges I faced, how I overcame them, and the lessons that I learned.

Teaching

Although I was far from a perfect teacher, I was still effective in helping my students. Combined with patience, and being a good listener, there were also a few strategies I relied on.

Interviewing the Student

Every student is different, so you need to be flexible enough to change directions if a certain strategy is not working. But it is also better to be proactive rather than reactive, so one technique I use is asking lots of questions to get a sense of what my students know and do not know. If there were a recipe for teaching, this would be a basic ingredient that I added other items upon.

Scaffolding

Often times, a student will be overwhelmed by an assignment because they do not know where to start, and may not even know which language constructs to use in their programs. In these situations, I try to build the student's knowledge up piece-by-piece. First, I identified which concepts were fundamental to solving the program, asked the students how comfortable they were with them, and planned based on the response.

For example, if I got a worried look in response to the mention of "for loops", I first demonstrated basic use cases, and then got students to write for loops for other basic cases.

Here, I provide a scaffold for the student's knowledge, and build their knowledge piece-by-piece. In the for loop case, I gradually asked my students to implement harder and harder for loops until eventually, I was confident that they had sufficient knowledge to complete the lab.

Just a Hint

As a math major, there were plenty of times where I was completely lost on how to do a proof. After going through several sheets of scratch paper, searching through my textbook without direction, I explained my circumstance to my TA. Maybe after a minute or two of thought and asking me about what I knew, they would then confidently toss a helpful hint in my direction, asserting their small tidbit was all I needed. Some TAs would then smirk, as if they were insensitive to my past hour of suffering.

But then I started doing that myself. Most recently, there was student who had finished a lab that involved writing his own test cases. He seemed quite confident in his code, but he could not pass the adversarial tests on submit.cs. Believing that his test cases were very robust, he was vexed by the red failure messages. I asked him if he covered edge cases such as empty linked lists and lists with only one node. He told me he had done so, and even showed me the area in his code where he did so–and he was not wrong. Despite his frustration, I assuredly told him that the test suites were not mistaken.

I then noticed that the test suite had tried to insert the same number twice. I then suggested that perhaps his code did not handle duplicate insertions well, and try to implement a test which covered that case. I also asked him to think real hard about even more tests, reminding him that a lot of real world software breaks frequently due to unforeseen issues.

When this strategy was used on me, although I hated it, I gained a sense of pride and accomplishment having completed the proof using my own smarts (with an assist by my TA). At the cost of having my own students resent me, I used this because it could be effective in the right circumstances, and allows the student to gain confidence in their own abilities. Lo and behold, a few minutes later, he created a test suite that segfaulted, which gave him a window into why his code was not passing the submit.cs test suite. This is also the first time in my CS career I was happy to witness a segmentation fault.

Swallowing My Pride

One of my strengths was that I was able to put my pride aside and ask other mentors for help. I recall I was helping this one student through her CS 8 assignment, one of the problems which entailed implementing a function which returned true if a word could be produced from the letters in some string. This was sort of like an anagram function, although unlike an anagram the string could have extra (unused) letters.

Probably before the student I was helping had finished reading the question, I already had an algorithm in my mind. I was going to build a counter of letters for both strings

using Python's dict. Comparing the counters, I could easily check if the string had enough letters to build the word. However, the wording in the lab suggested that a totally different implementation, one involving pop and remove operations from a list.

Because I was so invested in my idea, I got a bit stuck, so I asked the other mentors in the lab if they had any idea. Very quickly, they agreed the correct idea was to convert both strings into lists. From the list formed from the word, we would keep popping off the last item, checking to see if it was in the list of letters. Once we confirmed that it was, we would "scratch it off" by removing it from the list of letters. Rather than force my implementation on an unwitting CS 8 student, with the help of the other mentors, I was able to guide her through an implementation that was more aligned with her professor's wishes and course objectives.

Providing Motivation

Learning to program, especially in C++, is something that requires a bit of motivation. As a mentor, I thought it was not only important that I taught my students, but that I also motivated them. For example, I once saw a pair that seemed a little bit exhausted by lab involving linked lists. "Do you think you can get this?" I asked one of them to which she unethusiastically replied, "Maybe." "Do you know why I think you will finish this?" "Why?" "Because I believe in you."

As corny as it is, it is very easy to get flustered when programming, so I find it helpful to provide comedic relief once in a while.

Research

In the paper I read, a point that stood out to me was how a majority of students wanted their mentors to be confident in their answers. In my experience, there is a place and time to show confidence. As mentioned earlier, sometimes a display of confidence can motivate students. However, I recently helped a student who–with the help of another mentor–correctly implemented a function that accomplished a task which the lab never asked for. Once I had the student open up the associated header file, we both realized that a good hour or so of time had been wasted. Perhaps the mentor could have been a little less confident?

When I tutored, I openly used Google and reference websites for answers. I routinely answered "I don't know" and referred to other mentors or looked up the answers with my students. Perhaps it would be more prudent to advise tutors to approach problems with confidence–not in their answers, but their overall method.

However, one tip I read from the paper did turn out very fruitful, which was circulating around the lab frequently. Whether in section or lab, it is tempting to say "nobody is raising their hand, so I can just sit in and do my own work." However, I personally ended up answering a few nontrivial questions by walking around and popping students with a quick "how's it going?" once in a while.

Service

In addition to tutoring, I also tried to improve the course itself by breaking up partners which I saw were not working well, writing Piazza posts, and editing the lab documentation. Although at the beginning I had four pairs of students, by the end of the quarter I had one–the pair that I matched myself. Most pairs broke off naturally, i.e. one of the students simply stopped working with their partners. However, there was one pair that was only broken when I emailed the partners asking why they were not collaborating with each other over GitHub.

In addition to making sure pairs played nice, I wanted to make sure the class as a whole interacted well, e.g. on Piazza. Like any community forum, Piazza is a product of its users and most importantly–the moderators. I wanted it to be a welcoming place where students who were stuck could lean on the expertise of their fellow students and course faculty. I was told that in one CS 130A Piazza post, one student's question received a reply along the lines of, "Maybe you're not cut out for computer science." Happily, we did not witness any such negativity of that sort with our class.

However, one pesky trend I noticed (around midterm time) was a rash of questions asking "What does this C++ code do?" It was quite worrying, because the inquiring student could simply copy the code, and then compile and run the program to get the answer within seconds. I did not want Piazza to be a crutch for students only interested in doing the bare minimum. So, to prevent enabling them, I decided to write a strongly-worded, but non-condescending post, reminding students to try a little bit harder before they post.

Lastly, I tried to complete the majority of the labs–my own personal schedule permitting. At times, I found some of the instructions to be somewhat confusing, or unnecessarily verbose. However, I was reluctant to edit them because I thought "the professor must have written it that way for a reason," but that all changed one afternoon while I was completing a lab which involved points, boxes, and methods for determining if two boxes occupied the same space on a plane. There were no provided test cases for my box methods, but I was told to write my own and model them after the test cases for the point methods. Specifically, I was to test a method which encoded the geometry of the box in a string, with the desired precision of floating point numbers being passed in via a parameter.

After reading the test suite for the point methods, I thought to myself that I my test suite ought to have at least two different box instances, since that was the case for the points. Furthermore, since the lab only vaguely mentioned that I should test my box to string for "different precisions", I simply tested them for a few arbitrary numbers of my liking. As I ran my code through submit.cs, I still felt a moment of apprehension, the same feeling I had when I took CS 16 as a student–even though the score did not matter this time. When the screen refreshed, splashes of bright red highlighted that my code had failed some tests.

"This can't be," I thought to myself. Ironically, none of the actual code I wrote broke, but the tests for my test suite failed. Despite the lack of any mention in the lab instructions,

the tests were expecting me to test only one box, and I was supposed to assert that the box's string representation was accurate for precisions from 1 to 6. I re-read the instructions multiple times to confirm. "This is bullshit," I thought to myself. After modifying my tests such that they passed those on submit.cs, I angrily modified the lab instructions to include the expected output of the box test suite. I reasoned to myself, there was simply no way the students could guess or conclude that their test suite should take that specific form. Ironically, my edit was not clear enough, as yet more students ran into the same issue I did and asked me about it.

It was on that day that I learned that computer science professors were not omniscient, and that they could make mistakes and be humans too.

Course Feedback

Although the course was a valuable experience, it still much to be desired. For instance, I did not think enough course time was devoted to practicing and improving our actual tutoring skills. While my ability definitely improved, much of it was either due to feedback I received from other mentors, or through mental notes I took observing others. I really appreciated the experience of being observed, but I thought the helpful criticism I received would have been more useful if I had received it earlier on in the quarter.

Conclusion

Although there was a lot of tedium associated with teaching, such as writing out student evaluations, I still found it to be a great experience. I wrote this in the hopes that future tutors will gain some lessons, and in the hopes that whoever read this found it an enjoyable experience. Furthermore, I hope that with enough tutors, future labs will feature more concise, less ambiguous wording.