Cornell University

CS Department curriculum reform: "Vectors"

Lillian Lee

Cornell University

The Cornell CS Department, which offers the (same) CS major to students in both the College of Arts and Sciences and the College of Engineering, is in the midst of a multi-year revision of our educational offerings. We began with a several-year build-up of an array of novel introductory non-required courses to attract students to computing --- one example is David Easley and Jon Kleinberg's "Networks" course, which regularly draws over 300 students from all 7 undergraduate colleges at Cornell, representing over 30 different majors. A description of some of these courses can be found in a AAAI symposium paper.[93]

Having worked very hard on introductory "attractor" courses, we have now turned our attention to education within the CS major.

Problem(s) being solved with the new curriculum

How can we balance "core" versus "specialization" in a way that can seamlessly preserve the "product" of the two while our field continues to quickly evolve? "Core" knowledge should be that which is fundamental for all CS undergraduates to know, but deciding what deserves to be "core" can be tricky, especially as the "cutting edge" moves forward at lightning pace.

"Specialization" is, by dint of dichotomy, not "core", but we still want to provide support for students to learn about rich and mature or maturing sub-areas that have developed either within CS or in interactions with other fields. This is no minor consideration, given that students often have very little inkling of how broad, deep, and enabling computer science is.

Unfortunately, framing the debate explicitly as one of "what is core, versus what isn't" can lead to standoffs based on (perfectly reasonable) philosophical differences.

What is the solution

We instead framed our central question as, what are some "*vectors*"(so called to suggest "directions of study"), that we want a CS education to enable our students to be able to pursue, and what knowledge supports these vectors singly or as a whole?

The term "vector" is meant to be evocative. Vectors have a direction (intellectual coherence) and a magnitude (coursework requirements), and need not be even close to orthogonal, but rather can have high inner product (overlap). We thus did not take a "top-down" approach of trying to divide CS up into a relatively few distinct sub-fields, but rather, a "bottom-up" approach, where we can create new vectors on the fly as the field evolves. Thus, we have vectors for traditional fields ("graphics"), newly-emerging fields ("network science"), fields that are quite close to each other ("AI" and "human-language technologies", or "systems" and "security and trustworthy computing"), and collections of knowledge or practice such as "software engineering/code warrior" and a breadth-emphasizing "Renaissance/basis vector" (think "Renaissance person"). (In the latter two cases, one can also think of the "vector" notion as representing the normal to a "cross-cutting plane".) The full set of 12 vectors we currently offer is:

- Renaissance (Basis)
- Artificial Intelligence
- Computational Science and Engineering
- Data-Intensive Computing
- Graphics

[93] http://www.cs.cornell.edu/home/llee/papers/teachai.home.html

- Human-Language Technologies
- Network Science
- Programming Languages
- Security and Trustworthy Systems
- Software Engineering / Code Warrior
- Systems
- Theory

(Again, recall that these represent vectors we can currently reliably offer, not a conception of a partition of CS. We considered other vectors during our study, as well, and we expect to add more pending future faculty hires.)

Our subsequent analysis of what knowledge supports the vast majority of the vectors --- and thus can be deemed "core" --- triggered the following major changes. (Many other changes increasing flexibility were made, too; see the URLs below[94] for more info). We added a requirement that students take a full rigorous upper-level course in probability (we specified a menu of courses across the university, and allow the course to double-count towards other requirements, easing the burden of the addition). This meant that room was made in our early required discrete-math course, since much of the probability material was moved out. We removed the requirement of an automata/computability course. The finite-state-machines material was re-located to the "probability hole" in the required discrete math course. The undecidability material was re-located to the required algorithms course. We also removed the scientific computing requirement.

This makes our core courses beyond the two intro programming courses as follows: discrete math, probability, functional programming + data structures, algorithms, computer architecture, and systems. Furthermore, we added the requirement of completion of at least one vector, and increased flexibility in myriad technical ways. So, in terms of number of semester-long courses required by the major (ignoring requirements of the student's College, which could be either Arts & Sciences or Engineering), we have:

Students must take the following courses

(A) 2 intro CS/programming courses (many get AP credit for the 1st)

(B) 5 core CS courses

(C) 3 CS electives

(D) 5 CS practicum

(E) 3 technical electives (e.g., math, chemistry, psychology)

(F) 3 upper level external specialization courses (coherent program in something other than CS)

(G) 1 free elective (the Colleges have additional free elective requirements)

The courses the students take to fulfill the above or their College requirements must also include a probability course and must fulfill at least one vector. Thus, the probability and vector requirement are implemented as "predicates" on a student's transcript: so that courses can be counted both towards a vector and towards another requirement; this is part of the reason why it is easy for students to complete more than one vector, and for us to allow different vectors to require different numbers of courses.

[94] http://www.cs.cornell.edu/ugrad/InfoMtgPresent_12-2008.pdf
http://www.cs.cornell.edu/ugrad/TownHallPresent_1-2009.pdf
http://www.cs.cornell.edu/ugrad/CSMajorTransition08-09.htm

Success - what can we say so far, either quantitatively or qualitatively?

The new rules were only instituted in Spring 2009, so it is too early to measure impact on enrollments, placement, etc. However, here are some preliminary data of interest.

The CS classes of '10, '11, and '12 could choose between the "old major" and the "vectors major" because they entered Cornell when the "old major" was in place. Having completed most of the old-major requirements, 30% of class of '10 went with the old major; but 70% of '11 chose the vectors one, and while most of class of '12 hasn't affiliated yet, 100% of our currently very small sample have chosen vectors.

Moreover, the students' vector selections across these three classes are surprisingly evenly distributed (except that the Renaissance/basis vector is very popular, as can be expected because we encourage all students to complete it whether or not they choose other vectors as well). Thus, we believe that we are identifying and satisfying the diverse interests of our student body.

In what contexts/situations is this solution applicable

We believe the vector-driven analysis led to much more productive discussions than simply asking, "should course X be required?" We also think a "bottom-up" approach, in which a new vector can be added any time a subfield arises, makes the approach very adaptable to future changes, avoids sometimes irresolvable debates about what is "most important", and allows for smaller fields to be brought to students' attention.