I care deeply about teaching. The aim of this document is to illustrate the approaches I use, and to expound upon my opinions about the qualities that make a great teacher.

The overarching theme in my teaching is this: I wish to be the professor that I would have loved to have had as an undergrad. It can be difficult for all of us to recall the people that we were in our youths. But I try to remember all of my professors—both good and bad—and remember how they looked to the eyes of a 20-year-old. I want to be the teacher that the students appreciate, who prepares them thoroughly for graduate school or a job in industry. I want to be the professor that students may not like every day, but whose class they remember five years later as the one they are very glad that they took.

As part of my formal education, I completed a Delta Teaching Certificate at the University of Wisconsin. The Delta program (`http://delta.wisc.edu/`) is a curriculum for science and math PhD students at the university who wish to focus on their teaching skills, in addition to the standard research requirements of a doctorate. The program emphasizes three pillars: "teaching-as-research", "learning-through-diversity", and "creating a learning community". I took educational classes (including teaching theory and teaching using technology) to complete this course of study, while also completing a semester-long internship with a local space museum.

The vast majority of my teaching decisions have been geared toward being a professor at a small liberal arts college. I have very fond memories of my own time as a student at Lewis & Clark College in Portland. I believe that a liberal arts education can create a well-rounded student who is able to think critically, to understand the underlying issues, and to write and speak eloquently about them. And I believe that computer science and algorithmic thinking are very important parts of this equation, to educate a young person to live in the modern world.

All Students Are Individuals

This is one of the most important rules to keep in mind when teaching at a small liberal arts college. Since earning my doctorate, I have spent most of my time working with bright students in a small, intimate environment. Having attended a liberal arts college and been one of those students myself twenty years ago, I think I understand their outlook better than most professors. Many of them were in the top of their class in high school, and are used to receiving good grades with minimal effort. They are out of their parents' houses, and still deciding whom they wish to be and what they want to do with their lives. It is my job to introduce them to computer science and computational biology—two things which I believe to be inherently cool, and which can quite literally change the courses of their lives. I try to do so in such a way that is inviting, while still being extremely rigorous and maintaining high standards. It is also vital to understand that students and their families are making large financial sacrifices to attend a small college. Without a personal level of attention, they will not stay the whole four years. If they believe that they are personally appreciated and valued, they are much more likely to graduate with us.

At the very minimum, a teacher at a small college needs to create an inclusive classroom environment and be flexible in his or her expectations for students with different needs. Three years ago, one of my coworkers explained one of her teaching rules to me, that I've come to think of as "the rule of ten". In a classroom with ten students, always assume that one of them is x. Here, "x" can mean gay, conservative and religious, struggling with depression, being a survivor of abuse, or any other quality that can make a student feel isolated and alone. A good teacher needs to be able to see the class from the students' perspective as well as his or her own. I have experience dealing with students with learning disabilities, serious injuries (including concussions), severe family issues, obsessive-compulsive disorder, autistic-spectrum disorder, histories of sexual abuse, drug-abuse problems, and suicidal-level depression. Many of these students have thanked me at the end of the semester for being understanding of their needs, while at the same time keeping up the pressure on them to finish the class.

Computer science frequently suffers the reputation of being an "old boys' network" which is not accepting

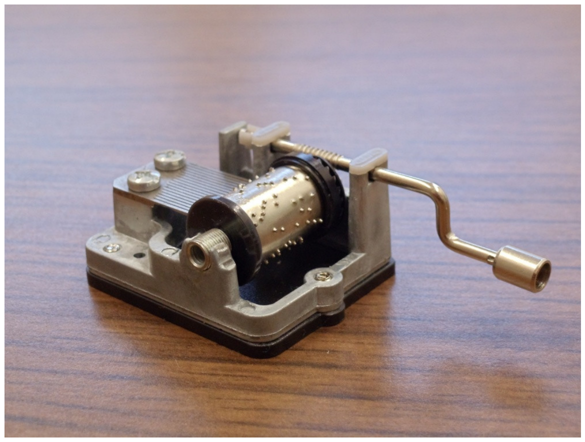

Figure 1: Digital technology from our great-grandparents' time. This music box encodes its tune digitally as bumps on a cylinder. Showing this to students helps them have a more concrete idea of what "digital" really means (and that it is not just a synonym for "technological").

of outsiders (especially women). A teacher who is disconnected from his or her students does not help this situation. My approach to help solve this problem is to encourage my students on a one-to-one level. No one wants to be singled out in public because of his or her gender or race—even if the intention was positive. But I believe that encouraging a student in private or via e-mail can be much more heartfelt and believable. When I see students who are struggling and who I suspect may not feel that they belong, and I believe that they are more capable than they think they are, I will go out of my way to tell them that. I have had students thank me years later for this approach.

I try to be accessible to my students. For example, I try to work with my office door open when possible, if students have questions. My students also know that I check e-mail late into the night, and I have found that they appreciate prompt responses when most of the world has gone to bed.

My first semester as a TA in grad school, one of my students was having trouble deciding on a major. He thought that his previous schooling did not prepare him well enough for computer science. I listened to him and answered his late night e-mails. At the end of the semester he sent me a note thanking me for my help, and said that I was one of the best TAs that he had ever had. In a lecture class of over a hundred students, I had helped him to feel more grounded and more that he mattered. This is the way to encourage students to persevere with their courses of study.

Make the Abstract Concrete

Years of school have trained me how to understand the function of a complicated algorithm, simply by reading a mathematical description of it. However, it is easy to forget that this is not an ability that comes naturally. Beginning students will often trip over something that accomplished programmers consider to be elementary. Thus, any tool that can make an abstract concept more visualizable is a welcome one. For example, beginning students often have difficulty understanding what "digital" and "analog" mean, often assuming that the former is just a synonym for "technological". In one of my introductory courses, I brought in a small music box on the first day, with the tune encoded as bumps on a cylinder (see Figure 1). This object would not have been considered exotic in the nineteenth century, and yet the method of data storage is decidedly digital. By showing an object that I could pass around the classroom, I solidified the meaning of this word in their memories. This is what I mean by "making the abstract concrete," and it drives much of my lecture design.

Of course, not every concept has a small toy that can demonstrate it. I make frequent use of figures and animations—many of them that I have drawn myself. My lecture notes on algorithms to reconstruct phylogenetic trees are located at `http://mathcs.pugetsound.edu/~adamasmith/sample-lec/`. These are from the course I designed in computational biology. I use HTML for my lecture notes for several reasons.

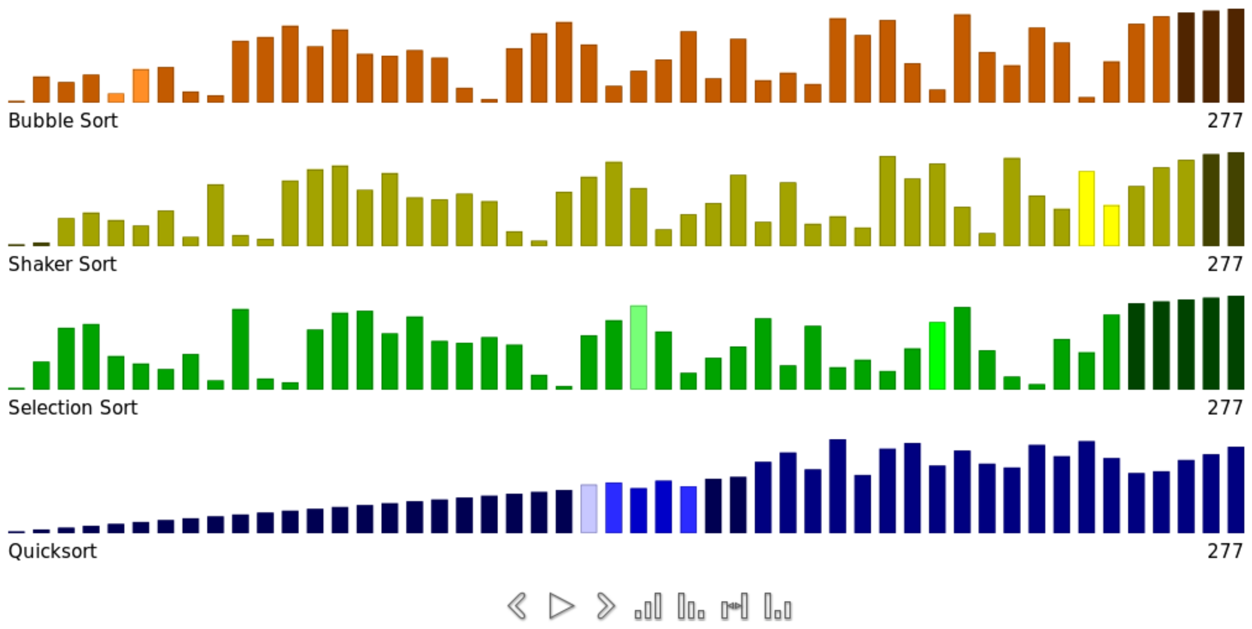

Figure 2: An applet I designed to illustrate sorting algorithms. Students are much more likely to understand an algorithm if it can be shown in a concrete manner. This applet is designed to be dropped into a lecture, or to be accessed by students on their own time.

The environment makes it much easier to create my own equations and figures. There are no discrete slides that disappear before students can write down their contents. It is also easier to share these notes (after the lecture, to encourage note-taking), since no special reader beyond a web browser is necessary to view them.

I am currently working on creating smart animations, that actually perform a given algorithm in real time and allow the students to see how they function. One such example, demonstrating sorting algorithms, can be seen at `http://mathcs.pugetsound.edu/~adamasmith/sorters/`. It is also illustrated in Figure 2, above. Most computer science students know that a $O(n \log n)$ algorithm like Quicksort will usually finish more quickly on a given data set than a $O(n^2)$ algorithm like Bubble Sort. But seeing the algorithms in action gives the students a much deeper understanding of the processes involved. It helps them to implement their own versions, and to make good programming choices down the road. Recently, I have been awarded some small grant monies to lead students in developing similar applets.

Another, older applet that I created is an interactive virtual planetarium, as seen in Figure 3. I called it "cSky", and I designed it using Adobe's Flash environment* as a part of my Delta Teaching Certificate. The program can be viewed online at `http://pages.cs.wisc.edu/~aasmith/csky/csky.swf`. It shows how the entire night sky moves over time from any location on the earth. Although it is not as refined as the sorting applet, it still illustrates the principles I value in such a program. Teachers can insert it seemlessly into a lecture to illustrate important concepts, while students can access it from home on their own time to reinforce these ideas. The applet illustrates many important phenomena, including the rotation of stars around the pole, the cause of the seasons, the sun's path through the ecliptic plane, the wanderings of the planets, and many others. To test it, I conducted a trial of college non-major astronomy students. Self-reported answers indicated that not only did it significantly improve students' understanding of the material, but it also erased a gender gap in perceived understanding that had been present in the pre-testing.

These applets also provide another opportunity. They are time-intensive to create, yet once finished they can be shared among teachers easily. Thus, I believe it will be possible to apply for further grants to hire students to create more of them, especially relating to bioinformatics algorithms.

*Unfortunately, this means that it cannot be viewed on many modern smartphone and tablet devices.

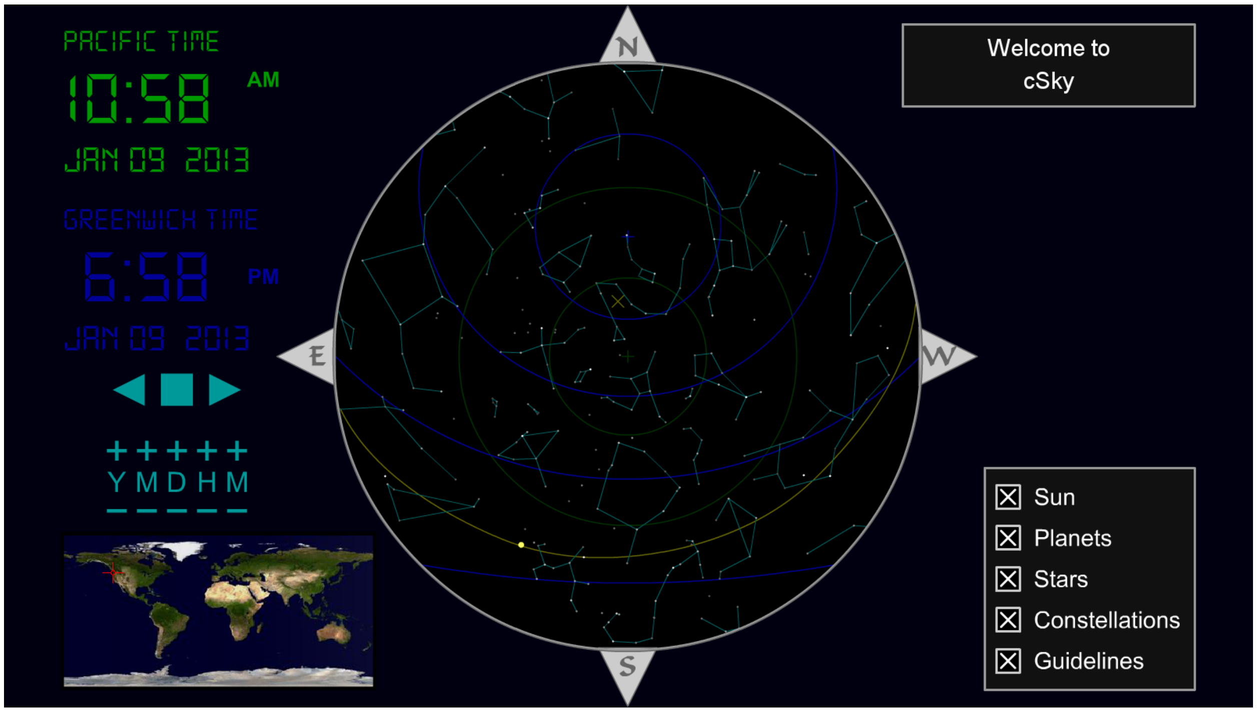

Figure 3: Another applet I designed to be a virtual planetarium, called "cSky". This one can be used to show the entire night sky moving from an arbitrary location on earth, through arbitrary times. It is also intended to be a teaching tool, that students can access freely to reinforce concepts learned in class.

Challenge One's Students

I am not an easy professor, and I consider this to be a point of pride. Although I would never give a student a bad grade for its own sake, I am a firm believer that a student receiving an A must show mastery of the material taught in class. A class should be worthwhile for the knowledge imparted, not for its effect on a student's GPA. My goal in every class that I teach is for it to be a class that I would have loved to have taken as a student: challenging, educational, and worth the time spent.

In 2010, for the first time I was able to design and teach a class in computational biology. This was a challenging course to create, since there were many students from varying backgrounds who were interested in the course. I ended up with a class of about 20, roughly half biologists and half computer scientists. I partnered them up, so that each team had one person from each background. For each unit in the class, they read a paper together about a recently solved problem in biology, and then implemented the program necessary to solve it. It was difficult to keep the class together, all working at the same pace. However, at the end several students approached me and thanked me for everything they had learned. One student in particular, who had been critical the whole way through, finally admitted that he did not truly know how to program until he took my course. The next time I taught it, the class went much more smoothly.

Another significant experience came when I taught a survey computer science course for non-majors. Many students in the class confessed to me that they had enrolled in it because they had heard it was an easy course—some from their advisors! I told them on the first day that if they wanted in easy class, they should drop out and take it another year. Instead of a soft course about the Internet, I gave them a rigorous course on binary and hexadecimal arithmetic, circuit design using a hardware simulator, programming in Python, basic artificial intelligence, and how to encrypt and decrypt a message using an RSA public-key cryptosystem. The feedback I received at the end of the class confirmed that the students had learned far more than they had planned on.

Conclusion

When I first started teaching, I made a list of all the professors from whom I had taken classes. Next to each one's name, I wrote a single lesson, good or bad, with regard to their teaching style. This exercise was extremely valuable, since it allowed me to distill and to quantify many of the traits of a good teacher. By watching my professors, I learned tricks for making good exams and assignments, that I can make myself into an extrovert during class and that that helps engage students, and that communicating my excitement about the material makes the students excited about the material. I also learned some traits to avoid, such as expressing frustration with students in the middle of class.

There is one more rule that I follow, that every computer scientist should be able to appreciate. Don't be so attached to your ways of doing things, that you stagnate and stop growing. Every computer scientist runs the risk of becoming a dinosaur, teaching outdated skills to people who have moved on. The same is true of teaching. I fully admit that I could be wrong about everything I have written here. For this reason, I will continue to challenge myself, and to consider ideas that I may not like right now.

My student evaluations confirm that I am a good teacher. My goal is to be a great one. To achieve that, I will have to continue the process of learning myself, for as long as I am able.