Troy Vasiga
Lecturer, Cheriton School of Computer Science, University of Waterloo
Chair, Canadian Computing Competition

Teaching computer science: a search for fundamentals

Outline

- Overview
- Defining fundamentals
- Previous attempts at fundamentals
- Fundamental problems about computer science
- Fundamental problems in computer science
- Addressing the fundamentals
- Perceptions
- Making good problems for students

Overview

- This talk will focus on finding fundamental concepts for what computer science is, and how educators can go about focussing on these fundamental concepts to improve the educational experience for their students.

Fundamental

- What defines something as fundamental?
  - Deeply-rooted
  - Essential
  - Basic
- Blame the US Election cycle
  - "We hold these truths to be self-evident…"

Another definition

- Something is fundamental if there is still problems making it a reality
- For example, see US constitution
  - "… that all men are created equal…"
- As teachers, we want to impart the fundamental concepts to our students

Fundamental ideas of computer science: Abstraction

- "Computer science is the science of abstraction" (Aho & Ullman, *Foundations of Computer Science*)
- By abstraction, we mean the ability to ignore irrelevant details and focus on the important details
  - That is, we worry about the fundamental concerns of the problem, rather than extraneous details of the problem at hand

Idea of abstraction

- Plato (The Republic) talks about the ideal form
- When we imagine a "chair", there is an ideal form of "chair" that we are imagining
  - In effect, the details (colour, height) are ignored, but the fundamental essence is in common

Problems of abstraction

- What/when to abstract

  We abstract away
  - the electrons
  - …and the gates
  - …and the bits
  - …and the RAM
  - …and the byte-code
  - …and the operating system

- But not always: depending on the circumstances, we may care about some of these levels (rarely do we care about electrons, though)

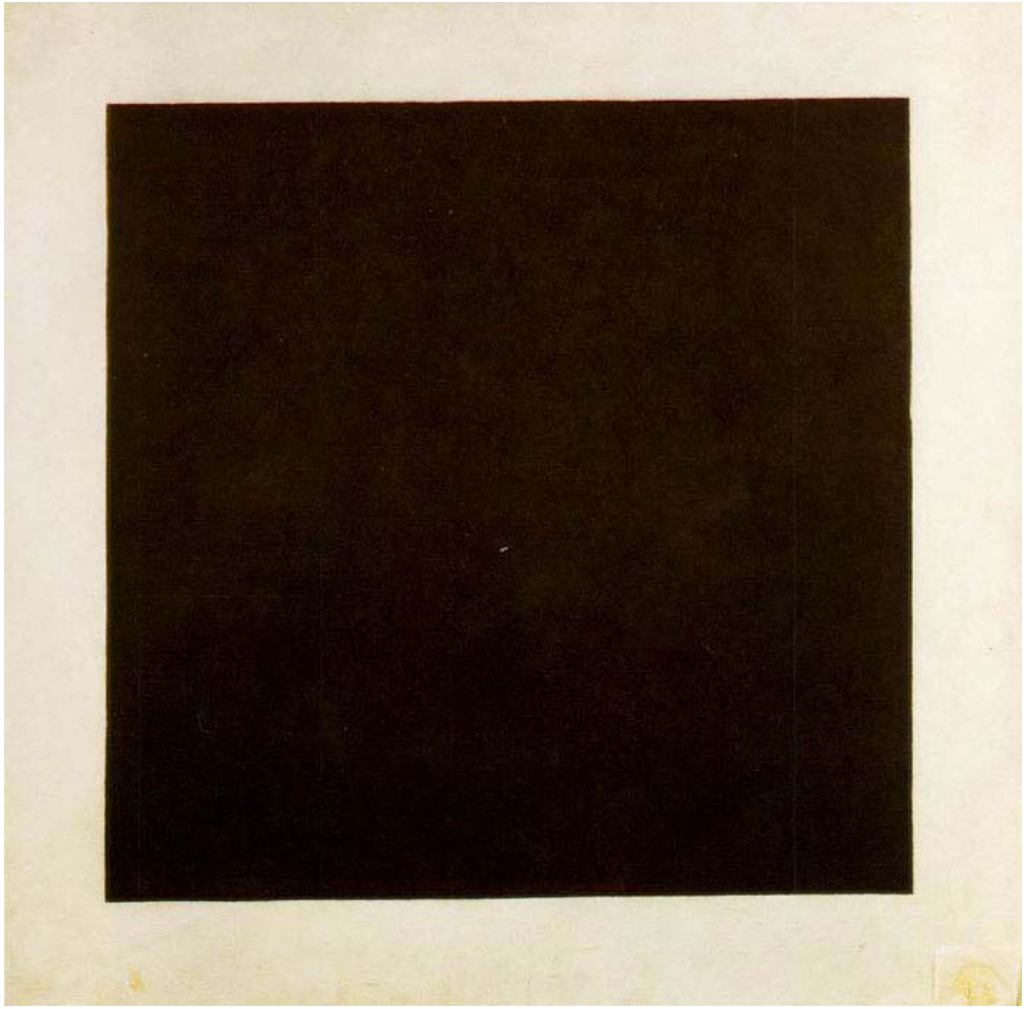
Making abstract realistic

- We need to show that these abstractions have fundamental similarities
  - idea of "flow"
  - idea of "logic"
  - idea of "decisions"
  - idea of "patterns"
  - idea of "process"
  - idea of "algorithms"

Problems of abstraction

- How to abstract
  - Abstraction is an art

Black Square, Kazimir Malevich (1913)



The art of abstraction

- Knuth's The Art of Computer Programming
- "At the end of 1999, these books were named among the best twelve physical-science monographs of the century by *American Scientist*, along with: Dirac on quantum mechanics, Einstein on relativity, Mandelbrot on fractals, Pauling on the chemical bond, Russell and Whitehead on foundations of mathematics, von Neumann and Morgenstern on game theory, Wiener on cybernetics, Woodward and Hoffmann on orbital symmetry, Feynman on quantum electrodynamics, Smith on the search for structure, and Einstein's collected papers. Wow! " (Knuth's webpage)

The art of computer science

- We have to teach students to think like artists
- Example: architecture
  - Students have to read 10 classic works of literature
- Artists become good because they are exposed to a wide-variety of experiences

Solutions to the problems of abstraction

- Need to talk about practical examples of abstraction
  - Need to get away from languages as the only abstraction
  - Need to incrementally introduce abstraction
    - Lean on mathematics!
    - We abstract other concepts (quantity, functions)
    - We need to tie computer science to mathematics early in student's education
  - Need to generate a good range of abstractions at grade appropriate levels

The problem of languages

- Abstraction allows students to be adaptable to new situations
  - Students will not be programming in Java in 30 years
  - Assuming Moore's law applies to number of "new" languages, current secondary school students will be exposed to roughly 100 "languages" in their lifetime

Fundamental issues vs. language issues

- My being a teacher had a decisive influence on making language and systems as simple as possible so that in my teaching, I could concentrate on the essential issues of programming rather than on details of language and notation.
  Niklaus Wirth

Language wars

- Many people tend to look at programming styles and languages like religions: if you belong to one, you cannot belong to others. But this analogy is another fallacy. Niklaus Wirth

Language wars

- Language wars
  - The language becomes the focus, rather than the fundamentals
  - Language become popular for the wrong reasons
    - Marketing
    - Pressure to conform to non-language issues (i.e., operating system tie-ins)
    - "Herd" mentality

What's in a name?

- The most important thing in the programming language is the name. A language will not succeed without a good name. I have recently invented a very good name and now I am looking for a suitable language.
  Donald Knuth

Language wars

- Dijkstra's "How do we tell truths that might hurt?"
  - FORTRAN: "the infantile disorder"
  - "COBOL cripples the mind"
  - "APL is a mistake"
  - "It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration"
  - "Many companies that have made themselves dependent on IBM-equipment (and in doing so have sold their soul to the devil) will collapse under the sheer weight of the unmastered complexity of their data processing systems."

Incrementally introduce abstractions

- Computer science is no more about computers than astronomy is about telescopes.
  Edsger Dijkstra
- In astronomy, this focus is on how planets work, rather than how telescopes work
- We need to focus on logic, control, process, algorithms

Good range of abstractions at appropriate grade levels

- Schwill talks about fundamental ideas in computer science
- "algorithmization"
  - leads to subideas of design paradigms, programming concepts, process, verification and complexity
- "structured dissection"
  - modularization, hierachization, orthogonalization
- "language"
  - syntax and semantics

Good range of abstractions

- Games provide the rich ground of introducing abstraction
- THINK LIKE A CHILD!
- Block games
  - Talk about strategy
  - Process
  - Abstracting away details of the colour of the blocks (for example)

Worldwide attempts at providing good abstractions

- Australian Informatics Competition (http://www.amt.canberra.edu.au)
- Example: **Beetles**
  - In your house is an unusually active colony of beetles. You have been watching these beetles for some time, and you have observed the following facts.
  - Each week, the number of beetles doubles;
  - When the beetles first came into your house, there was an odd number of beetles.
  - Given that there are 544 beetles in your house now, how many weeks ago did the beetles first come?
  - **(a)** 3 **(b)** 5 **(c)** 7 **(d)** 9 **(e)** 11

Another world-wide attempt at good abstractions

- Computer Science Unplugged (http://csunplugged.org)
- Resources that deal with a variety of concepts at a variety of grade levels, requiring only pencil-and-paper
  - binary numbers
  - image representation
  - information hiding
  - cryptography

Computers as the problem

- The problem with computer science is computers
  - they are everywhere: students think they know what they can/should be used for
  - cannot see the forest for the trees
- how to get an appreciation of something?
  - Red Room in MC at UW
    - huge machine
    - fascination
    - freak-show quality

Wonder

- Need to show computers are wondrous (or even, wonderful)
- THINK LIKE A CHILD!
- What makes something wonderful?

Adding wonder

- Topics/tasks/problems should be
  - attractive
    - Add something unusual (graphics, sound, new environment, strange constraint)
  - approachable
    - allow lowest achievers to attempt the problem
  - challenging
    - allow high achievers to go beyond the simple task
  - practical (generalizable or extendible)
    - problems of study should been seen as applicable

Perceptions of computer science

- Need to change public perception of computer science
  - Media
  - Students
  - Parents

Media

- Computer science people are viewed as
  - geeks
  - freaks
  - criminals
- This is something that both teachers and corporations can alter

Students

- Need to get good students interested in computer science
- Need to show them:
  - career is worthwhile
  - show them there are cool things being done
  - show them there are rewarding ($) careers in computer science
  - show them they will need computer science

Curricula problems in computer science

- computer science curricula needs to be more adaptive and flexible
- math curricula is static, but has a deep root of entrenchment
- need to make computer science a fundamental issue
  - I would argue that students need to know how a computer works more than they need to know how to solve x=3x-4.

Parents

- Getting parents of students interested
  - Convince them what computer science is important
  - Convince them they can make money on it
    - Bill Gates
    - Warren Buffet: he thinks algorithmically!!!!
    - Google
    - RIM
  - Debunk myths; explain
    - Discovery channel; Natural Geographic programs

A vision for CS curricula

- Troy's vision
  - Really good curriculum through grade levels
  - Required courses
  - Really good teachers
  - Evangelize how knowing how computers work is fundamental to dealing with the real-world
  - Remove the mystery behind the machine by way of abstracting away some mysterious details occasionally to focus on necessary details

The role of well-designed competitions

- What competitions can test
  - A simple UVA problem:
    - Read print all numbers between a and b.
- what competitions should test
  - Problem solving
    - And problem solving requires abstraction!
- What do competitions look like
  - hand-written (Germany)
  - multiple-choice (Australia, Lithuania)
  - short-time frame competitions (USA/Canada)
  - long-time frame competitions (Netherlands)

Good programming tasks

- What is problem solving?
  - creative ideas
  - intelligent ideas
  - original ideas…
  - …plus prior experience
  - …applied in a new situation

How good problems go bad

1. Take an easy problem and make it harder by increasing the information processing aspect.
2. Take a hard problem and make it easier by removing the problem solving aspects, leaving the information processing or memorization of algorithms.

Problem Solving Distractors

1. Detailed information processing
   - details of programming languages or libraries
   - tedious input/output formatting
   - optimizations that result in constant-time factor improvement

Certainly, some information processing is required. However, it should be as minimal as possible.

Problem Solving Distractors

2. Detailed esoteric knowledge of algorithms
   - implementation of leading-edge algorithms (i.e., recent research papers)
   - memorization of details of complicated algorithms (i.e., implement red-black trees)

Certainly, there is a need to know some algorithms. However, this should be "typical" or "reasonable" algorithms.

Problem Solving Distractors

3. Mystery

- hidden aspects of a task or evaluation criteria

Certainly, some mystery is required. However, too much mystery can be frustrating (and thus, make the task less approachable, appealing, etc.)

One goal, many distractors

- The (student's) goal, given a task, is to solve it.

- We define *solving* as

  being able to communicate the underlying algorithmic process that will consume input in the desired way to produce correct output

Making good questions for students

- Do you understand the problem statement?
- Is there extra information in the problem statement that can be deleted?
- Can some descriptions in the problem tasks be simplified or clarified?
- Do you know how to solve the task?
- What do you consider the components of the task to be?
- If you solved it, was the programming effort tedious? What were the implementation (rather than problem solving) challenges you faced?
- Was your solution fewer than 50 lines of code?
- Describe your thought process in solving the problem. What false starts or incorrect attempts did you encounter while solving the problem?
- Can the input format be simplified?
- Can the output format be simplified?
- Can you imagine problems/circumstances/issues where this task may be generalized to?

Conclusions

- Computer science needs to be presented as something fundamental
- Computer science teaching needs to focus on fundamentals (and stop focussing on non-fundamentals)
- Resources need to be devoted to change perceptions of computer science
- When we ask problems, we should be aware of what it is we are wanting student to do
  - focus on fundamental ideas in problems