Panel Proposal: How Should Data Structures and Algorithms be Taught?

Dr. Danny Kopec
Senior Lecturer/Associate Professor Computing,
Systems Engineering and Management
Richmond, The American International University in London

An undergraduate course in algorithms may be taught on a separate high pedestal, primarily to fourth year students. However, many chances of refinement have reinforced a combined approach that emphasizes the teaching of data structures as implemented by algorithms and some mathematical methods using recurrence relations and their proofs. This approach tackles traditional issues involving choice of representative data structures (arrays, linked lists, stacks, queues), but also considers the particular problem at hand and its unique features.

Two more extreme approaches might approach the subject from different perspectives: 1) a problem solving perspective which would look at typical computer science problems, develop and analyze alternative solutions to them and then consider them from the point of view of algorithmic complexity, practicality of implementation, and transparency for the human programmer. 2) a mathematical perspective which would teach and develop a recurrence relationships for a many algorithms.

Years of experience with the course have led to an approach which combines the "discrete" almost cookbook approach of Robert Sedgewick (***Algorithms*** , ***1988; Algorithms in C++. 1992)*** with the "continuous" mathematical analysis, largely recurrence relation styled approach to the subject by Gregory Rawlins (***Compared to What, 1991***). Typical topics covered will include: elementary data structures, trees, recursion, analysis of algorithms, implementation of algorithms (that is, the first 8 Chapters of Sedgewick) followed by Sorting Algorithms (Sedwick, Chapters 8-13, possibly skipping Chapter 10 (Radix Sorting) and Chapter 13 (External Sorting). Other Topics;/Sedgewick Chapters covered will necessarily include Elementary Searching Methods (Chapter 14), Hashing (Chapter 16), and Graph Algorithms (Chapter 29, 31, and possibly 32). Optional topics depending on the rhythm of the course will include string searching (Chapter 19) and cryptology (Chapter 23). Mathematical rigor and the curiosity of all involved is enhanced with Rawlins' proofs of the complexity of diverse algorithms, including, for example, esoteric methods like the Jump Search. This helps produce a course that is broad in its coverage and sufficiently rigorous in its approach. As a final course project students are asked to develop programs investigating algorithms of their choice or to investigate course topics that could not be investigated in class, or to investigate certain course topics, like crytology, NP complete algorithms, Turing machines, etc. on their own and make class presentations as well as produce short papers.

Richard T. Close
U. S. Coast Guard Academy

Somewhere along the way in the undergraduate computer science curriculum, "data structures and algorithms" have become "algorithm analysis and computability". That is, the content of this course has tended to become more theoretical as the discipline has matured and the latest curriculum recommendations have appeared. Teaching a course with a large theory component is always a challenge but it seems that this one is especially daunting because it is likely to be the first time that computer science majors come to grips with abstract mathematical concepts. This is somewhat surprising because the assumption is that since CS majors regularly deal with abstract concepts, they can readily understand rigorous mathematical concepts.

Some success has been achieved by including a closed lab to provide an experimental component in this course. The normal algorithms for selection, searching and sorting can be programmed and analyzed experimentally. In addition, there are quite a few animations available for almost every well-known algorithm; now easily found on the internet. Several industrious students have also found it feasible to design and implement their own animations. It also is possible to include several exercises on genetic algorithms, finite state machines and computability. The operation of a closed lab at this level is not trivial and may be unusual but the benefits seem to be worth the effort. Students appreciate the more concrete approach and their depth of understanding seems to be improved.