How Should Data Structures and Algorithms be Taught

Danny Kopec
Senior Lecturer / Associate Professor
Richmond,
The American International University
London,UK
kopec@drk2500.demon.co.uk

Richard Close
Professor
US Coast Guard Academy
New London, CT
(860)444-8622
RClose@cga.uscg.mil

Jim Aman
Associate Professor
Wilmington College
Wilmington, OH
(937)382-6661X517
jaman@infinet.com

ABSTRACT

Data Structures and Algorithms is clearly a very important topic and course in the Computer Science curriculum. It has been taught at several levels by a number of approaches. Should the approach be mathematical, theoretical and abstract or very concrete and "hands on"? Whichever method is used, the ultimate goal is the same: enhancing student comprehension. The panelists discuss three distinct and well-defined approaches.

Keywords

Evaluating Teaching Methods, Computer Science Education Research, Innovative Instructional Methods, Impact of Technology on the Curriculum., Closed Laboratory Experience

1. Danny Kopec, 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 recurrence relationships for 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 (Sedgewick, 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 cyptology, NP complete algorithms, Turing machines, etc. on their own and make class presentations as well as produce short papers.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ITiCSE '99 6/99 Cracow, Poland
© 1999 ACM 1-58113-087-2/99/0005...$5.00

2. Richard 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.

3. Jim Aman, Wilmington College

One semester several years ago the Data Structures class produced an enrollment of only four students. The instructor decided to conduct the course in a seminar format rather than the more traditional lecture/lab style. Since then the course has been offered three times in the seminar format, with a mixture of results. The design of the course requires extensive research, analysis, and testing of the data structure or algorithm class assigned. The department has designated Data Structures a "writing-intensive" so the final report of the student is also a major writing exercise. This report must be distributed to the instructor and all class members in "near-final" form at least one week prior to an assigned presentation date. The presentation is a 2+ hour block in which the student presents the report, explains background, methodology, and other relevant information, and then submits to an open question-and-answer period. The audience is classmates, other students (both lower- and upper-division), faculty from other departments, and alumni. All in all, it is an intense, high-energy period. There are several solid positives and glaring negatives to this approach. First, the pressure of the presentation is (by the students' own evaluations) a growth experience. They learn a great deal about literature searches in a technical area. They also learn that one pass through the writing of a scholarly paper is not enough; the instructor's editing is complete and demanding. From the presentation itself and the Q&A period following it, they learn a lot about themselves and about the importance of maintaining emotional balance under pressure. There is, of course, a very significant danger that a student might be pushed too far by the process. Fortunately, that has never happened. But it is always a possibility. Second, mastery of a broad range of data structures and algorithms is not at all assured in the seminar format. If anything triggers a return to traditional format, it will be this one factor. There is no question each student becomes master of the structure(s) and/or algorithms each studies. No such mastery can be claimed for other structures or algorithms. Over the years students have always been asked to evaluate the course. Their responses have been quite uniform. The seminar/presentation format is difficult, but it is both a welcome relief from lectures and definitely a growth experience.