

Make first year computer science representative and accessible

Jeffrey Forbes
Computer Science Division, University of California, Berkeley
jforbes@cs.berkeley.edu

At an increasing number of schools, the introductory computer science courses are among the most popular. As CS1 takes its place among the canon of undergraduate education, we should compare it with the introductory offerings of other subjects such as physics, economics, and psychology. Introductory computer science remains less representative of the overall major and less accessible to those without previous experience than other popular introductory courses. First year computer science should not just be about programming. It should be about teaching the basics of CS and, more importantly, precise, logical thought processes. Software engineering and the details of programming languages can and should be taught later. I offer a few guiding principles for first year computer science education.

- Less “let’s see if they can hack” and more “let’s make these students into good computer scientists”

There still exists a significant population of students not being reached by current introductory computer science curricula. Many wonder why CS majors rarely reflect the demographics of the overall student body. Computer science differs from other subjects in wide variance in the amount of experience students have in the subject, yet often students are expected to enter as freshmen and be writing programs their first week of class. Those who are unable to handle the class are termed those who “just don’t get it” and are tossed aside. However, success in the major as a whole is generally predicated more on theoretical and analytical foundations rather than ability to quickly code up toy problems. The students who did not come in as the best hackers may turn out to be some of the best computer scientists.

- Survey the field as a whole in illustrating and motivating the basic concepts wherever possible

A course should survey topics such as systems, artificial intelligence, and graphics while explaining essential CS 1 concepts. Alan Biermann proposes such a course for non-majors that touches on a number of areas in computer sciences. The survey is made concrete through relatively simple programming assignments and theoretical exercises [1, 2]. Courses need not be watered down to be more comprehensive though. Techniques such as apprentice learning and the use of libraries allow a student to tackle much more significant problems without having to conceive, write, and debug every last bit of code [3, 4]. A more technical survey course in this style is at offered at Princeton in their first course which surveys the main ideas in computer science while teaching students how to program [5]. The course web pages are listed below:

- Duke CS 01: Computer Science Fundamentals (<http://www.cs.duke.edu/courses/cps001/spring00/>)
- Princeton CS 126: General Computer Science (<http://www.cs.princeton.edu/courses/archive/spr00/cs126/>)

- Basic discrete math needs to be well integrated into the curriculum from the start

The discrete mathematics course is often an afterthought, but it should be well integrated into the overall curricula by being driven by computational tasks and programming assignments. The goal of such a course should be to develop precise, reliable, powerful thinking skills and the ability to state and prove nontrivial facts, in particular about programs. Some example courses are listed below:

- Berkeley CS 70: Discrete Mathematics for CS (<http://www-inst.eecs.berkeley.edu/~cs70>)
- Stanford CS 109: Introduction to Computer Science (<http://cse.stanford.edu/class/cs109/>)
- Carnegie Mellon CS 15-151: Mathematical Foundations of Computer Science (<http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15251/Site/>)
- MIT 6.042J: Mathematics for Computer Science (<http://theory.lcs.mit.edu/classes/6.042>)

In his essay on “Natural talent for computing,” Leon Tabak remarks “I’m beginning to believe that natural talent is a matter of a fresh, patient, and open mind. In our field, I think that the talent of the future may look very different than the talent of the past [6].” Whether this new breed of computer scientists ever materializes may be due to our efforts to make introductory courses more representative of the overall field and accessible to those other than hackers.

References

- [1] Alan W. Biermann. Computer science for the many. *IEEE Computer*, 27(2):62–73, February 1994.
- [2] Alan W. Biermann. *Great Ideas in Computer Science*. MIT Press, second edition, 1997.
- [3] Owen Astrachan and David Reed. The applied apprenticeship approach to CS 1. In *SIGCSE Technical Symposium on Computer Science Education*, Nashville, TN, March 1995.
- [4] Eric S. Roberts. *The Art and Science of C: A Library-Based Approach*. Addison-Wesley, Reading, MA, 1995.
- [5] Robert Sedgewick. *Algorithms in C: Parts 1-4*. Addison-Wesley, third edition, 1998.
- [6] Leon Tabak. Natural talent for computing. *IEEE Computer*, 27(2):120, February 1994.