

National Computer Education Week Interview

Congress recently passed a resolution to make the week of December 7 National Computer Science Education week in honor of Grace Hopper's birthday. To mark this event, the MIT Press asked several scholars about their thoughts on the current state of the field, as well the current state of education in computer science. You can find out more about the resolution (H.Res.558) [here](#).

Interview with Thomas Cormen

1. What drew you to the field of computer science?

When I took my first programming course in high school, back in 1973, I loved that I could build complex systems that were essentially just concepts. I also saw computer programs as mathematics brought to life. When I took my first computer science course in college, I saw linked lists for the first time. It had never dawned on me that I could make such a structure in a computer. It was when I saw linked lists that I decided I wanted to be a computer scientist.

2. How do you think the field has changed since you first got into it?

Oh, hardly at all, unless you consider progressing from learning on an IBM 1130 with 8 KB of core memory with punchcards, to an IBM 370 running VM/370, which could emulate the ineffable total of 16 MB of memory, to today's laptops with 4 GB and more, today's embedded computers everywhere (my kitchen has at least five), and today's mobile phones. The very concept of a computer is nothing like it was when I started in 1973. A computer used to fill a room; now we walk around with computers in our pockets.

3. Which areas in computer science are you most excited about? Which projects in particular?

I categorize computer science research as capability based or performance based. The former has to do with making computers do things they didn't do before, and the latter is about making computers do things that they can already do, but faster. Although the dividing line is not sharp, it's usually easy to determine which type a particular project is. My research has always been performance based, and I certainly love seeing programs achieve remarkable speedups. But, like most other people, I'm always amazed at the new capabilities that we're developing. I'll pass at listing the projects that most excite me.

4. What kinds of changes (if any) do you think we need to make in computer science education?

Two things make me feel like an old fogey: seeing surfers off Lighthouse Point in Santa Cruz, and thinking about computer science education. Although it's great that students have their own computers now, there was an educational advantage in the old machine room/batch processing model that I grew

up with. Because we got very few runs in during the course of an evening, every run of every program was precious. When my program didn't work, I pored over the listing, really trying to understand what was going on. Today's students tend to randomly morph the program until it works, and when they finally get it working, I'm not sure they understand why. (I will confess that's exactly how I write LaTeX macros.) So there are times that I think, not so facetiously, that we should revert to a model where students get to run a program at most once every 20 minutes. My real answer is not that I think we need big changes in computer science education; it's that I think we need students to have a better understanding of what computer science is. Too many think that it's about making websites, or that being a dedicated gamer gives a leg up in learning how to write good programs. We might have been better off in the old days, when students had no idea what computer science was before taking a course in it, than now, when students harbor misinformed ideas about computer science.