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 responsibility, I view teaching as a rewarding and enjoyable activity to interact with bright young minds, and a process of collective improvement and mutual learning for both students and the teacher. I am willing to become a dedicated teacher and advisor, and lead my students to the enthusiasm and broad knowledge 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, which are **efficient algorithms with simplicity**, and **parallel algorithms: theory and practice**. I also state my teaching plan in these two examples. Finally, I will conclude my **general teaching philosophy** and **teaching interests** in the end.

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 to give informal weekly recitations on revising classes and helping students with difficulties to understand the course. I have taught a number of courses including calculus, linear algebra, data structures, graph theory, operating system, and so on. For both experiences, I learned that teaching is more than just thoroughly understanding the materials myself, but conveying the bright spots of the deep and abstruse knowledge to people with a less solid background. This forces me to rethink the intuitions and connections between knowledge, and to 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 teaching assistants (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. Such experience helped me develop the ability in explaining technical concepts with simple but accurate words, as well as finding the best angle and intuition to illustrate involved algorithms.

Advising experience. One of the most gratifying parts of teaching is to interact with junior researchers. In addition to teaching courses, I also had opportunities to mentor several undergraduate and master students in conducting research. Advising students provides me the chances 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. This experience has built my confidence and interest to become a good advisor.

Efficient algorithms with simplicity. Part of my research lies in developing efficient algorithms with simple implementations, which 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. This also provides a more intrinsic understanding to algorithms, and most importantly, provides a clean interface for teaching. For example, I have worked on an algorithmic framework of 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 of insertion, deletion, and the like, for each different balancing scheme. 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 by analogy the solution to a set of problems.

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 in parallel programming. Just like learning sequential algorithms, learning parallel algorithms requires thorough understanding in from models and theory, to write real code and debug in practice. I have experience in both designing parallel algorithms and developing parallel libraries, and thus am eligible in *teaching on topics related to parallel algorithms, computing, and programming*.

To teach students in parallel computing, it is worth starting with straightforward examples to develop their interests by giving them a sense of the power of parallelism. Then I will introduce the foundations in 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, computational geometry, and more. Along with the algorithms, some general technics 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 develops their interests in computer science. Because of its profoundness, it is also possible to have courses purely on theory or programming on parallel computing, or have both entry level and advanced level. I believe I am eligible to teach any of them. I have also worked on a wide range of applications utilizing parallelism, and am also happy to teach other courses (as I will list below), and combine parallelism in the course design.

General Teaching Philosophy. I believe that teaching and mentoring styles of professors will have far-reaching impacts on students. In teaching, I will **consider and address the challenges faced by the diverse student audience**, such as non-native speakers, students with different backgrounds and pre-knowledge, and students with different expertise. I also would like to **encourage students from other departments to take my courses**, and to develop their interest and skills in computer science. Although courses in computer science are believed to be hard, I also believe that they are useful and beneficial to other disciplines. I would like to encourage students from different backgrounds to apply the useful techniques (algorithms, programming, etc.) they learned in my class to their own disciplines. I am also excited about seeing myself or my students collaborating with them.

In my class, I also would like to **integrate related research results and real-world applications into my class**. I will introduce them in a manner that is easy to understand. I would also like to encourage them reading related papers or solve real-world problems related to my class. Firstly, these activities can motivate the students and intrigue their interest. Secondly, this can be a good preparation for students interested in research or industry jobs. I am also excited about providing undergraduate students opportunities to work on research topics with me.

I would like to **encourage my student to collaborate with people from different backgrounds**. I believe that collaboration and communication with people from different areas and disciplines can bring up new perspectives and will spark new great ideas. In addition, teamwork is important for the students' future career.

Finally, I would like to **communicate closely with students, understand each student's special needs, and adjust my teaching and mentoring through communication**. For classes, I plan to regularly **ask for feedback** from students. I would like to face challenges and solve problems with my students. I would like to cultivate their capability in multiple aspects, such as speaking, writing, communication, leadership, and teamwork.

Teaching Interests. As stated above, I will be happy to teach any course related to parallel computing, and most theory courses, like different levels of course on algorithms and data structures. I have been worked on research topics in many different areas, which equips me with the capability to teach a wide range of courses. In particular, I am willing to teach many entry-level undergraduate courses including graph theory, the foundation of programming, and computational geometry. 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.