Commonsense computing: What do students know before we teach? Episode 1. Sorting

Beth Simon
Univ. of California San Diego

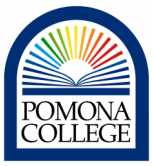
Tzu-Yi Chen
Pomona College

Gary Lewandowski
Xavier University

Robert McCartney
Univ. of Connecticut

Kate Sanders
Rhode Island College

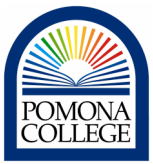
Student performance studies



Student performance studies

- McCracken et al., 2001:

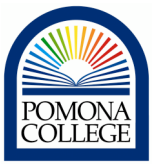
  Can students write code?



Student performance studies

- McCracken et al., 2001:

  Can students write code? **(No)**



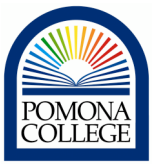
Student performance studies

- McCracken et al., 2001:

  Can students write code? **(No)**

- Lister et al., 2004:

  Can students read and trace code?



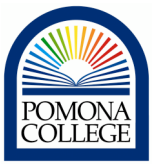
Student performance studies

- McCracken et al., 2001:

  Can students write code? **(No)**

- Lister et al., 2004:

  Can students read and trace code? **(No)**



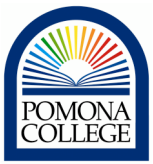
Student performance studies

- McCracken et al., 2001:

  Can students write code? **(No)**
- Lister et al., 2004:

  Can students read and trace code? **(No)**
- Eckerdal et al., 2005:

  Can students design code?



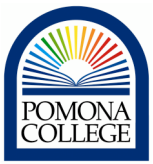
Student performance studies

- McCracken et al., 2001:

  Can students write code? **(No)**
- Lister et al., 2004:

  Can students read and trace code? **(No)**
- Eckerdal et al., 2005:

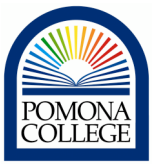
  Can students design code? **(No)**



Student performance studies

- Simon et al., 2006:

  Can students *do anything?*

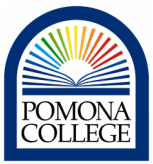


Student performance studies

- Simon et al., 2006:

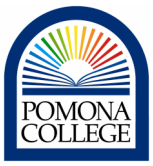
  Can students *do anything?*

YES!



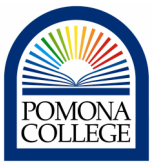
Overview

- Why commonsense knowledge?
- Related work
- Methods (who, what, how)
- Analysis: what can they do?
- Effects of instruction
- Conclusions and future work



Why commonsense knowledge?

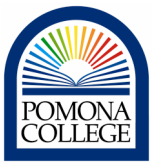
Why sorting?



Specific questions

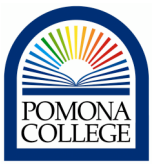
- Can students provide an algorithm?
- How do students approach the task?
- Do students use control structures?

- Can we use these results in teaching?



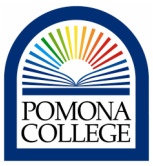
Key observations: entering students

- Most students described a correct algorithm to sort numbers
- Most students used length and individual-digit comparisons to compare numbers
- In iteration, preference given to post-loop tests.



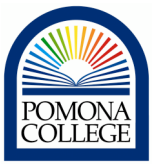
Related work

- Onoroto & Shvaneveldt (differences between naive/beginner/experts)
- Miller (natural language descriptions of programming task)
- Bonar and Soloway (natural language pre-programming knowledge vs. Pascal)



Related work

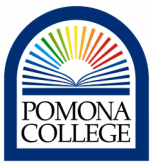
- Ben-David Kolikant (student preconceptions about concurrency from real-life experiences => synch mechanisms)
- Gibson & O'Kelly (algorithmic understanding in precollege and beginners)
- BRACE (Simon et al) (various tasks with beginners: paper folding, giving directions, telephone-directory searching)



Methods (who)

409 subjects:

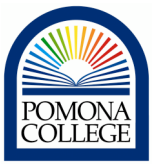
- 118 students in CS 1 (2 institutions)
- 274 students in a non-CS course with no CS experience
- 17 students in CS 1.5, with either CS 1 or other background
- (49 students in CS 1.5 who had been part of 118 above)



Methods (what)

Write a paragraph in complete English sentences describing how you would arrange a set of 10 numbers in "ascending sorted order" – that is, from smallest to largest. You might consider the following list of numbers, but make sure that your paragraph describes how to do it with any 10 numbers.

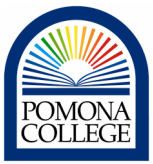
33 14 275 326 213 190 205 4 428 254



Methods (how)

Develop categorization along various dimensions from a subset of 20 CS1's:

- Is it correct? In general, or only for this case?
- What approach? Strings or numbers? What did they focus on?
- Did they use control structures, specifically iteration and conditionals?
- Other content: length, use of example, use of CS "terminology"



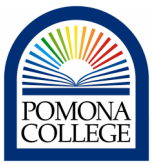
Categorization

Correctness: Does it make sense and "work"? In general, or only for this case?

Approach: Strings or numbers? What did they focus on (main task)?

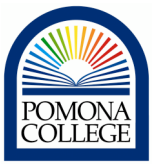
Control structures: Did they use iteration? Did they use conditionals?

Other content: How long? Included example? How much use of CS "terminology"?



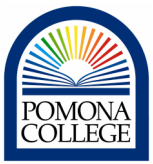
Correctness

	Correct (specific)	Correct (general)
Beginner CS	69%	57%
Naïve non-CS	31%	25%



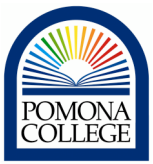
Approach: string vs. number

	Numerical	String	Other
beginner	35%	63%	3%
naive	36%	53%	12%



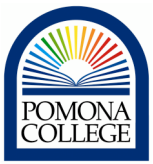
Approach: focus

	Beginner	Naïve non-CS
Digit/length based	50%	47%
Choose	19%	20%
Find smallest	8%	3%
Define	7%	5%
Programming statements	0%	0%
Put number in correct place	8%	14%
Other	9%	10%



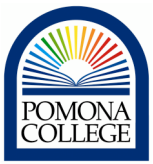
Control structure

	Expresses Iteration	Expresses conditional
Beginner CS	65%	43%
Naïve non-CS	56%	25%



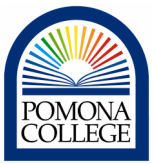
Effect of instruction: CS 1 considered harmful?

	CS 1	CS 1.5 (paired)
Correct (specific)	71%	53%
Correct (general)	57%	45%
Numeric sorts	35%	92%
Length	169.4	74.0
CS terms used	1.61	4.39



CS 1.5 paired vs. different CS 1.5?

	CS 1.5 (paired)	CS 1.5 (other)
Correct (specific)	53%	88%
Correct (general)	45%	76%
Numeric sorts	92%	72%
Length	74.0	184.8
CS terms used	4.39	6.6

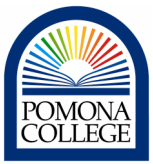


Conclusions

Students can express sorting algorithms, although may be different than instructor

CS students do this better than non-CS, given same experience

CS 1 has some negative effects on performance

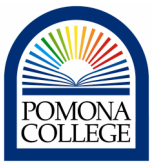


Future work

There are other potential skills to examine that are based in commonsense understandings: troubleshooting; evaluating interfaces; concurrency; discrete probabilities;…

Currently:

- Collecting new data, more varied schools
- Piloting other questions



Thank you!

My collaborators:

Tzu-Yi Chen, Pomona College

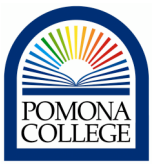
Gary Lewandowski, Xavier University

Kate Sanders, Rhode Island College

Beth Simon, Univ. of California, San Diego

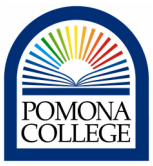
Thank you!

Other collaborators who are currently collecting preconception data.



Thank you!


UNIVERSITY OF CALIFORNIA
SAN DIEGO


POMONA
COLLEGE


XAVIER
UNIVERSITY


UNIVERSITY OF CONNECTICUT
1881


RHODE ISLAND COLLEGE
1854