Teaching Statement

Yihan Sun
yihans@cs.cmu.edu
http://cs.cmu.edu/~yihans

Teaching is one of the most important duties of a faculty member in universities. In addition to enjoying the responsibility, I view teaching as a rewarding way to interact with bright young minds through a process of collective improvement and mutual learning for both the students and the teacher. I am willing to become a dedicated teacher and advisor and lead my students with enthusiasm to the broad knowledge areas of computer science, comprehensive capability in research, and a bright future.

In the following sections, I will first state my past teaching experience and advising experience. Then, I will elaborate on how my research experience can help develop distinctive courses. I use two examples, efficient algorithms with simplicity and parallel algorithms: theory and practice. I also state my teaching plan in these two examples. Finally, I will conclude with my teaching interest.

Teaching experience. My teaching experience started in high school, when I served as a math teacher in a summer school to help junior high school students with their studies. During my undergraduate studies in Tsinghua University, as one of the best students in our class (25 students), I was responsible for giving informal weekly recaps of classes and helping students who had difficulty understanding the course. I have taught a number of courses including calculus, linear algebra, data structures, graph theory, operating systems, and so on. For both experiences, I learned that teaching is more than just thoroughly understanding the materials myself, but also includes conveying the bright spots of deep and abstruse knowledge to people with a weaker background. This forces me to rethink the intuitions and connections between knowledge and express them in a vivid way that is easy to remember.

In CMU, I was exposed to more opportunities in teaching. I have served as a teaching assistant (TA) for two courses, 15-451/651 (Algorithms), which is the advanced undergraduate algorithm course, and 15-853 (Algorithms in the “real-world”), which is a graduate course on algorithms used in practical applications. For 15-853, I also gave a guest lecture (1.5h, 40 students) on parallel tree structures, which is a classroom-ready version of part of my research work. I also helped design problems for homework and exams, from which I learned to select proper problems that can both consolidate and evaluate the students’ understanding of courses. These experiences helped me develop the ability to explain technical concepts using simple but effective words as well to find the best angle to illustrate involved algorithms from.

Advising experience. One of the most gratifying parts of teaching is to interact with junior researchers. In addition to teaching courses, I have also had opportunities to mentor several undergraduate and master students in conducting research. Advising students provides me with the chance to think of research at a higher level and develops my ability to seek research topics. My collaboration with undergraduate and master students has led to several conference publications in top conferences such as SPAA and PPoPP. This experience has built my confidence and furthered my interest to become a good advisor.

Efficient algorithms with simplicity. Part of my research includes developing efficient algorithms with simple implementations that reduce the coding effort for a list of algorithms and data structures. A key idea that I used a lot is to model different applications by abstracting the common components, which especially provides a clean interface for teaching. For example, I have worked on an algorithmic framework of balanced binary search trees, which unifies
multiple balancing schemes. This framework bases all the tree algorithms on a single primitive, such that all the other algorithms (except the primitive) are identical across different balancing schemes (e.g., AVL trees, red-black trees, weight-balanced trees, treaps). This abstraction is especially classroom-friendly because students can avoid dealing with the frustrations of memorizing different algorithms for insertion, deletion, and the like for each different balancing scheme. In addition, this framework yields highly-parallelized algorithms. This framework is already used in existing courses. A simplified version of this framework is taught in CMU course 15-210 (Parallel and Sequential Data Structures and Algorithms), which is the entry-level algorithm course. A more advanced version was also taught in my lecture in the CMU graduate course 15-853 (Algorithms in the real-world).

The second example is an abstract data type (ADT), the augmented map, that I proposed in my research. This ADT effectively models many real-world problems, especially some low-dimensional computational geometric problems. By introducing the paradigm with simple examples, students should be able to draw inferences and know the solution to a set of problems through analogies.

I would like to teach courses on algorithms and data structures with high-level abstractions to show the elegance of algorithm design. I am also happy to teach courses on general and broad topics in algorithms and data structures and introduce these useful toolkits as part of the course.

Parallel algorithms: theory and practice. Nowadays, even PCs and smartphones use multi-cores. For software engineers, it is imperative to take parallelism into consideration in programming. As a result, even undergraduate students should be equipped with the necessary knowledge of parallel programming. Just like learning sequential algorithms, learning parallel algorithms involves a thorough understanding in models, theory, writing real code and debugging in practice. I have experience in both designing parallel algorithms and developing parallel libraries, and thus am eligible to teach topics related to parallel algorithms, computing, and programming.

To teach parallel computing, it is worth starting with straightforward examples to develop students’ interest by giving them a sense of the power of parallelism. Then, I will introduce the foundations of the theory of parallel computing, such as analyzing the complexity of parallel algorithms. After that, I plan to present several practical and classroom-ready parallel algorithms and data structures, like sorting, permuting, graph algorithms, balanced trees, and more. Along with the algorithms, some general techniques will be brought up, such as divide-and-conquer, packing, pointer jumping, and randomness. Later, concurrency issues and system-level concerns should be addressed, like race conditions, contentions, scheduling, and so on. It is also worth mentioning several parallel libraries and let students have exercises to implement simple algorithms. Finally, I believe that working on projects is the most efficient way for students to grasp knowledge. Besides improving their problem-solving ability, finishing projects brings up a sense of achievement and confidence in students and further develops their interests in computer science.

Because of the profoundness of parallel computing, it is also possible to have courses purely on theory and programming, or have courses on entry and advanced levels respectively. I believe that I am eligible to teach any of them. I have also worked on a wide range of applications utilizing parallelism and am happy to teach other courses (as I will list below) and combine parallelism in the course design.

Teaching Interests. As stated above, I will be happy to teach any course related to parallel computing and most theory courses, including courses of different levels on algorithms and data structures. I have worked on research topics in many different areas, which equip me with the capability to teach a wide range of courses. In particular, I am willing to teach any entry-level undergraduate course including graph theory, the foundation of programming, computational geometry, image processing, introduction to theory, data science, machine learning, and so on. I can also teach mathematics’ foundations in computer science, like probabilistic, number theory, and almost all undergraduate mathematics-related courses for CS majors. I will also be happy to hold overview courses or seminars.