A Case for An Open Source CS Curriculum

Tom Anderson

h/t Aditya Akella, Jeff Chase, Armando Fox, Wyatt Lloyd, Dave Patterson, Geoff Voelker

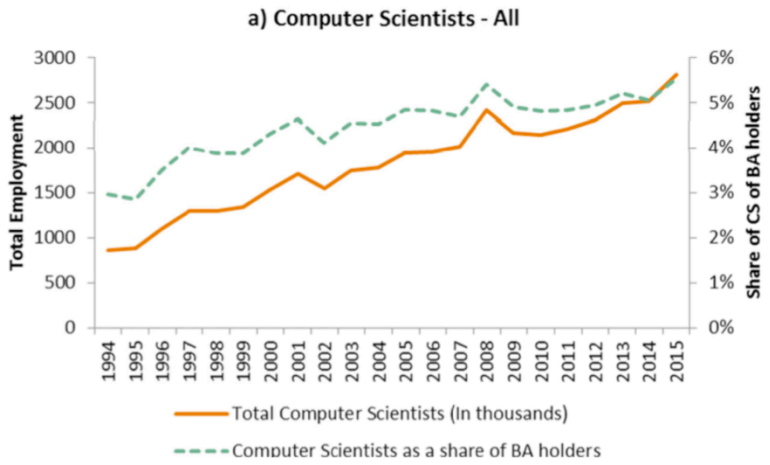
You probably think: an open source CS curriculum would be good! Glad you are working on it. What I'm trying to do is proselytize: to convince a few of you that you want to help in a more significant way. I'm going to make several arguments, some may resonate with you and some might not. That's ok!

consider computer science education as a system. How is it working, and can we make it work better? can we estimate how much benefit making it work better would have to society? I'm going to try to convince you that the potential benefit of doing things differently is enormous.

My own journey to this started with writing an undergraduate operating systems textbook, and seeing the people who were using the book − schools you've mostly never heard of. What could I do to help the students at those schools? The consistent answer I've gotten is to develop a set of assignments that would help teach the material without a huge amount of overhead on the instructor's part.

Given how many people are using my textbook, shouldn't I be spending more time working on project assignments to go along with the book? We need that not only for OS, but many other areas as well.

Is There a Problem?



Bound and Morales, Workforce Trends in Computer Science, 2016
US added an average of ~ 100K new jobs in CS per year since 1994

Source material: National Academy, Assessing and Responding to the Growth of CS Undergraduate Enrollments
https://www.nap.edu/catalog/24926/assessing-and-responding-to-the-growth-of-computer-science-undergraduate-enrollments

Y-axis is total employment in CS jobs, now over 2.5M, having added (with a few small ups and downs) about 100K jobs/year for the last 20 years. 5.5% of all jobs held by college graduates, nationwide, of all ages, is in CS. Total number of bachelors in the workforce has also been growing, so the green line has been growing but not as rapidly.

Note that industry would have hired more if they could − adding 100K/year isn't enough to keep up with demand.

Do We Need That Many CS Grads?

- Supply or Demand?
  - 60K CS bachelors degrees in 2015 in the US
- US added 100K new CS jobs/year (1994-2015)
  - Steady state attrition requires 60K new hires/year
- Estimated 240K CS job openings in US
- Largely not a problem of student interest
  - 15% of entering UW freshmen intend to major in CS
  - Not surprising given the job market for non CS BAs

We don't need everyone to become CS students, but we do need a lot more CS students. We're producing about 60K CS BA degrees year, but industry is hiring about 2.5x that number, between new hires and attrition from the 2.6M currently employed in CS.

But of course even that isn't keeping up with demand − there's a quarter of a million unfilled openings for CS today, or 4 years supply at our current rate.

Some of that is filled by CS masters degrees − there are about 25K CS masters degrees per year (unclear how many of those are additional education for CS BA holders vs. people new to the field (or immigrants). But of course masters degrees are expensive, further pushing CS in the direction of being for the already well off.

Nationally, there is a lot of activity in the "get kids interested in coding" front, and that's great, but at schools that know how to teach CS, we're getting about 10-15% of entering students deciding to major in CS. K-12 outreach can help fill the pipeline, but only if we have a system for those students to learn CS when they get to college. That is a knowledge supply problem: students can't study CS no matter how interested they are, if there's no one to teach them.

Q&A: some of the workforce gap is filled by students who didn't major in CS, or who minor in CS. Employers will train workers if that's the path of least resistance. But I do think the evidence is that there's enormous value to what we are teaching (more on that in a bit). There would also be a benefit to have more people who know some CS without that being their full time job. I don't have a way of quantifying that benefit.

BLS: https://www.bls.gov/emp/tables/emp-by-detailed-occupation.htm

How Big an Opportunity?

- Impact of supply of tech degrees on metro GDP
  - Peri et al., Journal of Labor Economics, 2015
- 1% increase in tech degrees in metro area
  - 7% increase in avg income, for *other* BA holders
  - 3% increase for *non-college* educated workers
- Similar to estimated benefit to DC of Amazon relo
  - $15B per year by 2030
- Plus benefit to students themselves
  - New Berkeley CS grads 2x salaries of non-CS grads
- Nationwide? 1% of workforce is 1.5M
  - 10 years at 150K degrees/year => $1T/year

Study of the economic impact of metro areas between 1990-2010, based on where H1B's went to live. The idea is that it is easier to hire workers of a nationality if there is already a cluster living there, so tech-based H1B's tended to go to cities with existing nationality clusters, setting up a natural experiment − what does increasing the supply of tech workers do to a local economy?

The net result of the study was this: adding 1% of total workers with a CS or engineering training sufficient to get that person hired (since that's the only way you qualify for an H1B) − that increases everyone else's salary by 3-7%. Note that's 1% of total workers, not 1% of tech workers. So in Seattle, we have 2M jobs in our metro area; 1% is roughly 20K jobs, the number Amazon just decided to move to DC and NYC because they found they couldn't hire fast enough in Seattle.

If a 3-7% increase just for having more tech workers seems like a lot, note that the DC area recently estimated the value to the local economy of adding 20K Amazon tech jobs. That is estimated to add $15B/year by 2030; that's roughly in line with the Peri study.
https://www.washingtonpost.com/local/virginia-politics/amazons-va-headquarters-expected-to-have-15-billion-economic-impact-by-2030/2018/12/07/

[Q: Wouldn't this be eaten away by rising housing prices? Yes, to an extent. Peri estimated that about half of the gain went into housing. We'll see later that at a national scale, part of the reason for tech clustering is the lack of tech supply.]

There's also a benefit to students themselves. Berkeley collects data on incomes of students based on major. We should all do that! The result is that a CS degree increases incomes of new Berkeley grads by about 2x. So while you can fill the employment gap with non-CS majors, the market is valuing what we are teaching students.

Higher wages for young people has knock-on effects − they buy houses, cars, furniture, etc. In an economy that has far too little spending by normal people.
https://career.berkeley.edu/Survey/2016Majors

The net is that if we could meet the workforce need for people trained for CS jobs − 150K more BA degrees per year if those degrees were effective at training students to the state of the art, we could add roughly adding half a percent per year to economic growth, adding $1T/year if you were to keep that up for 10 years. That is, roughly, creating a new Google or Apple or Amazon every year.

The Impact of Technology (1980-2014)

You've likely this graph or something like it before. X axis is the percentile on the income distribution, and the y-axis is how much benefit those people have seen from the tech-based economy that started kicking into high gear in 1980. There are vastly different outcomes in terms of who benefits in the US from technological progress. Since 1980 − the dawn of the microprocessor -- the vast majority of US has basically been stalled at 0-1%/year economic growth.

Most of the benefit of our research program, and I'll argue our education program, is to benefit the already very well off. CS is an efficiency mechanism − the net output of what we do is to make certain types of activities more efficient. That's great, efficiency is how you create social wealth − without productivity improvement, we'd all still be subsistence farmers. But most of what we're doing is aiding the return to capital, a benefit to the already wealthy.

For the last 40 years, blue collar earnings have gone up at about half a percent a year. Filling industry's need for engineers, if the students were well-trained, would almost double that for blue collar workers.

Is the public wrong to be skeptical of the value of research? When we say our research produces economic growth, who wins? Certainly hard to argue that the typical american wins, except in that counterfactual sense − without CS research the US economy would be a lot worse off. But we shouldn't think we're immune to being blamed for this. Ask yourself: how many papers at ISCA or SOSP or wherever will actually improve the lives of the median american?

Piketty, Saez, Zucman
https://www.ft.com/content/e494f47e-ce1a-11e7-9dbb-291a884dd8c6

The Heroic Professor?

- Writes papers
- Manages graduate students
- Stays up to date on recent research
- Understands how students learn best
- Determines best way to teach each topic
- Develops great custom-built course projects
- Writes textbooks to share educational knowledge
- Leaps tall buildings

We're all busy of course! So where are we going to find the time? What we're all supposed to do is pretty staggering, and its not too surprising we fall short.

The Reality

- Shortage of faculty for in-demand topics
  - Typical OS teacher took an undergrad OS class
- CS teaching lags technology frontier, badly
  - Almost no one has time for course development
- Assignments and projects aren't widely shared
  - It is a lot of work!
  - And you'll see even more online solution sets
- Even slides are often shared only on Piazza
  - Except among friends, or by textbook authors
- Fewer textbooks are being written

Things are maybe ok at the top few private schools, but both at public flagship schools and at mid-tier schools the reality is pretty far away from ideal. It is hard for anyone to find enough people to teach in-demand topics, and those topics move really rapidly − you need someone at the state of the art, innovating the curriculum on an annual basis, to keep the education up to date. And there aren't that many of us who know all the pieces needed to solve that puzzle.

For my textbook I know everyone who teaches OS in the US - the typical person teaching OS is not an OS researcher. There just aren't enough of those to go around.

Everyone is so overworked that there is very little progress on our collective responsibility − the quality of the education we are providing the nation. Most instructors and teachers at mid-tier schools are far overworked, so there's little time to figure out what students should be learning. Mostly we keep doing what we always have done, but now for more students.

Then if you look at how much sharing of educational materials there is, well, it's a slog. Is there a widely used, well-defined, supported set of assignments someone can use off the shelf for the course you teach? Is that also true for slides, homework assignments, etc.?

In writing my textbook, I trawled the copies of people's slides; this was about 8 years ago, and you could find a lot. Now almost everyone only puts their slides up on piazza.

We used to do this kind of sharing of "how to teach" by writing textbooks, but the textbook market is a disaster zone (due to students using pirated copies) despite the soaring demand for CS education. Where we as a community have put effort into tutorial writing is mostly for the corporate market (where copying doesn't have the same incentive).

Private Schools Aren't Big Enough

Graduating the most CS/CE BAs (US News top 50, first majors)	**Graduating the least CS/CE BAs (US News top 50, first majors)**
1. MIT (31%)	18. Chicago (4.6%)
2. Caltech (26%)	19. Yale (4.5%)
3. Stanford (16%)	20. Hopkins (4.5%)
4. CMU (12%)	21. NYU (4.4%)
5. Princeton (10%)	22. USC (4.2%)
6. Columbia (10%)	23. Northwestern (3.7%)

Private schools in top 50: 3600 total BA degrees (2017)

Heroic professor model works for a few students.

These are the top 50 ranked CS programs, private schools edition. Many are educating enough students. But they are in aggregate small. Even if you triple what the top private schools are doing − kind of hard to do - it won't make much of a dent.

Source: NCES College Navigator/IPEDS

Public Universities to the Rescue?

Public colleges produce the large majority of BAs

- 65% of 4 year degrees in the US
- 73% of 4 year engineering and CS degrees

Tier 1 research schools aren't enough

- Top 50 ranked schools in CS (public and private) produced about 15K CS/CE BAs in 2017
- Still far short of workforce need

Need solutions that work inexpensively, at scale, at non-tier 1 schools

Who can fix this? Before you all shout coursera, we have an existing system of education. Most of the degrees, and the large majority of practical degrees, comes from public universities.

But we need to understand I'm not talking about just the top handful of schools. The top 50 schools (public and private) produced only about 15K BAs last year, in aggregate (NCES).

Historically, low and middle income students were more interested in technical degrees than private school students; this has become inverted over the past 10 years for CS. (cf. the National Academy Study on enrollments). As we'll see in a second, low and middle income students is where the gap is.

If we're to use public schools to address this, any solution needs to be relatively cheap − capable of fitting inside public school tuition, at schools that don't offer PhD degrees.

Public Research Universities

Increasing technical sophistication of economy

Hire faculty who do both research and teaching

- Develop knowledge that pushes the economy forward
- Train the next generation of students in that knowledge

An education open to all who can benefit from it

- a path to levelling income inequality (in theory)

Basic idea of the public research university dates to Lincoln, renewed post-war with Vannevar Bush

The Paradox of Public Research Universities

- The public thinks we are being paid to teach
- Students are funding most of the cost of the flagship state universities

- We think of our job as research
- We spend most of our time doing research
- Most of us try to minimize the amount of time we spend teaching

The model of public research universities − that it benefits society, and students, to teach students at public research schools.

That win-win model looks increasingly like a scam.

At UW, for example, tenure track faculty spend about 10% of their annual time teaching undergraduates (on average). We're a bit of an outlier but not that much of one. Of course there's not much innovation happening if you are only spending a small fraction of your time doing something.

And the courses we do teach tend to be senior/grad courses, leaving most of the heavy lifting for undergrad education to instructors, who really don't have time to do curriculum innovation.

Public Schools

Graduating the most CS/CE BAs (top 30 publics, first majors)	**Graduating the least CS/CE BAs (top 30 publics, first majors)**
1. Georgia Tech (16%)	25. Ohio State (3%)
2. UCSD (11%)	26. Arizona (3%)
3. Michigan (8%)	27. Washington (3%)
4. UC Irvine (8%)	28. Utah (2.9%)
5. UIUC (7%)	29. UC Santa Barbara (2.7%)
6. Minnesota (6%)	30. UCLA (2.6%)

15% of incoming UW want to major in CS. Most public schools are serving many fewer students than the underlying workforce need.

Note that almost all public schools fall far short of meeting incoming student demand. Some like UW and UCLA, are just pathetic.

One could try to get those schools to increase the number they are teaching. And we should. But if we focus on just the top 50 schools, it will not be enough.

[I often get asked the question why UW has 15% entering and only 3% graduating. UW has a post-enrollment application process into majors (not just CS, almost all majors), and a centralized (aka glacial) process for adding slots. We are (somewhat) moving to a model where we admit students directly into the major as freshmen, but that just pushes the problem elsewhere − those 12% we aren't teaching are not going to get taught CS if they go elsewhere (that's why they keep coming to UW despite the long odds).]

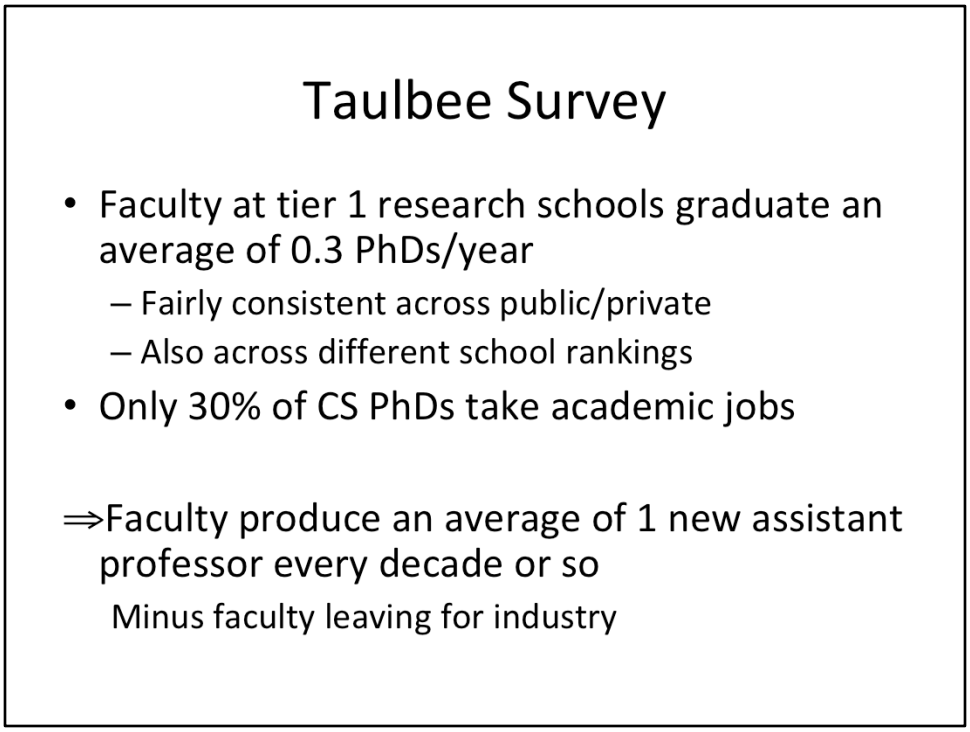
Remember that 5.5% of all BA holders are now working in CS, and that isn't keeping up with demand or future growth. So almost all flagship public schools are producing fewer students than steady state workforce needs − much less the extra 100K/year industry is telling us we need.

Data: NCES/IPEDS, 2016-2017

The CRA Consensus

1. Need more CS BAs
2. Fund CS PhD education via NSF
3. Wait 6 years
4. CS PhDs take faculty jobs
5. They produce more BA degrees
6. Wait 6 years
7. More CS PhDs take faculty jobs

When I started as a junior faculty member, this was the plan for dealing with CS enrollment pressure. This is the plan we're still using. Is it working? Rofl



Taulbee survey 2017

The current system works ok for the top handful of schools, and not for anyone else.

And it won't change anytime soon. Sure, there are people who retire from industry into teaching, but mostly that happens for local schools -- that further clusters knowledge and CS production in places that already have an existing CS industry. E.g., great if you live in the valley and go to SJState, but not so good if you're at Middle Tennessee State.

Rate of new PhD grads going into academics has been on a long term slide downhill.

Can we dramatically increase the rate PhDs stay in academics? Maybe if we doubled everyone's salaries. Maybe we could increase the rate of production of PhDs by doubling the salaries of grad students. Not sure anything would work short of that. We're bidding against Google, Microsoft, Apple, etc., and obviously that's a losing strategy.

On the upside, the people now going into academics often really do care about teaching − its why they are there.

Is NSF the Bottleneck?

- NSF funding for CS research has increased 3.5x since 1994
  - $200M/year in 1994 -> $712M/year in 2017
  - Excluding NSF cyberinfrastructure
- Roughly in line with increase in CS jobs in broader economy
  - 800K in 1994 -> 2.6M in 2015

So yes, none of us has enough research funding. And maybe we could attract more faculty if we needed to spend less time writing grants. But there's been a factor of 3x bump in NSF funding, and a factor of 3x bump in the share of CS jobs in the economy.

Put another way, if you want more research funding, grow the number of CS jobs in the economy. Why is that? Does society think what we're doing is essential, but it can't afford to put more money into it? Or is it that society thinks we're just one priority out of many?

NSF budget: https://www.nsf.gov/about/budget/fy2019/pdf/24_fy2019.pdf
NSF CISE budget over time:
https://www.socialsciencespace.com/2014/03/how-much-nsf-funding-goes-to-social-science/graph/

https://cra.org/crn/2006/01/an_analysis_of_cise_funding_in_fy_2005/

Recap

- Massive underproduction of CS degrees relative to industry need
  - Large missed opportunity for economic growth
- Tier 1 schools producing too few students
  - Even large increases would only make a dent
- Need solutions that work for mid-tier colleges
  - Typically not where research PhDs go to teach
  - But it is where middle and low income students go to learn

The last point, where do middle and low income students go to learn? That is the topic of the next portion of my talk.

Equality of Opportunity Project

Innovative study of anonymized tax records
- Parent and child tax returns, zip codes
- Anonymized college student records

US neighborhoods are segregated by income
- Move a poor child to a wealthy neighborhood
- Outcomes converge at rate of 4%/year

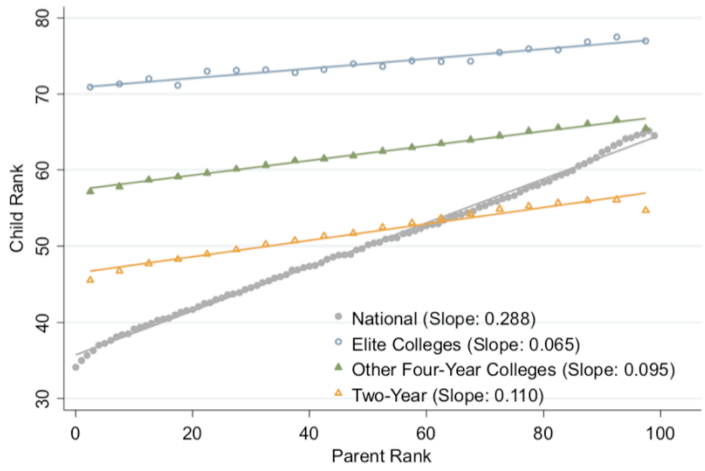
Universities as segregated as neighborhoods
- Few low/middle income students attend tier 1s
- Those who do end up similar to wealthy kids

OK, but at least those of us at public schools are teaching the middle class? Uh, no.

One of the best things you can do to improve outcomes for low income kids is to move them to high income neighborhoods. Or to admit them to high income colleges.

https://opportunityinsights.org
(both data and papers)

College Can Be A Social Leveller

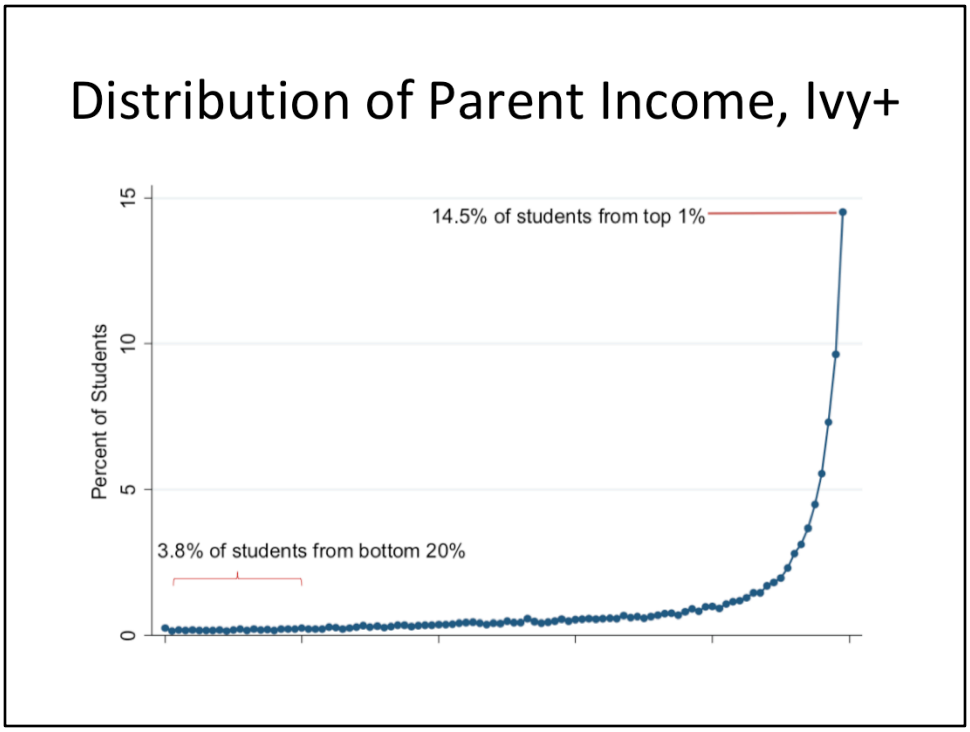


This graph is from Chetty et al., The Role of Colleges in Intergenerational Mobility

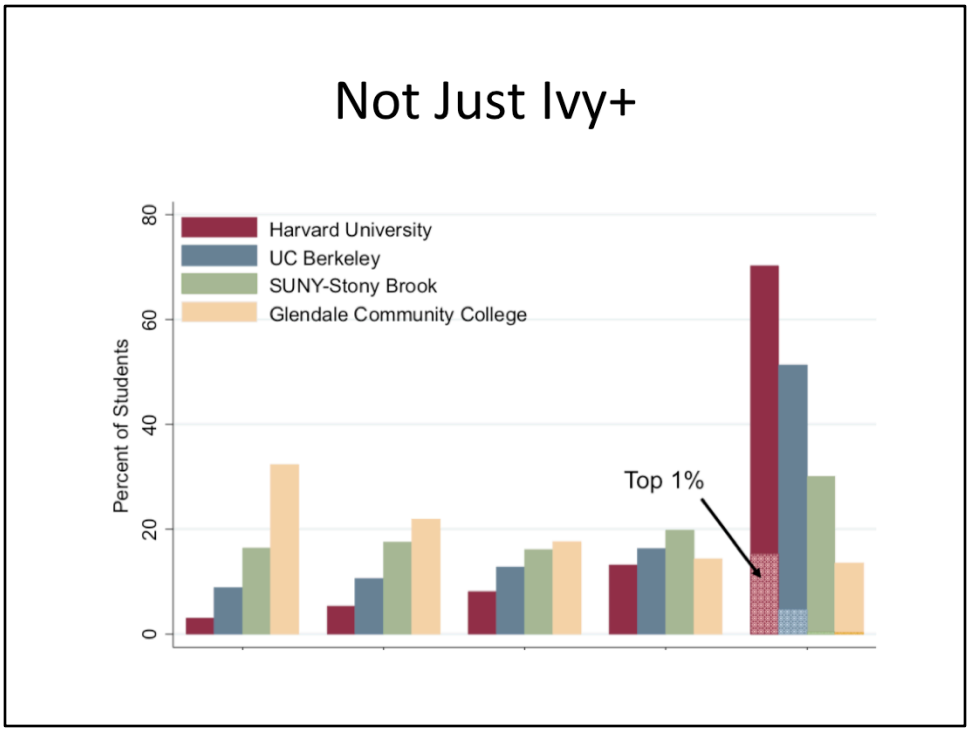
How children do as young adults, in terms of income, as a function of how their parents did. The gray line is all children born in a given year − there's a real problem with social mobility in this country. If your parents are poor, you are likely to be poor as an adult. But colleges do well at leveling − low and middle income students do almost as well as high income students, PROVIDED that they attend the same colleges.

The thing is, poor kids and rich kids don't attend the same colleges. Colleges have as much segregation by income as housing does. How many middle and low income students go to your local elementary school?

If we could improve the outcomes for mid-tier and 2 year schools to be near those students who attend colleges with research programs, we'd do a lot to address income inequality!



This graph is also from the Chetty paper. Sorry it cut off. X axis is parent income, y axis is % of students at the Ivy+ schools (e.g., including MIT) for each parent income level. Not just too few poor students, too few middle income students, too few upper middle class students, at these schools. Massive overrepresentation by the 1%, and beyond that, by the 0.1%.



And this also applies to flagship public schools. Bar graph is for students in each quartile, from poorest to richest, at each of the different schools. Here's Chetty's graph for UC Berkeley, but the other data is online, and for example UW doesn't look that different.

Top CS Research Universities
US students from top 10% / bottom 60%

Public		**Private**	
UW	1.2	Stanford	2.8
Berkeley	1.3	MIT	1.8
UCLA	1.0	Princeton	4.2
UCSD	0.6	CMU	2.6
Michigan	2.9	Cornell	2.5
Texas	1.4	Caltech	4.5
GaTech	1.8		

Data for students born in 1991, includes (most) transfer students
International students at public schools are likely even more skewed

So if we narrow down at the most highly ranked CS schools, and look at what types of students they are teaching. Of course we don't have data on who is in the major at these different schools. But across each campus, public schools teach more low and middle income students than the private schools, but in absolute terms not really that much difference. Uniform access across income levels would have this metric at around 0.15 − even at UCSD you are 4x more likely to come from a well off family than a median family. Data from students born in 1991. Note that any student whose parents make median income gets a free ride at every school on the right.

(Wisconsin and UIUC aren't listed because of issues in how they report their data on branch campuses, but I'll have data on them in a second that shows they are broadly similar)

This is an underestimate on income inequality on campus since it excludes international students! The CS major at many schools requires a high GPA, so it tends to be even more segregated by income, favoring those who start with a stronger background in high school.

CS is the gateway to a good job, and most of those spots are going to people who are kids of parents who are already well off.

Top CS Research Universities
Median Family Income (US only)

Public		**Private**	
UW	$113K	Stanford	$168K
Berkeley	$120K	MIT	$137K
UCLA	$105K	Princeton	$186K
UCSD	$ 82K	CMU	$155K
Michigan	$154K	Cornell	$152K
Texas	$124K	Caltech	$146K
GaTech	$130K		

Data for students born in 1991, includes (most) transfer students
International students at public schools are likely even more skewed

Another way to look at it is median income at each of the schools. The typical student in one of our classes is not exactly middle class, unless you happen to teach at UCSD.

Top Research Universities
Pell Grants, all undergrads

Pell grant threshold is roughly median income (due to ARRA)

Public		**Private**	
UCSD	34%	MIT	17%
UCLA	34%	Cornell	16%
Berkeley	28%	Princeton	16%
Texas	24%	Stanford	15%
UIUC	21%	Caltech	12%
UW	21%	CMU	12%
GaTech	15%		
Michigan	15%		
Wisconsin	13%		

OK, that was for students born in 1991. A lot of schools have made efforts to bring in more low income students, so how is that going?

This is data from the NCES for students currently enrolled at each of the schools (as of 2016-2017). Basically consistent with the other data. Note that pell grant eligibility is complex (depends on parent wealth, income, and # of kids in school), but ARRA dramatically widened the set of students who qualify, so that the threshold is now roughly around 60K/year household income. 20% on pell grants means 4x as many students in the upper half as the lower half, but remember there is skew so most of the upper half is actually in the upper 20%.

(This has Wisconsin and UIUC included, and you can see they are broadly similar to the other publics.)

The NCES reports Pell %'s for both freshman and for all undergrads. Many of the schools on the left admit transfer students from community colleges that serve a more diverse population, but in aggregate that doesn't change the income distribution all that much − e.g., UW graduates about 1/5 of its students as cc transfers, but its Pell percentage is basically the same whether you look only at freshmen, or include all students (including transfers). Part of that is that low income students are more likely to drop out than higher income students. Notable exception is Berkeley: 19% Pell as freshman, 28% Pell overall.

Also note institutional muddying: I've reported #'s for the main campus only, the ones our research reputation is based on. UW has big signs around our campus saying our Pell grant #s are 29%. I don't know how it gets those numbers, but it reports totally different numbers to the Federal government − possibly by including UW's branch campuses, which do serve a more diverse community, but whose students take a different curriculum taught by different faculty, and students do not have the right to transfer to the main campus.

College Choices of Low Income Students (Hoxby and Avery 2013)

- 80% of highly qualified low income students are undermatched in school quality
  - Top 4% of SAT scores, high GPA in high school
  - That is, very strong students
- Most low-income students apply only to colleges within 50 miles of home
  - Going away to college is often not an option
- Need to teach students where they live

We can try to move low income students into high income schools, and we should, but that hasn't worked to date. If anyone has any ideas, great!

The question of how to do this has been studied extensively, as the private schools have a big incentive to look more representative, if only to improve optics. But they haven't found that easy to do. Low income students often have family and community responsibilities, that aren't a good match for moving away to school. And gentrification of our cities means that many low income families live far away from a tier 1 school, don't know that many people who have moved away to school, etc.

My point here is that there are a lot of highly qualified low and middle income students who attend some local school − if we're going to meet workforce needs, we need to teach those students CS at an effective level.

A Few Examples

- University of Central Florida
  - 38% Pell grant; 37% URM
  - Graduated 480 CS+CE majors in 2017 (3.6%)
- Texas A&M Commerce
  - 47% Pell; 43% URM
  - Graduated 42 CS majors (2.5%)
- UMass Boston
  - 40% Pell; 31% URM
  - Graduated 37 CS majors (1.4%)

Some example schools, NCES data. Pell grants primarily go to those in bottom half of the income distribution. All of these #'s are for all enrolled students, e.g., including transfers from community colleges.

Some points:

There is interest in CS in lots of different places around the country; it isn't just the students at the research intensive schools who want to learn CS.

And it is not all large classes. Most places with low and middle income students are relatively small local colleges serving local students.

These are schools that look like America; in my view unless we figure out how to help the teachers who work at these types of schools, we will not make much progress against either racial or income diversity in the tech industry. For example, the numbers for UW: 21% Pell; 11% URM

Don't expect leadership from top down: every major college save one in the Boston area lobbied against the creation of UMass Boston. The institutions where we work may all be non-profits, but they have the ideology that their mission is to put their own interests first. (You can see that in how they dealt with Coursera.)

Fallacy of Elite Projection

Belief that public services should be designed to be attractive to the elite

- World class airports, train stations, universities, …
- Means fewer services get built
- Leaves most citizens behind

Instead, we need solutions that are designed to work for everyone

We ought to ask ourselves: how should we design a system that works for all students? The one we have isn't doing that.

Geographic Dispersion

- Software is eating the world
  - Becoming an essential element of every major enterprise
  - Ex: manufacturing, agriculture, health care management, elections…
- Geographically dispersed economic benefits if we can create a dispersed knowledge base
  - Not everyone will move to the valley

If you look back at economic development over the last thirty years, we've seen the deindustrialization of large regions of the country. As CS is eating the world, CS is becoming essential to every enterprise − e.g., including manufacturing, agriculture, etc. Today, that means companies moving to places like Seattle to be near software talent. But we could make it easier for companies to succeed where they are if we had geographically distributed CS expertise, something that is very difficult to do in a model where the knowledge about CS is clustered in a relatively few schools.

Companies move to where the expertise is. If you are a manufacturing company that wants to hire in CS near your manufacturing plant because that will help you compete, good luck in our current system.

Types of Teaching Knowledge (h/t: CMU Eberly Center)

- How to teach in general
  - Not my expertise!
- Subject knowledge
  - Concentrated at tier 1 schools
  - Which are highly income segregated
- How to teach subject (in context)
  - Also concentrated at tier 1's, but lessons may not apply to all audiences

I am not pushing a type of teaching − flipped classrooms, video instead of lectures, etc. People have strong opinions, I don't.

There are lots of people with domain or subject knowledge, both in industry and academics, but we tend to congregate. This is pretty similar to how the most highly skilled K-12 teachers tend to congregate in those schools with the most engaged parents/students, that is high income neighborhoods.

There's a special type of knowledge that we have historically undervalued − how to teach students a specific subject − what are the types of problems students run into when they try to master a particular subject. Example: in my distributed systems class, I teach cache coherence before I teach Paxos, because I think it helps students learn Paxos better to do it in that order.

This knowledge is also concentrated at tier 1's. Hard for those at a mid-tier school to make much progress at this − they are often too under-resourced to have much time to spend on experimentation. Of course, lessons from tier 1's may not (often don't) generalize, unless you are intentional about it.

The tier 1's do have the resources to make life easier for mid tier schools, but we don't think that's part of our mission. While we publish new ideas to help industry, we don't publish new ideas in how to teach subject matter in context, we don't ask whether those ideas have had influence, and we definitely don't promote based on it. Those are social policy decisions − there's nothing inherent in research universities that says we have to value research influence over educational influence.

Coursera/EdX?

- Successful model for students who are already self-guided learners
  - Wherever they live around the globe
- What about everyone else?
  - Need a human being to help problem solve
  - Why are my students having difficulty with a concept? How can I explain it better to them?
- Not a curriculum
  - Most of the effort at CS1/CS2 and specific skills, partly due to the economic model

OK, this is my personal view. Coursera and EdX have proven really successful for a fraction of the students we need to teach, maybe 20% are self-starters enough to learn on their own. Great! What about everyone else? Need for CS is so intense that we can't afford to discard 80% of the potential learners.

For everyone else, we need a human in the loop.

And of course Coursera isn't a curriculum − steps to move in that direction, but we're a long way off from that.

Flipped Classrooms?

- Using someone else's videos at another institution?
  - Undercuts the authority of the teacher
  - Still need local problem solving and adaptation
  - May work well for CS1 outreach to high schools
- Using someone else's textbook/assignments/ autograding scripts at another institution?
  - This often works really well
  - Fewer bugs, clearer expectations, …

Again my view, the problem is not lecture prep. Feel free to disagree, esp if you have evidence one way or another. The problem with just pushing videos is that it undercuts the authority of the teacher. Why are we paying for tuition at school X if the class is just some other school's class?

And you need the teacher to have authority, to be able to problem solve for students, and to be able to adapt the material to the local context. Where flipping may work best is CS 1 in high schools − since most high schools will struggle to hire/ retain a qualified CS teacher, so there really is no other option.

By contrast, adopting other people's projects is just fine − we've all been doing that for years, especially if it means the course assignments are thoroughly debugged, with clear expectations, clear feedback to students as to whether their solutions work − things that often fall by the wayside where each school needs to develop its own projects.

A teacher who adopts a challenging project developed elsewhere will be seen by students as having high standards, finding the best option, etc. the students will need the teacher to do a good job so that they can learn the material. That is, the normal role of the college professor.

Tailored Instruction

- Controlled experiment at Texas-Austin (Psych 1)
  - Lecture vs. lecture + online quiz about previous lecture
  - Quiz auto-tailored to the student's comprehension, based on previous answers
  - Almost no effect on grades for high income students
  - About a half a grade point difference in GPA for low income students => impact on other classes!
- In the skill set of every instructor
  - Outside of the time budget of almost every instructor

There is stuff that we can do in our own classes that would benefit students, but largely we aren't because we don't have the time. We should be taking the best ideas − the ones we perhaps imagine happens in coursera classes (but often doesn't) and apply them more widely.

Here's one

Often there are large differences based on student background − an intervention can have no effect on students who already have the skill set to succeed in our classes, but would have a large effect on students who don't.

Citation needed for the Austin experiment

SIGCSE

- Isn't there already a conference dedicated to educational issues?
  - Almost exclusively attended by instructors
  - Very little mixing with research faculty
- Lots of innovation for CS 1, CS 2
  - Many competing alternatives
- Not as much progress as you go higher into the curriculum
  - Not for a lack of trying

What is to be done?

- Solutions that are effective at scale
- At schools without research programs
- With the faculty at those schools as allies
- Without spending a lot of money
- And you won't get rich or famous

- Increase number of CS students nationwide
- Improve quality of CS teaching nationwide
- Improve income diversity of CS graduates
- Improve ethnic diversity of CS graduates

What is to be done?

- How do we help ranked schools teach more students and more advanced material?
  - Many already teaching very large classes
  - Lots of time spent treading water
- How do we help mid-tier schools teach more students and more advanced material?
  - Often teach multiple classes/term
  - Lots of time spent treading water

We Have a Lot Already

- For some topics, textbook authors have already put together most of what we need
  - Examples: CS:APP, others
- For some topics, projects in wide use
  - Some even with autograding
- For other topics, there's work to do
  - Especially true in systems, imo
  - It's a large step from local use to global use
- What if we coordinated our efforts?

In giving this talk, seems to resonate most with textbook authors who have been working on these issues individually, or from those who have been developing course projects they'd like to be widely used. Many have worked quite hard to make it easier for instructors to adopt better assignments, but it's a bit of a lonely battle.

In a lot of cases, we're redoing work. Large scale impact for courseware is not whether it gets used by one school, but whether something gets wide adoption. And adoption takes a level of engineering and documentation that is maybe 10x what it takes for someone to get a project to work for their own class.

An Open Source CS Curriculum

- An entire undergraduate curriculum
  - Roughly 20 courses
- Focus on course software, not lecture delivery
  - Every school has teachers
  - Need to enable them, not compete with them
  - Need range of assignment difficulty
- Course in a box
  - Problem sets, programming modules, documentation
  - Automated grading, autotuning of difficulty
- Open source: design for change

Open Source: Linux Model

Architect for extensibility
- Ex: adding a new file system, or a new packet queue scheduler, or a new device driver
- Allows large number of people to contribute

Core architecture team
- defines APIs
- decides which community contributions are adopted into mainline source tree

Anyone can fork and customize
- Ethic of adoption of community contributions

What do I mean by the open source model? Linux is a pretty good example. A development effort far too large for any single group of people to do in their spare time. Instead, architect for change: make it easy to customize and incorporate new contributions

There are other similar examples, eg., in the high performance computing community, software projects that the community helps build and maintain, with some centralized resources to coordinate

A Curriculum

- Hard to make course materials effective in isolation
  - What have students already mastered?
- Need the set to hang together as a group
  - An answer to: what do I need to do to learn CS?
  - Requires coordination
- Enable others to contribute content
  - Ex: MIT has a number of P/F skills courses offered by students for students
  - Loosen idea that only faculty set standards

One of the hardest challenges for textbook authors is what you can assume about what students might or might not know when they start the class; virtually every school does their curriculum differently. That's works ok, but adds about 2x to the effort of writing a book.

Requires multi-faculty coordination, something we do internally at every school, but now needs to be done by a virtual faculty

I gave this talk at Princeton and someone came up and asked what they should do to learn CS (not for the first time). I'd like to have an answer: if you can do the assignments, you'll be able to do CS. We need an answer without gaps.

How Many Courses?

At UW, 20 (quarter) courses taken by a majority of majors

- CS 1
- CS 2
- Web programming
- Discrete math
- Probabilistic reasoning
- Data structures
- Software design and testing
- Machine structures
- Systems programming
- Programming languages
- Database systems
- Algorithms
- Database implementation
- Operating systems
- Distributed systems
- Computer security
- Networks
- AI
- Machine learning
- Computer vision

UW only requires the first seven of these for its major; the rest are electives. One could probably add HCI and advanced software engineering. No harm in having more courses, but this is where we should focus our effort, it is on making it possible for every qualified student, regardless of their college, to be able to master this material.

One could create a similar list for data science.

Case Study: Distributed Systems

- Grew course from 42 to 175 in 3 years
  - TA's entirely self-generated
  - Build a dynamically sharded linearizable and highly available key-value store with multikey transactions
- Oddity distributed systems debugger
- Model checked, autograded assignments
  - Students self-diagnose
- Lectures and labs online
  - And solution sets ☹

So I started a few years ago working in this direction. One of the courses I teach is distributed systems. We took a really aggressive assignment, and asked whether we could scale the class. What we realized we needed was better software development tools, for the students to use to understand if they needed to ask for help. All of the assignments are thoroughly model-checked − that's become standard in industry for this type of software, but for students it means the software will alert the student to when they don't know something.

We let them run the test cases as many times as they like, and that's their grade − no hidden tests, of students trying to figure out what the assignment is.

And on the downside, even though there were only 175 students, there were multiple solution sets online within a few weeks of the end of the course. (We have an internal git repo system so this didn't happen by default − it took explicit action by the students to put their solution sets online for others to view.)

To appear, Eurosys 2019

Community Colleges

- Can we extend this model to non-4 year degrees?
  - Two year program in math, data science, and computing
  - With employers at the other end
- Improve graduation rates
  - Focus teachers on helping students
  - Not on grading, assignment creation, …
- Continuing ed is part of their charter
  - Not just for young adults
- Berkeley is prototyping this
  - ¼ of all community college students are in CA

For more information, contact Armando Fox

What About Copying/Cheating?

- We're losing the battle
  - Posting solutions, and using posted solutions, is endemic
- Students have an exit test: their job interview
  - Enhanced by having a standard curriculum!
  - Copying often triggered by lack of help and unclear expectations
- Reduce salience of project grades?
  - Exams can test whether student mastered the material
- Centralize tools for copy detection
  - Teachers don't have enough time to make this a priority

Let's face it, we've already lost the battle for preventing cheating. An open source curriculum with standard projects will make this worse. So we need to think about how to address this. Not sure there is one right answer, but its pretty clear what we're currently doing isn't working, even for locally developed projects.

Cost

- Developing a 2-week assignment, hosted in the cloud with documentation and autograding
  - A small integer number of person-months of effort
  - Each course needs four or five of these, possibly building on top of each other
- Plus common software infrastructure
- About the cost of an NSF Expedition, in total across all subjects

Next Steps

- Community building
- White paper explaining the effort
- Workshop to organize a proposal
- Contact me if you are interested

Summary

We have an opportunity for a large step improvement in computer science education on a national scale, by taking the effort we are already putting into course software, and organizing it a bit differently