Do We Need ANY Mathematics in Computer Science Curricula?

Anthony Ralston

Professor Emeritus
Computer Science and Mathematics
SUNY at Buffalo
Buffalo, New York USA
ar9@doc.ic.ac.uk

Do we need math in CS? Silly question, isn't it? Just look at all the computer science curricula ever proposed, most recently CC2001, the ACM-IEEE/CS effort [1]. Mathematics plays a significant role in all of them. So why even pose the question? Well, bear with me.

I take it that almost all graduates of computer science / software engineering (CS/SE) curricula take jobs in the software industry, construed broadly. How many of these *ever* use mathematics in the workplace? Very, very few, I posit. How many of them should be using more mathematics than they do? I will not discuss here this quite different question. A good reference, however, is David Gries' editorial in the 2002 December issue of *inroads* [2].

Therefore, here is my thesis: CS/SE programs[1] spend almost all their time educating (training?) students to be software engineers. Software engineers almost never use mathematics. Therefore, there is no point in requiring mathematics in CS/SE programs.

Have I hooked those of you who do not know me to read on if only just to see how much more nonsense like this I can espouse? Have I hooked those of you who do know me to read on wondering how I am going to extricate myself from a conclusion that you cannot believe that I believe?

[1] Throughout this editorial, I refer only to computer science/software engineering curricula and *not* to computer engineering curricula.

Of course, I do not believe the thesis above. Nevertheless, perhaps in 2005 it is close enough to being believable that it is worth explaining why it is wrong. In doing so I will address two questions:

1. Why should we insist on some mathematics in CS/SE curricula?
2. What should that mathematics be and who should teach it?

Arguments Favoring Mathematics in CS/SE Curricula

One argument I am not going to make is that your graduates will have professional need for any specific bit or bits of mathematics. Yes, *some* practitioners will need *some* mathematics. However, neither the number who will need any mathematics nor the amount of mathematics these will need can possibly justify even one specifically mathematics course in a CS/SE curriculum.

Still, there are good arguments for including some mathematics courses in a CS/SE curriculum. Here are a few.

1. The crucial but unprovable reason for teaching mathematics at any level of education is its effect on the mind. Putting it in another way, studying mathematics does improve the learning ability of students. For CS/SE students, in particular, mathematics is important because the logical thinking inherent in all mathematical thought is so akin to the logical thinking required in all software development.

2. Some CS/SE graduates, but not many, will take jobs where using mathematics is necessary. This number may well increase as and when software development generally becomes more formalized than it is now so that the tools discussed, for example, in *The Science of Programming* [3] start to be more widely used than they are now.
3. Some CS/SE graduates, but relatively few, will go on to graduate school in CS. Some, perhaps much of what they study there will be much more mathematically based than anything in the undergraduate CS/SE curriculum. How much an undergraduate CS/SE curriculum should prepare students for the mathematical demands of graduate school is a topic I'll touch on briefly below.

So given that mathematics should, indeed, play a role in an undergraduate CS/SE curriculum, the question then is: *What mathematics?*

Calculus

How about calculus? Is it still the gold standard? Guzdial and Soloway argued in *inroads* [4] that "Calculus is generally considered part of a liberal education – truly educated people know something significant about calculus". That's nonsense, of course. Many educated people know nothing about calculus and little about mathematics or science generally. Still, I'm on Guzdial's and Soloway's side. Calculus is one of the great inventions of humankind and educated people should know something about it. The trouble is that too many subjects exist about which we could say the same so that only a polymath knows something about all of them.

In any case, I'm interested here not in the liberal education that I would like all computer science programs to be part of but rather about the mathematics appropriate for the professional portion of a computer science program. Many of your students will arrive at college supposedly knowing some calculus from a high school course. However, how many will know anything that can't be done faster and better with a computer algebra system? That is, how many will know anything in depth about calculus? Very few, I think. Therefore, many of them will take calculus as freshmen, often while taking CS1. (Alternatively, as a review of a discrete math book put it recently [5], "Typically, students taking their first course in discrete mathematics have calculus in their background, and sometimes linear algebra and differential equations".)

Does this make any sense at all? It is now almost a quarter of a century since I argued [6] that calculus and discrete mathematics should play coequal roles in the undergraduate mathematics curriculum. (It may not have been, perhaps, one of my more successful forays.) At least it's true, however, that discrete mathematics is now an established course in almost all computer science programs.

Returning to calculus, is there any excuse for requiring CS/SE majors to take a year or even a semester of calculus? I think not. When I was a computer science academic – over ten years ago now – I occasionally amused myself asking colleagues what undergraduate courses we taught that required any calculus at all. Good question! Sometimes one of my colleagues would mumble something about needing Lagrange multipliers or multiple integrals in some course s/he taught as if they ever had mentioned those subjects in the first year of calculus. I doubt things have changed much. There is just no substantial reason to require any calculus at all in an undergraduate CS/SE program.

You might argue that since mathematics departments typically require calculus as a prerequisite for (almost) all intermediate- or upper-level courses, that CS/SE students must take calculus if they are going to take any other mathematics courses at all. However, requiring calculus as a prerequisite for the intermediate- or upper-level mathematics courses that CS/SE students might take (e.g. combinatorics, graph theory, logic) is nonsensical because knowledge of calculus plays essentially no role in such courses. Indeed, the requirement of calculus as a prerequisite for these math courses is just a conditioned reflex; we have always done it that way and so we should continue doing it that way. Yes, calculus does develop the mathematical maturity needed for subsequent mathematics courses, but discrete mathematics does that just as well.

If discrete mathematics played the coequal role with calculus that I once envisaged for it, the prerequisite structure for undergraduate mathematics could and should be realigned to recognize this role for discrete mathematics. CS/SE programs should use the undoubted clout that their student numbers gives them to arm-twist mathematics departments rather than just accede to an historical anomaly.

As an aside, graduate students in computer science may well need to know some calculus as well as other mathematics that few of them get as undergraduates. This provides a good argument for a required course for all computer science graduate students without strong mathematics backgrounds. Such a course might have a title something like "Mathematics for Computer Scientists" (MCS) to assure that all entering graduate students get up to speed quickly in mathematics. I used to teach such a course myself. It never had any calculus in it because then (still?) just about all science undergraduates were dragooned into calculus as their first college mathematics course. Nevertheless, if some students arrive in graduate CS programs knowing little or no calculus, then MCS courses should include some calculus.

As another aside, it used to be argued that students heading toward a CS/SE major should not take discrete mathematics as freshmen because suppose they then decide not to major in CS/SE. Where are they then? In some trouble but then so are almost all science or engineering

students who decide to change their majors in their first year in college. However, in the sciences only physics majors will be in real trouble if they don't take calculus as freshmen. Moreover, since there are so few physics majors compared to CS/SE majors, we need not be overly concerned about this problem relative to physics majors. Students who switch to engineering majors after taking discrete mathematics their first year are a different problem since calculus is a prerequisite to so many engineering courses. However, since discrete mathematics is already important in electrical engineering, at least, and of growing importance in other engineering disciplines, there may be difficulty and some pain; no insuperable problem would exist if a prospective CS/SE major decides to switch to engineering.

Discrete Mathematics

Anyhow, my conclusion is that we should urge prospective CS/SE students to take discrete mathematics as freshmen. This notion is not a new thought for me since I first advocated it twenty years ago [7]. However, how much discrete mathematics should we have and what should be the content of that course?

I argued for a long time that CS students should take a year of discrete mathematics as early as possible and even got my department at SUNY/Buffalo to require this (when I was chair of the department!). I even coauthored a book [8] intended for such a course. Sadly, few CS/SE departments require (or have the fortitude to require) a year of discrete mathematics for undergraduates. Therefore, when my coauthor and I recently did a new edition of this book [9], we bowed to reality and produced a book for a one-semester course (although we cleverly included enough material so that it should be easy to use it for a full-year course!).

I am not going to discuss in any detail what should be in such a course because it does not matter all that much, particularly if it is a year course. Of course, there should be material from combinatorics and graph theory and logic (but reasonable people can argue about whether the emphasis here should be on classical propositional logic or its cousin, calculational logic). Difference equations and discrete probability should play a role, too. Moreover, mathematical induction should be emphasized both because it is quintessentially the most important method of proof in discrete mathematics and because, whatever their background, too few of the students in a discrete math course will ever have been exposed to many mathematical proofs. At some future time there will probably be standard syllabuses for a one-year and one-semester discrete math course for CS/SE students but that time is not here yet even though various reports have proposed syllabuses for such courses [1, 10].

Who Should Teach Discrete Math?

This is a more important question than what the contents of the course should be. Most CS/SE departments have faculty well able to teach discrete math just as most physics departments have faculty who could teach calculus. (I wouldn't push this analogy too far because, while physicists often know the subject matter of calculus well enough to teach it, few have a sufficiently broad mathematical outlook or background to do it well. CS/SE faculty members, on the other hand, often do have the requisite outlook and background although this may not be true for much longer.) In any case, CS/SE faculty members, as their physics counterparts, have quite enough to do teaching courses directly in the discipline without teaching strictly mathematical courses also. Physicists have usually been reasonably happy to have mathematics departments teach their prospective students calculus, particularly because, historically at least, they swung enough weight to make sure that calculus courses taught by mathematics departments contained the material they considered essential.

So, by analogy, shouldn't CS and SE departments be willing to have mathematics departments teach the discrete mathematics courses they want their prospective students to take? Surely, CS/SE departments could pretty much dictate the syllabus for such courses since they are by far the largest client departments for these courses. Moreover, mathematics departments are surely hungry enough for credit hours to be willing, in principle at least, to teach these courses.

Perhaps we should not jump so fast. I may have lost touch and, in any case, all the readers of this will know the situation at your own colleges and universities far better than I do. However, a decade ago it was true that very few faculty in mathematics departments were cognizant enough of computer science and its mathematical needs to teach a good discrete math course aimed at students who would be entering a CS or SE program. Is this still true? (I suspect it is generally true with perhaps the main exceptions being some quality small liberal arts colleges.)

In any case, it should surely be the long-term aim of CS/SE programs to have their strictly mathematics courses taught by mathematics departments. If your mathematics department is not yet capable of doing a good enough job to satisfy you, surely the lure of those credit hours gives you some advantage to persuade your mathematics department to hire the right kind of discrete mathematicians.

Conclusion

Guzdial and Soloway [4] recall Al Perlis's challenge of forty years ago that computer science "should be considered part of a liberal education". If, as I suspect, most academics still don't think CS/SE programs have the flavor of liberal education, one reason may be because,

since the days of Curriculum 68 [11], mathematics has played a steadily decreasing role in CS/SE programs (although the decrease has been less rapid recently than in the decade after Curriculum 68) [12]. However, Perlis was right. One way to meet his challenge is to insure that mathematics does play a proper role in CS/SE programs and, in particular, to do so by breaking the stranglehold of calculus on first and second year college mathematics.

Acknowledgement

I take this opportunity to thank David Hemmendinger for giving me some very useful comments on a draft of this editorial.

References

[1] See <www.computer.org/education/cc2001>
[2] Gries, D. Where Is Programming Methodology These Days? *Inroads 34*, 4 (December 2002), 5-7.
[3] Gries, D. *The Science of Programming*, Springer-Verlag, New York, 1981.
[4] Guzdial, M. and Soloway, E. Computer Science Is More Important Than Calculus: The Challenge of Living Up to Our Potential. *Inroads 35*, 2 (June 2003), 5-8.
[5] Benjamin, A. C. Review of *Discrete Mathematics:Elementary and Beyond* by L. Lovasz, J. Pelikan and K. Vesztergombi. *American Mathematical Monthly 111,* (2004), 635-638.
[6] Ralston, A. Computer Science, Mathematics and the Undergraduate Curricula in Both, *American Mathematical Monthly 88*, (1981), 472-485.
[7] Ralston, A. The First Course in Computer Science Needs a Mathematics Corequisite, *Comm. ACM 27*, (1984), 1002-1005.
[8] Maurer, S. B. and Ralston, A. *Discrete Algorithmic Mathematics (first ed.)* Addison-Wesley, Reading, MA, 1991.
[9] Maurer, S. B. and Ralston, A. *Discrete AlgorithmicMathematics (third ed.)* A K Peters, Wellesley, MA, 2004.
[10] See, for example, www.cs.hope.edu/lacs.
[11] Atchison, W. F. et al. Curriculum 68 – Recommendations for Academic Programs in Computer Science, *Comm. ACM 11*, (1968) 151-197.
[12] Ralston, A. and Shaw, M. Curriculum 78 – Is Computer Science Really that Unmathematical? *Comm. ACM 23*, (1980), 67-70. (Reprinted in: The Carnegie-Mellon Curriculum for Undergraduate Computer Science, edited by Mary Shaw, Springer 1985).

Editor's Note

Anthony "Tony" Ralston is a pioneer in computer science and I am delightful that he accepted my invitation to contribute to *inroads* through this Invited Editorial. Tony has accomplished much for the betterment of computer science through his many works and achievements. A past president of ACM and the American Federation of Information Processing Societies, Tony continually leaves his mark on the profession through his ongoing editions of his *Encyclopedia of Computer Science*. Tony now lives in England. You may find more details at his URL http://www.doc.ic.ac.uk/~ar9/.