Computational Thinking

Scientia lecture
April 9 2013
Devika Subramanian

We computer scientists are living in a golden era. The use of computation has pervaded every aspect of human endeavor. Computation has had a revolutionary impact on the way we all live, and the way we retrieve and process information to make decisions. Indeed it is hard to imagine how we ever got anything done before the year 2000, without Google, Facebook, and high-speed Internet. You might think that the phenomenal growth in computational tools and devices (the iPhone, iPod, iPad) and their adoption in our everyday lives would lead to an influx of undergraduate students clamoring to major in computer science. Alas, that has not been the case. Indeed, between 2004 and 2007, enrollments in the introductory courses in computer science at Rice, as well as the number of computer science majors plummeted, with 2008 being an all time low. That year, even closer to home for me, I could not get the slightest bit of interest in computer science from my then eleven-year old daughter. As she jauntily put it, she didn't care to be taught by someone who could not even pass her 5 th grade computer class (the passing bar was a typing speed of a certain number of words-per-minute!). That was a low moment to say the least. My reaction (to quote the philosopher Daffy Duck) was "What a revolting development this is!"

Is a computer scientist just a typist or a coder? That would be like calling a biologist a pipetter. Or calling a chalk artist a crayon pusher. Outrageous! I had to explain to my 11 year old and to members of her generation, who we are and what we do. And I had to find an explanation quickly in the language of generation Y.

I was inspired by the Disney movie Ratatouille, released in 2007 -- one of my favorite movies of all time. The hero is a tiny rat named Remy who is a genius recipe designer whose dream is to be a star chef in a Michelin restaurant. Remy's obvious physical challenges (he is a tiny rat) appear to be an insurmountable obstacle to achieving his dream. He teams up with a gangly, garbage boy named Linguini in the kitchen of a famous Parisian chef. Hiding under Linguini's cap, Remy translates his amazingly creative recipes into step-by-step instructions for the unimaginative but physically robust Linguini to execute, and soon the duo of Remy and Linguini become the toast of Paris! In 1998, two Remys named Brin and Page devised an exceedingly clever recipe to organize the billions of documents on the world-wide web. They recruited a silicon Linguini to execute their recipe and voila, the search engine Google was born. This is the essence of computational thinking.

Computational thinkers find a need, a problem to solve, and then devise a creative recipe or algorithm for the problem. By coding their recipes using their engineering skills and outsourcing the actual execution of the recipe to a tireless and reliable silicon beast, computational thinker Remys with their computer Linguinis extend creative problem solving to realms they could never have scaled on their own. So, how did I translate this

vision into a foundational course for computer science?

I confess I was motivated by my daughter's generation in this endeavor – my goal was to transform her generation of avid consumers of the products of computational thinking to inspired innovators and creators of brand new artifacts. With generous seed funding from Microsoft Corporation and with the intellectual and moral support of my program manager John Nordlinger of Microsoft, I developed comp140: computational thinking, a freshman course open to all Rice students.

The first version of the course was fielded in Fall 2008. It is a modular course organized around seven major problems. It emphasizes conceptualization and algorithmic thinking, and teaches students how to mathematically model, decompose and solve problems of scale. Problems are drawn from all disciplines. Here is a sample of questions we tackled in the Fall of 2012. How can you determine your social standing in your college from the Rice Facebook network? How can you train a machine to recognize handwritten letters and digits? Would you believe me if I told you that a single equation you learned in middle school forms the core of a five step recipe to get machines to recognize handwritten digits with accuracies of 85%? Students in my class test their recipe on the same dataset used by the USPS to select their zip code recognition vendors. Can a network analysis of Enron's email corpus reveal the culprits who were ultimately charged? Can you write a recipe to create a new Stephen-Colbert style routine, a sonata in the style of Mozart, or a novel à la Bret Easton Ellis? How does Google Maps find routes between places in a US map? Can you devise a better recipe for route finding? How can you detect if electronically recorded votes in an online election are tampered with?

Each module introduces an interesting problem, all the mathematics needed to model the problem is taught and students generate recipes to solve the problem. Students code their recipes in the Python programming language and can experience the joy of creating computational artifacts that can solve large-scale problems powered by their imaginative recipes. Students learn discrete mathematical models such as network theory, Markov models, probabilistic reasoning, discrete dynamical systems, cryptography, and of course, enough Python to code up their recipe ideas to solve these problems.

The class started in Fall 2008 with a modest 40 students and in Fall 2011, it mushroomed to 120. The course size was capped at 120 thereafter. I am very proud of the fact that in each of the five years I have taught the class, 38% of my students have been women. Not surprisingly, the top students in my course come from every school in our university – architecture, music, visual and dramatic arts, sociology, and engineering. Our major count now stands at an amazing 120 – the highest seen at Rice since the mid 90s.

We often draw a distorted distinction between science- and engineering- minded individuals and the seemingly more "creative types." I hope I have convinced you that computational thinking is indeed an innovative and artistic process. Computational thinkers equipped with programming skills can realize their creative vision and fashion new tools, new apps, and new companies. Facebook, Twitter, Wikipedia, the Google self-driving car, Roomba, are all products of computational thinking. And they are all a decade or less old. Few disciplines can turn ideas into realities so quickly. Few have such

a direct and positive effect on people's everyday lives. Computational thinking gives everyone the power to build something out of almost nothing and then to distribute it quickly all over the world. It is the 21st century medium for creativity. All you need is a laptop and a dream, and you can change the world. That is the promise of computational thinking and the power of computer science.