

# Developing a Formative Assessment of Instruction for the Foundations of Computing Stream

2013/04/24

Steve Wolfman

UBC CS

Acknowledgments: Help, data, and input from *many* faculty and students at UBC.

## Talk notes:

- do people compartmentalize knowledge (a la constructivist learning theory) into "test"/"common sense" belief categories? which will you access? which do you want to access?

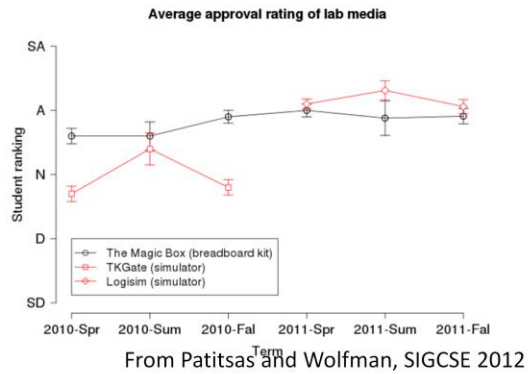
Jim Minstrel: first question about concept (i.e., the "answer"); then reason why question. FACETS work from Jim also.

"Threshold concepts" (from Jennifer): concepts where if you don't get that, it's hard to get the ones after.

Techniques for discovering what students (mis)understand, why they all stink individually, and some examples from our Foundations of Computing sequence

ABSTRACT: Pedagogy is changing quickly in CS and apparently for the better, but how do we know how it affects students' understanding of core concepts and skills? Science education assessment in several disciplines has benefitted from simple, broadly usable tools like the Force Concept Inventory. In this talk, I'll discuss my ongoing work identifying and studying core concepts in UBC's "Foundations of Computing" (FoC) sequence--AKA the required undergraduate "theory" courses--and developing from this a short multiple-choice assessment instrument. I will discuss the ideal end product, describe the methodology I am using, and present preliminary results from student and instructor interviews on FoC topics that are broadly accepted as important.

# Inspiration



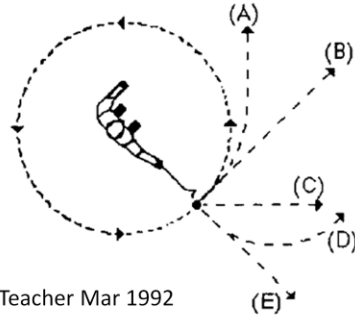
From Mazur, Int'l Newsletter on Physics Ed, Apr 1996

But about a year ago, I came across a series of articles by David Hestenes of Arizona State University(1) that completely and permanently changed my views on teaching. In these articles, Hestenes shows that students enter their first physics course possessing strong beliefs and intuitions about common physical phenomena. These notions are derived from personal experiences and color students' interpretations of material presented in the introductory course. Instruction does very little to change these "common-sense" beliefs.

Mazur: <http://www.physics.umd.edu/icpe/newsletters/n32.htm>

## “Role Model”

4. A heavy ball is attached to a string and swung in a circular path in a horizontal plane as illustrated in the diagram to the right. At the point indicated in the diagram, the string suddenly breaks at the ball. If these events were observed from directly above, indicate the path of the ball after the string breaks.



From Hestenes, Wells, and Swackhamer, Physics Teacher Mar 1992

Hestenes, Wells, and Swackhamer:

[http://cird.unive.it/dspace/bitstream/123456789/317/2/Forced\\_concept\\_inventory.pdf](http://cird.unive.it/dspace/bitstream/123456789/317/2/Forced_concept_inventory.pdf)

Hake: <http://web.mit.edu/rsi/www/2005/misc/minipaper/papers/Hake.pdf>

# Ideal Goal

A concept inventory has the following characteristics:

- It is a reliable, validated assessment instrument.
- It focuses on common student misconceptions.
- It covers a specific domain, but is not a comprehensive instrument (i.e., it is not a final exam).
- It is composed of multiple-choice items.
- It is designed to require at most 30 minutes to complete.
- Its scope may or may not match the scope of the corresponding course. Thus, it could be necessary to develop more than one CI to cover the topics of a course such as CS1.
- It can be administered as a post-test (for example, at the end of the course). Some concept inventories can also be given as pretests, allowing “before and after” comparisons.
- It can be used by instructors to identify aspects of instruction that would benefit from change, to assess the impact of modifications, and to compare pedagogical approaches.

a  
rse sequence.

k at the FCI:

.3 minutes

research on

From Almstrum et al.,  
ITICSE 2006

Cautionary note: Do students have preconceptions, physically-based notions of FoC concepts? Are they clean slates?

# Outline

- Motivation, Inspiration, Role Model, & Ideal Goal
- Idealized Methodology
- Practical Methodology
  - Goals: From Grand to Assessable
  - Exams: A Lesson in Quality
  - Post Hoc Analysis: Mystery of the Student Mind
  - Interviews: Hard-Won Gems +  
Prior Work: Easy-Won Gems 😊
- Discussion

Practical Methodology comment: In theory, theory and practice are the same. In practice, they differ.

# Idealized Methodology

In his overview of STEM CIs, Richardson [44] identified five broad activities that must be carried out in order to construct a concept inventory:

1. Determine the concepts.
2. Study the student learning process for those concepts.
3. Construct multiple-choice items.
4. Administer beta versions of the instrument to determine reliability and validity.
  - (1) Establish topics that are important to teachers (in our case, college or university faculty members).
  - (2) Through selected interviews and observations, identify student thinking about these topics and the various ways it can deviate from expert thinking.
  - (3) Create open-ended survey questions to probe student thinking more broadly in test form.
  - (4) Create a forced answer test that measures student thinking.
  - (5) Carry out validation interviews with both novices and subject experts on the test questions.
  - (6) Administer to classes and run statistical tests on the results.
5. Revise the inventory for validity, and fairness.
  - Validity: From Alms
  - Pilot assessment
  - General use analysis, etc.

Modify items as necessary.

From Adams and Wieman, Int'l Journal of Science Ed 2010

Adams-Wieman\_FASI\_IJSE2010  
iticse-2006-working-group-concept-inventory-why-to-how-to-MUST-READ-p132-  
almstrum

# Outline

- Motivation, Inspiration, Role Model, & Ideal Goal
- Idealized Methodology
- Practical Methodology
  - Goals: From Grand to Assessable
  - Exams: A Lesson in Quality
  - Post Hoc Analysis: Mystery of the Student Mind
  - Interviews: Hard-Won Gems +  
Prior Work: Easy-Won Gems 😊
- Discussion

Practical Methodology comment: In theory, theory and practice are the same. In practice, they differ.



## How do we assess these??

### Grand Goals

What are the key learning goals for the Foundations of Computing stream?

- Recursive/inductive thinking
- Analysis of resource (time, space, energy, ...) costs of solutions
- Formalization/specification of ill-specified problems
- Comfort with "dense" formal descriptions
- Proposal and explanation of multiple solution approaches to a problem
- Meta-cognitive management of the solution process
- Generalizing/abstracting problems/sol'n properties

Interviews (analysed) w/10 of the 14 faculty who taught the "foundations of computing" stream courses in the past three years.

Ask **them** how to assess these. Ask **them** how to assess these in a closed-ended MC format.

Notes on successful/unsuccessful instructor interviews:

The trouble in my least productive interview was that the instructor and I enjoy chewing over big, philosophical issues. That's a fabulous and important pastime, but the point of these interviews was really to get the concrete specifics to form an assessment that can fuel that pastime, not to engage in it.

The best part of the interview was the moment when, looking at an old exam problem, the instructor pointed out a problem students faced with a simple question (determining whether a node is in the left subtree/right subtree of an ancestor or, more likely, determining that it's important and recognizing it). This is a moment fueled by a specific observation of a difficulty students had and a theory as to why in the context of a very concrete problem.

Again, this reinforces the trouble we both had, because we'd often dismiss problems where students did badly by, basically, saying "sure, I'd probably do badly there as well". \*BUT\*, it's not clear that the students were doing badly for the reasons we imagined. We were too focused on the high level and not sufficiently focused on what the data were telling us.

Conversely, in two of my most productive interviews, the instructors had a broad set of example problems on hand that they could use to fuel discussion in a very concrete way. This led to a laundry list of issues tightly coupled with example problems that might probe these issues.

Moral of the story: Try to get people to look through an old exam or two \*before\* you see them!

The LiveScribe pen was extremely useful. I didn't actually heavily use my written notes except to orient myself while relistening, but I /did/ love using the high-speed playback. It made it so I could actually finish reviewing my nine interviews in a single day!

## Getting Assessable Goals Instead

“Think about times when you cringe inside because your students just don’t get something that seems very important to you, and which you expect any expert to get.”

+

Quick review of low-/mid-/high-scoring exams.

Here’s a history of (some of the) prompts used in my second round of instructor interviews.

PB: I think the best way to describe what I'm looking for is places where you sort of feel like you want to cringe because you keep seeing students doing something that feels non-expert, feels naive to you on some important problems.

AH: What are things that students have trouble with that make you cringe? Places where you feel like this is really important, an expert knows how to do this and does it very differently from the way that students are doing it.

EK: What is it that you find cringe-worthy that students have trouble with, where their thinking is very non-expert-like but you'd expect them to get it?

MA: What are the times when you kind of cringe inside because your students just DON'T get something that seems very very important to you, where their thinking is just not expert?

WE: [No opening Q; instead, worked from areas of difficulty on recent exams, which was great.]

## Assessable Goals?

Induction (6/2/-1): “should be able to do .. themselves from scratch without requiring additional input”

Divide & Conquer/Recurrences/Dynamic Programming (4/0/-3): “express the solution to a problem in terms of subproblems”

Logarithmic Tree Height (3/0/-0): “How many times can I give away half my apples before being left with just one?”

...

Mention how much agreement I had.

# Outline

- Motivation, Inspiration, Role Model, & Ideal Goal
- Idealized Methodology
- Practical Methodology
  - Goals: From Grand to Assessable
  - Exams: A Lesson in Quality
  - Post Hoc Analysis: Mystery of the Student Mind
  - Interviews: Hard-Won Gems +  
Prior Work: Easy-Won Gems 😊
- Discussion

Practical Methodology comment: In theory, theory and practice are the same. In practice, they differ.

# Exam Analysis

## (200+ problems, 16 exams)

Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

(b) BST:



**Mean on problem: 32.3%**  
**Problem value: 12/80**  
**Adjusted problem value:  $\frac{1}{16}$**   
**Adjusted exam mean: 70.6%**  
**Adjusted exam stdev: 14.5%**  
**Relative hardness: -2.67 stdevs**  
**Correlation: 37.5%**  
(note: correlation req's per-student scores)

Example problem analysed.

# Outline

- Motivation, Inspiration, Role Model, & Ideal Goal
- Idealized Methodology
- Practical Methodology
  - Goals: From Grand to Assessable
  - Exams: A Lesson in Quality
  - Post Hoc Analysis: Mystery of the Student Mind
  - Interviews: Hard-Won Gems +  
Prior Work: Easy-Won Gems 😊
- Discussion

Practical Methodology comment: In theory, theory and practice are the same. In practice, they differ.

Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

(b) BST:

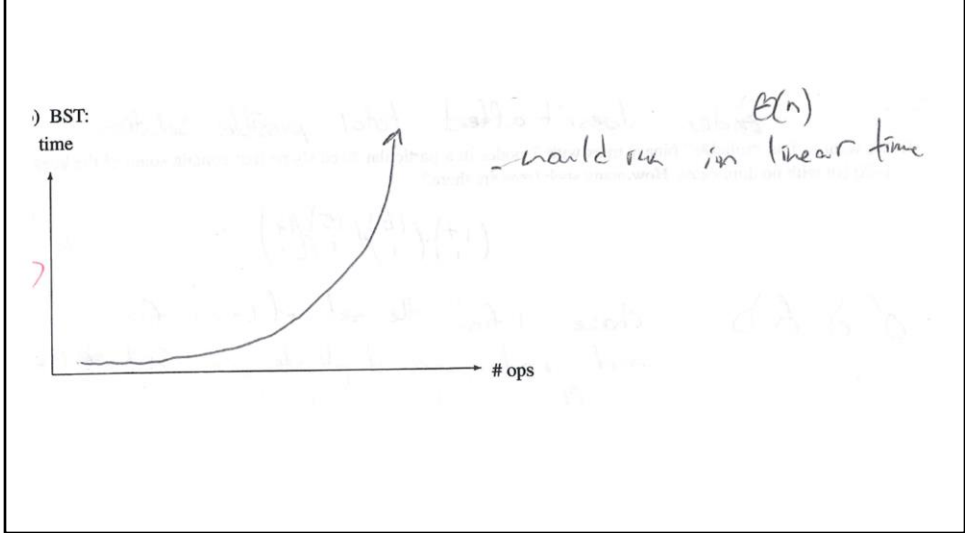
time



# ops

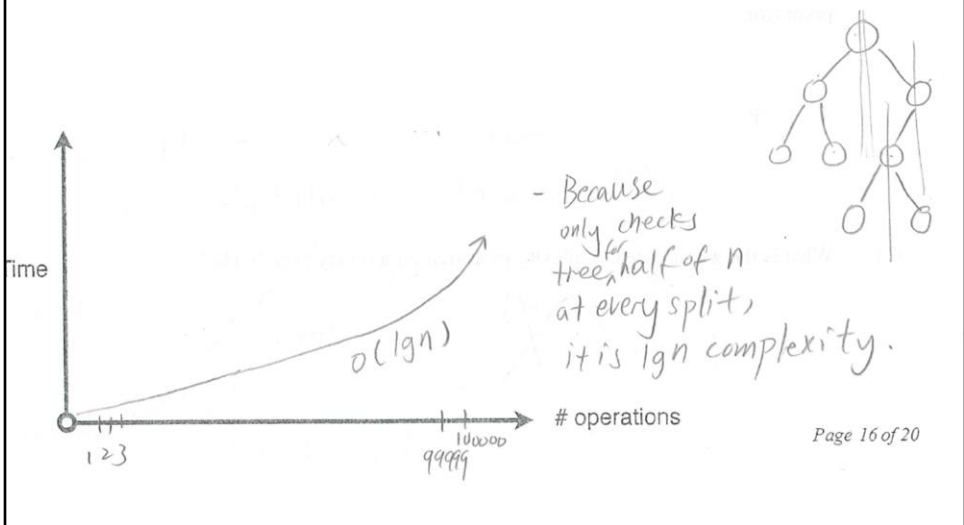


Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

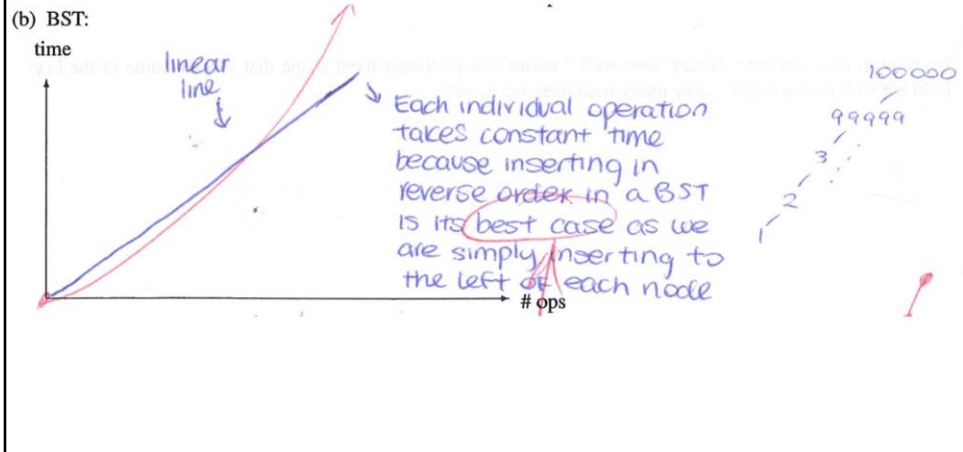


What we might like to see in a solution.

Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).



Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

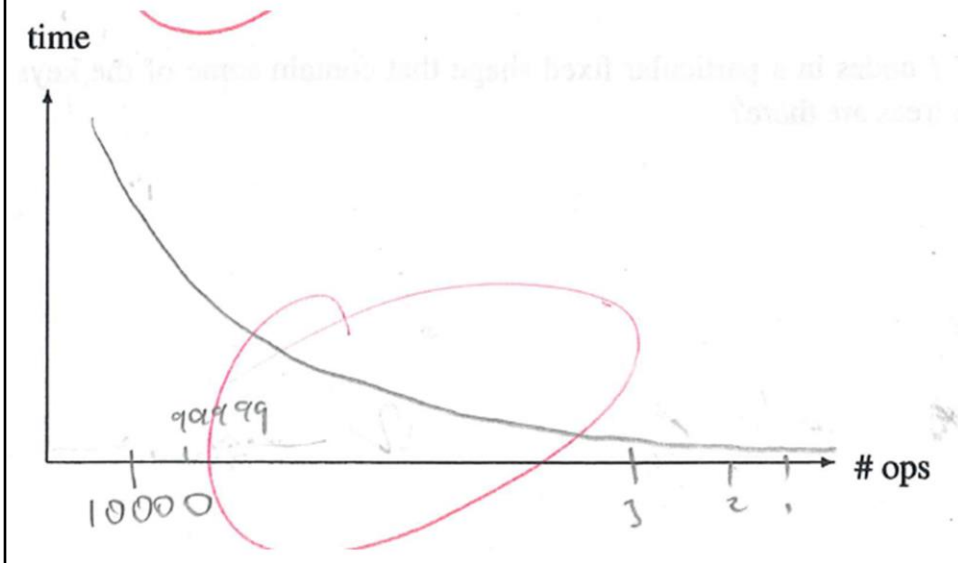


“WISHFUL THINKING” (from an assessment design perspective) and then reality:

Wishful thinking: students don't understand that a structure that is superficially formed like a linked list does not necessarily have the behaviour of a linked list.

Reality: I have no idea if this student instead believed they were working with a Splay Tree, a data structure we didn't teach but did use as a “learn a bit about a data structure on your own” example in an assignment.

Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).



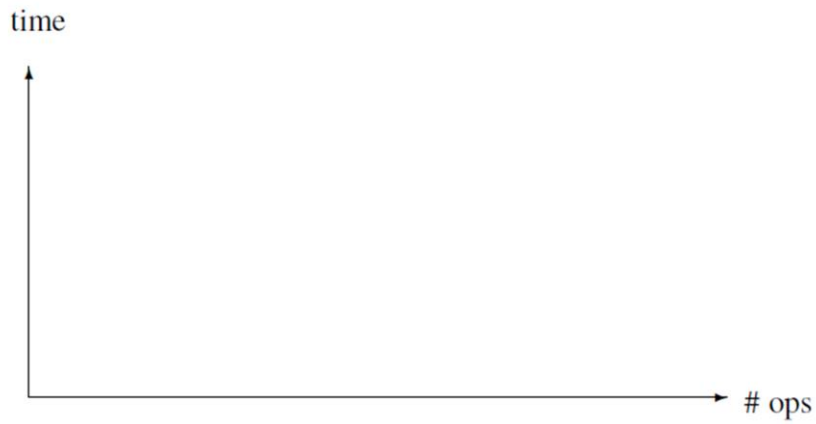
Does this student have the misconception that the “ $n$ ” in our asymptotic analysis is the key value inserted? (Of course, almost none of our students probably have a good understanding that the bottom-line  $n$  is the number of bits used to represent the input.)

Or, perhaps they misread “# ops” to mean “operation number”?

What did the rest of this students' graphs tell us? They were all in “reverse order”. Bizarrely, the unsorted linked list got  $O(n^2)$  as its behaviour; the hash table  $O(n \lg n)$ . Does this student just have no idea? Tempting, but the student wrote this down for *some* reason, and we don't know what.

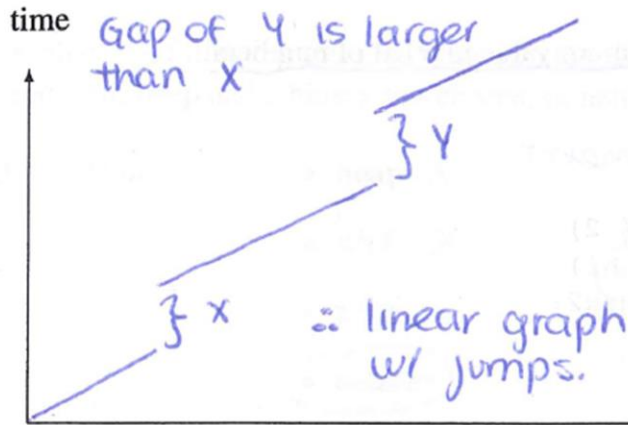
Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

(c) Hash Table:

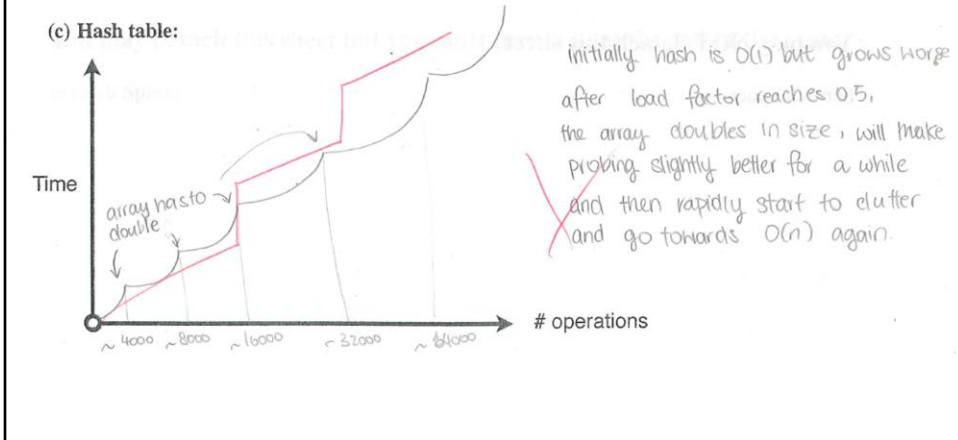


Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

(c) Hash Table:



Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).



This “scallop” is an extremely common pattern for the class that did not spend assignment time viewing graphs of resizing hash table performance.

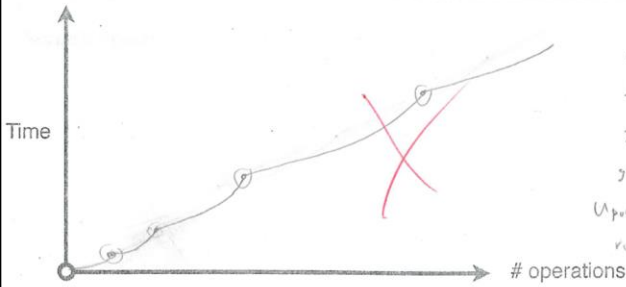
The obvious interpretation is that students believe the hash table resize is quick, and they both fail to perform the reasoning that tells them that collisions do not occur *and* overestimate the likely impact of collisions on practical performance of a half-full hash table with linear probing.

Is this really true?

Does this mean they believe the resize takes no (or very little) time? Does it mean they believe that collisions are quite expensive? Do they not understand that *this particular case* has no collisions? (Ignoring the students who, perhaps, believed they were making general comments due to  $O(\lg n)$  or  $O(n \lg n)$  answers on the previous question, we still get 29% of students exaggerating the cost of collisions (and more who believe collisions have at least some cost) compared to 37% of the group as a whole or 45% of the non-logarithmic crowd.)

# Why "zero cost" resizes? Didn't count as one of the operations? Just overlooked?

(c) Hash table:



(Note  $2^{17} > 100000$ , so table size  
doubles only 4 times)

circled points are where  
the table doubles in size.

Insertion is typically  $O(1)$ , but  
grows worse as load factor increases.

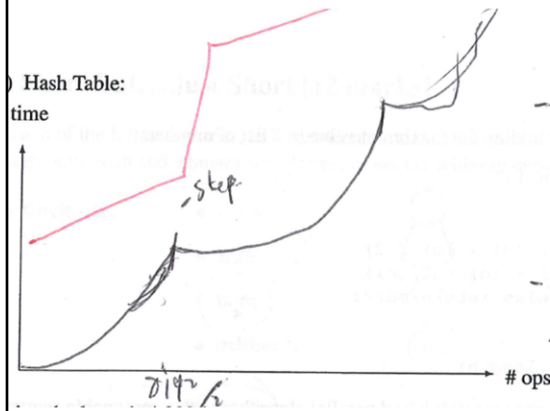
Upon doubling, it is "reset" to  
roughly linear behaviour, but deteriorates  
again as table fills up.



## Why "zero cost" resizes? Internally consistent reason for cheap resize?

X  
as the table size increases <sup>However</sup> and the number decreases, the smaller numbers would have their hash collide with the bigger numbers that was  
→ # operations hashed before with a smaller table size. Therefore, the linear probing would take them <sub>worst case  $O(n)$</sub>  for

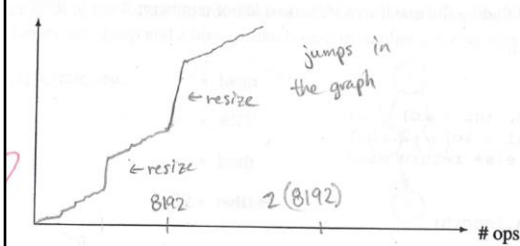
# Why "zero cost" resizes? Buried in exaggerated collision cost?



- $\Theta(n)$
- would run in linear time because it has to check for duplicates on every insertion.
  - step at  $1/4 n$  when table is full, take time to resize.

# Why exaggerated collision cost? Same sorts of questions...

(c) Hash Table:  
time



• small bumps are possible collisions  
from the hashing no collisions here

• since the table doubles in size at  $\frac{1}{2}$  load factor, the "old" table must be copied over to the "new" table  
↳ this jump in the graph should approx. double each time/resize

# Outline

- Motivation, Inspiration, Role Model, & Ideal Goal
- Idealized Methodology
- Practical Methodology
  - Goals: From Grand to Assessable
  - Exams: A Lesson in Quality
  - Post Hoc Analysis: Mystery of the Student Mind
  - Interviews: Hard-Won Gems +  
Prior Work: Easy-Won Gems 😊
- Discussion

Practical Methodology comment: In theory, theory and practice are the same. In practice, they differ.

Imagine you were creating a dance. Here's an algorithm you could use to describe the dance:

```
Dance(n) :  
  if n = 1:  
    walk forward 1 meter  
  else:  
    turn 45 degrees to the left,  
    do the steps in Dance(n - 1),  
    turn 90 degrees to the right,  
    do the steps in Dance(n - 1),  
    turn 45 degrees to the left
```

Let  $M(n)$  be the number of meters of walking you have to do in the dance. So,  $M(1) = 1$ .

Give a formula for  $M(n)$  that is correct for all  $n \geq 2$ . Your formula can and should be in terms of  $M(\cdot)$ . For example the following is a formula for  $M(n)$  in terms of  $M(\cdot)$ , although it is *not* a correct formula for this problem:  $M(n) = M(n/3) - 2$  for  $n \geq 2$ .

Imagine you were creating a dance. Here's an algorithm you could use to describe the dance:

```
Dance (n) :
  if n = 1:
    walk forward 1 meter
  else:
    turn 45 degrees to
    do the steps in Da
    turn 90 degrees to
    do the steps in Da
    turn 45 degrees to
```

Let  $M(n)$  be the number of meters

Give a formula for  $M(n)$  that is correct

$$1 + (1 + 1)$$

$$2. \text{ Dance}(1). \text{ Dance}(1). \\ = 1 + 1 + 1 \\ M(2) = 3.$$

$$3. = \cancel{1 + 1 + 1 + 1}$$

$$M(2) + M(2) + \cancel{M(1)} + 1$$

$$= 3 + 3 +$$

$$M(n) = 2M(n-1) + 1$$

"When it's 3, ... it will be  $M(2) + M(2) + M(1)$  ...  
That would be  $3 + 3 + 1$ ?"

Note: of first 14, 3 suffer the "+1" misconception with 1 giving final answers reflecting this misconception. (Later interviewees have also demonstrated this misconception, including at least one at the 3<sup>rd</sup> year level.)

9 complete quotes:

+ So, I'll add plus 1 for the case when it's 1.


+ When it's 2, it's not 1; so, I'll pass. I'll turn 45 degrees, do the step in Dance.. Dance minus 1 so Dance 1. Turn 90 degrees to the left. But.. do I go.. I will go back /here/. Dance 1, Dance 1. So, that will be 1 and 1. Plus 1 when it's 2.

+ When it's 3, ... it will be  $M(2) + M(2) + M(1)$  ... That would be  $3 + 3 + 1$ ?

+ Oh! OK. So it will be  $M(n)$  will be  $2M(n-1) + 1$ .

“Starting at 4, you.. Do 4, turn right, do 3, turn right, do 2, turn right, walk forward 1 m, turn left. ... So only at one point will I do 1, that's when you walk forward. So  $M(n) = 1$ .”

4 r 3 r 2 r 1 L

  $M(n) = 1$

“If I'm correct, it would always be reduced to 1 ultimately. Since it's just Dance(n-1) and then it's just turning not really moving... and that wouldn't affect .... the number of metres that I've walked.”

$M(n) = (n(0) + 1 \cdot )$  for  $n \geq 2$

Note: of first 14, 5 suffer the loop misconception with 2 giving final answers reflecting this misconception.

(Later interviewees have also demonstrated this misconception, including at least one at the 2<sup>nd</sup> year level.)

"kind of looks like  $n^2$  because you're getting two recursive calls"

$$n^2$$

$M(n) = \# \text{ m walking for } n \geq 2.$

Let's. Since  $n \neq 1$

$$M(2) = \text{Dance}(2-1) = \text{Dance}(1)$$

$$M(n) = M(n-1) + 1.$$



# Kahney, CHI 1983

SOLUTION-1:

```
TO INFECT /X/  
1 NOTE /X/ HAS FLU  
2 CHECK /X/ KISSES ?  
2A If Present: INFECT * ; EXIT  
2B If Absent: EXIT  
DONE
```

SOLUTION-2:

```
TO INFECT /X/  
1 CHECK /X/ KISSES ?  
1A If Present: INFECT * ; CONTINUE  
1B If Absent: CONTINUE  
2 NOTE /X/ HAS FLU  
DONE
```

paragraph. Under the Loop model, however, a programmer would argue that the first Solution would be okay, but not the second. In

“Starting at 4, you.. Do 4, turn right, do 3, turn right, do 2, turn right, walk forward 1 m, turn left. ... So only at one point will I do 1, that's when you walk forward. So  $M(n) = 1$ .”

“If I'm correct, it would always be reduced to 1 ultimately. Since it's just  $Dance(n-1)$  and then it's just turning not really moving... and that wouldn't affect .... the number of metres that I've walked.”

# Götschi, Sanders, and Galpin, SIGCSE 2003

**Step model (S):** Students with these models have no concept of any recursive flow of control. With this model, students simply evaluate the IF-THEN-ELSE and execute either the recursive condition once (one-step), both the recursive condition and the base case (two-step).

BUT... at 2<sup>nd</sup> and 3<sup>rd</sup> year level?

Consider:

When it's 2, it's not 1; so, I'll pass [the if branch]. I'll turn 45 degrees, do the step in ... Dance(1). Turn 90 degrees to the left. But ... I will go back here.

$$\begin{aligned} 2 & \text{ Dance}(1), \text{ Dance}(1) \\ & = 1 + 1 + 1 \\ & \text{M}(2) = 3 \end{aligned}$$

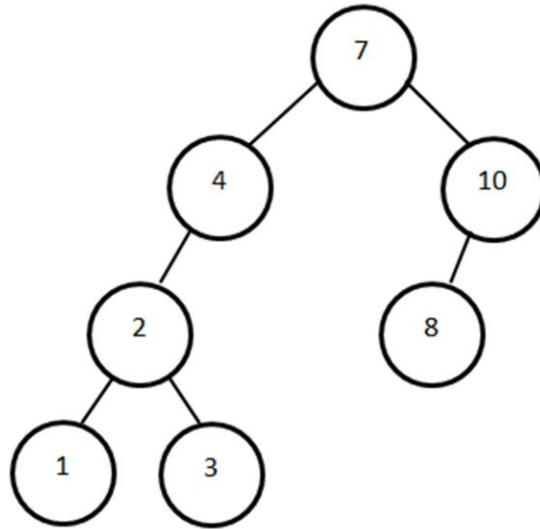
$$1 + (1 + 1)$$

$$M(n) = 2M(n-1) + 1$$

"When it's 3, ... it will be  $M(2) + M(2) + M(1)$  ...  
That would be  $3 + 3 + 1$ ?"

Note the value of a FASI that can be used to track student "trajectory" through responses!

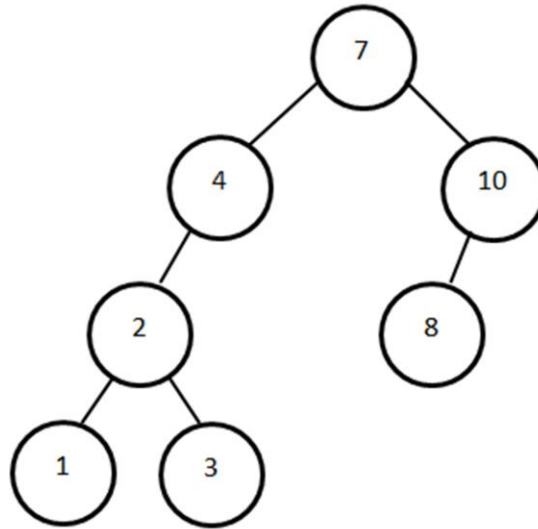
We often draw diagrams of binary search trees like this one:



We have shown the keys but not the values in this tree. Where are the values?

Note: **where** not **what**.

We often draw diagrams of binary search trees like this one:

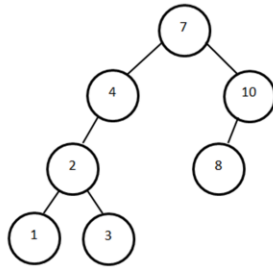


**Exam analysis lesson: communicating clearly is hard.**

We have shown the keys but not the values in this tree. **Where** are the values?

Note: **where** not **what**.

We often draw diagrams of binary search trees like this one:



We have shown the keys but not the values in this tree. Where are the values?

*"the values are pointed to or stored in the same node object"*

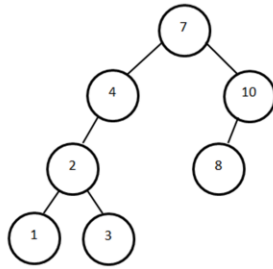
*"The value [are] stored in these bubbles"*

*"the values would reside in memory or on disk"*

*"So the values are.. in the nodes"*

What students say who have expert-like conceptions.

We often draw diagrams of binary search trees like this one:



“the values should be 1, 2, 3, 4, 5, so they're index values right.. 6, 7, 8.. um.. there should be index values”

“it could just be an array ... and ... the keys could just be the indices of a giant array”

We have shown the keys but not the values in this tree. Where are the values?

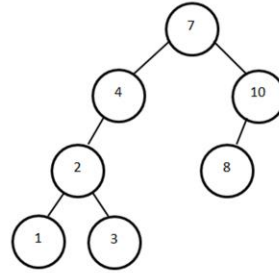
“[the key is] a lookup for the value”

“the values are being represented in the tree by the keys. So knowing ... the key like unlocks ... what the value is”

“the values would actually be in the leafs”

What students say who have expert-like conceptions.

We often draw Binary Search Trees (BSTs) like this, showing the keys but not the values:

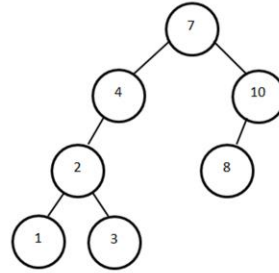


The keys in this BST are numbers; assume that the values are as well.

Where are the values in such a BST?  
Choose the **best** answer.

- (a) The values are stored in the same node as the keys.
- (b) The values are at the leaves.
- (c) The values are pointed to from the same node as the keys.
- (d) The keys are indices into an array that stores the values.
- (e) The keys point to the values.
- (f) The values are 1 (for the node labeled 7), 2 (for the node labeled 4), 3 (for the node labeled 10), 4 (for the node labeled 2), and so on.
- (g) Not enough information to tell.

We often draw Binary Search Trees (BSTs) like this, showing the keys but not the values:



The keys in this BST are numbers; **this time, assume that the values are *images*.**

Where are the values in such a BST?  
Choose the **best** answer.

- (a) The values are stored in the same node as the keys.
- (b) The values are at the leaves.
- (c) The values are pointed to from the same node as the keys.
- (d) The keys are indices into an array that stores the values.
- (e) The keys point to the values.
- (f) The values are 1 (for the node labeled 7), 2 (for the node labeled 4), 3 (for the node labeled 10), 4 (for the node labeled 2), and so on.
- (g) Not enough information to tell.



# Danielsiek et al. SIGCSE 2012

The central new insight with respect to heaps is that **students** – even though they seem to have a rather good passive knowledge of the formal definition of a heap – **tend to conflate heaps with binary search trees**. Since the vast

A: *A heap is, er [laughs], a heap is a tree with an ordering and, er, that's hard to explain. Er, a binary tree.*

T: *Let's talk about binary trees, shall we?*

A: *Binary trees are trees, all of them, which have, have two children, that's indicated by the word binary already, two, er, children, or two child nodes, at most that is, we could also have null. Er, and the child node, the one that's stuck to the left, is smaller than the the node itself and the one, the one stuck on the right has a bigger value than the node itself and that's than kept up recursively and then we get, if we keep hanging on to that thought, far, far, far down on the left the smallest one, the smallest element; far, far, far on the bottom on the right is the biggest element.*

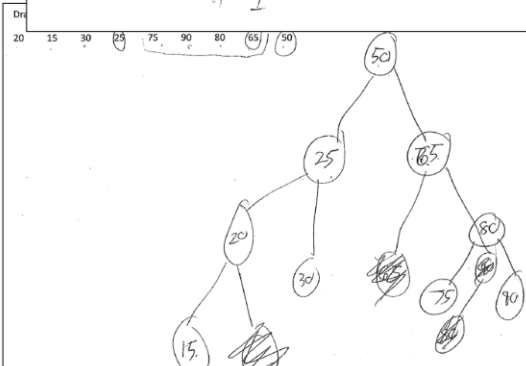
B: *Yes, the heap was like this, er. We put the first number into the root and then we take the second number and we check whether it's bigger, er, than the root. If it's bigger, we write the number down on the right side, and if its smaller, then we write it down on the left side. And then we take the next number and do the same again starting with the root, we check whether it's bigger. If it's smaller, we go to the left again. And then we take a look at the next. If this one is smaller again then we will go to the left again of this one and to the right if its bigger.*

# Heap/BST Confusion? Never hinted at by my faculty i'views...

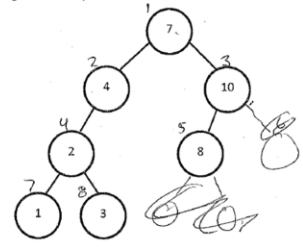
Draw a binary search tree whose keys printed in post-order traversal are:

20 15 30 25 75 90 80 65 50

A  
B C  
D E F G  
H I



We often draw diagrams of binary search trees like this one:



We have shown the keys but not the values in this tree. Where are the values?

In answer to "Why isn't this a BST?"

"it's because ... the right only has depth 1, while the left has depth 3. ... BSTs should have both sides equal depth. Is that a heap? It doesn't matter."

The post-order traversal responses are (again) different students.

Ref to sigcse-2012-detecting-understanding-ds-algo-miscons-danielsiek, sigcse-2013-algo-ds-miscon-instrument-follow-up

# Outline

- Motivation, Inspiration, Role Model, & Ideal Goal
- Idealized Methodology
- Practical Methodology
  - Goals: From Grand to Assessable
  - Exams: A Lesson in Quality
  - Post Hoc Analysis: Mystery of the Student Mind
  - Interviews: Hard-Won Gems +  
Prior Work: Easy-Won Gems 😊
- Discussion

Practical Methodology comment: In theory, theory and practice are the same. In practice, they differ.

# Extra Slides

## Exam Analysis

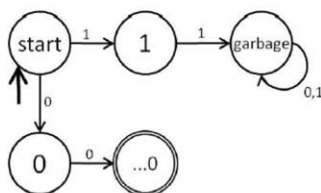
- For each exam: average and standard deviation if each problem on the exam were worth the same amount
- For each question:
  - average on the question,
  - correlation to unweighted exam score
  - “contextual hardness” (problem mean in # of exam stddevs away from exam mean)

Per-Q exam analysis on 15 final exams (200+ Qs) from 121/221/320 (including precis of each question, correlation to unweighted average, std normal "means" for each Q).

Sample exam analysis on batches of ~10 exams for several crucial questions

Unweighted exam score de-emphasizes problem's value; better would be correlation to rest of exam, easy to compute but seemed unnecessary.

**For Questions 9–11:** You are building a DFA to recognize the language: “strings of 0s and 1s with at least two digits that do *not* contain two consecutive 1s”. So far, you have the following, which is correct but incomplete:



An arc labeled 0 will have to be drawn from the state labeled “1”. Which of these *could* that arc legitimately lead to? There may be multiple correct answers. Circle **ALL** that apply! [2 marks]

- (a) A new accepting state.
- (b) A new rejecting state.
- (c) The start state.
- (d) The state labeled “...0”.
- (e) The state labeled “garbage”.

Besides the two discussed in the previous problems, how many *more* arcs still need to be drawn from the *existing* states in the diagram? [1 mark]

- (a) No more arcs need to be drawn from the existing states.
- (b) One more arc needs to be drawn from the existing states.
- (c) Two more arcs need to be drawn from the existing states.
- (d) Three more arcs need to be drawn from the existing states.
- (e) Four more arcs need to be drawn from the existing states.

The question asks: "An arc labeled 0 will have to be drawn from the state labeled "1". Which of these /could/ that arc legitimately lead to? There may be multiple correct answers. Circle **\*ALL\*** that apply.

- A new accepting state.
- A new rejecting state.
- The start state.
- The state labeled "...0".
- The state labeled "garbage".

The correct answers are "a new accepting state" and "the state labeled "...0"". The answer **MUST** be accepting, which is enough to eliminate all but these. To support these, students would need to recognize that "merging" into "...0" is OK (because starting 00 or 10 will make no difference to whether a longer string accepts or rejects). Additionally, they must recognize that it's OK to "duplicate" portions of a DFA, just in terms of correctness.

On “add arrow”, of the 5 students to whom it’s posed (but note that at least one student mentioned that they seemed not to be allowed to add extra states):

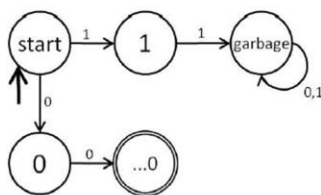
- 12: adds but retracts -> 0 arc; adds -> ...0 arc; states no other arcs possible

- 11: adds only  $\rightarrow \dots 0$  arc
- 9: adds  $\rightarrow 0$  and  $\rightarrow \dots 0$  arcs
- 8: doesn't really understand Q (but adds  $\rightarrow 0$  arc)
- 7: adds  $\rightarrow 0$  and  $\rightarrow \dots 0$  arcs

Of 6 students to whom the “# of arrows” question is posed:

- 7 misunderstands Q (tries to correctly complete DFA), finally sees real Q and responds after that point w/in 30s
- 8 unfamiliar w/concept (but also tries to correctly complete DFA); never really settles into problem
- 9 partially misunderstands Q (approaching as if correctly complete DFA), but settles down to 4
- 10 doesn't know what DFAs are
- 11 even after rephrasing to elim stated purpose of DFA, distracted by “correctness”, gives 4 as answer
- 12 20s after remembering def'n of “legal”, finishes w/4 as answer

For Questions 9–11: You are building a DFA to recognize the language: “strings of 0s and 1s with at least two digits that do *not* contain two consecutive 1s”. So far, you have the following, which is correct but incomplete:



This was a bad question!

(Lots of them are, as it turns out!)

Besides the two discussed in the previous problems, how many *more* arcs still need to be drawn from the *existing* states in the diagram? [1 mark]

- (a) No more arcs need to be drawn from the existing states.
- (b) One more arc needs to be drawn from the existing states.
- (c) Two more arcs need to be drawn from the existing states.
- (d) Three more arcs need to be drawn from the existing states.
- (e) Four more arcs need to be drawn from the existing states.

The question asks: "An arc labeled 0 will have to be drawn from the state labeled "1". Which of these /could/ that arc legitimately lead to? There may be multiple correct answers. Circle \*ALL\* that apply.

- A new accepting state.
- A new rejecting state.
- The start state.
- The state labeled "...0".
- The state labeled "garbage".

The correct answers are "a new accepting state" and "the state labeled "...0"". The answer **MUST** be accepting, which is enough to eliminate all but these. To support these, students would need to recognize that "merging" into "...0" is OK (because starting 00 or 10 will make no difference to whether a longer string accepts or rejects). Additionally, they must recognize that it's OK to "duplicate" portions of a DFA, just in terms of correctness.

On “add arrow”, of the 5 students to whom it’s posed (but note that at least one student mentioned that they seemed not to be allowed to add extra states):

- 12: adds but retracts -> 0 arc; adds -> ...0 arc; states no other arcs possible

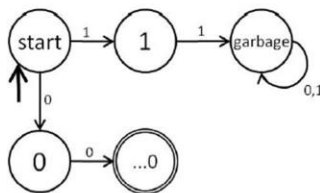


- 11: adds only  $\rightarrow$  ...0 arc
- 9: adds  $\rightarrow$  0 and  $\rightarrow$  ...0 arcs
- 8: doesn't really understand Q (but adds  $\rightarrow$  0 arc)
- 7: adds  $\rightarrow$  0 and  $\rightarrow$  ...0 arcs

Of 6 students to whom the “# of arrows” question is posed:

- 7 misunderstands Q (tries to correctly complete DFA), finally sees real Q and responds after that point w/in 30s
- 8 unfamiliar w/concept (but also tries to correctly complete DFA); never really settles into problem
- 9 partially misunderstands Q (approaching as if correctly complete DFA), but settles down to 4
- 10 doesn't know what DFAs are
- 11 even after rephrasing to elim stated purpose of DFA, distracted by “correctness”, gives 4 as answer
- 12 20s after remembering def'n of “legal”, finishes w/4 as answer

For Questions 9–11: You are building a DFA to recognize the language: “strings of 0s and 1s with at least two digits that do *not* contain two consecutive 1s”. So far, you have the following, which is correct but incomplete:



An arc labeled 0 will have to be drawn from the state labeled “1”. Which of these *could* that arc legitimately lead to? There may be multiple correct answers. Circle **ALL** that apply! [2 marks]

- (a) A new accepting state.
- (b) A new rejecting state.
- (c) The start state.
- (d) The state labeled “...0”.
- (e) The state labeled “garbage”.

Almost everyone puts the 1 → 0 arc in their solution.

Almost no one understands that they can add states. Bad question design? (At least paired w/the previous problem?)

(Detailed interview analysis is still TODO ☹.)

The question asks: "An arc labeled 0 will have to be drawn from the state labeled "1". Which of these /could/ that arc legitimately lead to? There may be multiple correct answers. Circle \*ALL\* that apply.

- A new accepting state.
- A new rejecting state.
- The start state.
- The state labeled "...0".
- The state labeled "garbage".

The correct answers are "a new accepting state" and "the state labeled "...0"". The answer **MUST** be accepting, which is enough to eliminate all but these. To support these, students would need to recognize that "merging" into "...0" is OK (because starting 00 or 10 will make no difference to whether a longer string accepts or rejects). Additionally, they must recognize that it's OK to "duplicate" portions of a DFA, just in terms of correctness.

On “add arrow”, of the 5 students to whom it’s posed (but note that at least one student mentioned that they seemed not to be allowed to add extra states):

- 12: adds but retracts -> 0 arc; adds -> ...0 arc; states no other arcs possible

- 11: adds only  $\rightarrow \dots 0$  arc
- 9: adds  $\rightarrow 0$  and  $\rightarrow \dots 0$  arcs
- 8: doesn't really understand Q (but adds  $\rightarrow 0$  arc)
- 7: adds  $\rightarrow 0$  and  $\rightarrow \dots 0$  arcs

Of 6 students to whom the “# of arrows” question is posed:

- 7 misunderstands Q (tries to correctly complete DFA), finally sees real Q and responds after that point w/in 30s
- 8 unfamiliar w/concept (but also tries to correctly complete DFA); never really settles into problem
- 9 partially misunderstands Q (approaching as if correctly complete DFA), but settles down to 4
- 10 doesn't know what DFAs are
- 11 even after rephrasing to elim stated purpose of DFA, distracted by “correctness”, gives 4 as answer
- 12 20s after remembering def'n of “legal”, finishes w/4 as answer

# Low Mean Problems

Unweighted Mean	Problem Mean	Problem Correlation	Adjusted Mean	Topic
0.73222744	0.199122807	0.391418802	-4.205337015	Heap/Asymptotic Analysis (2 variable analysis, find k-th smallest in n nu
0.705661765	0.32254902	0.374684015	-2.668872943	Asymptotic Analysis/ADTs ("dictionary dora" question, sketch behavio
0.688011564	0.41087963	0.543997801	-2.203896611	Induction Proof (open-ended, strong, code-based (Racket-ish recursive
0.617412418	0.303191489	0.572218428	-2.02942582	Heaps (complex problem troubleshooting)
0.705661765	0.464705882	0.50537874	-1.678567584	Combinatorics (order doesn't matter, replacement, "at least 10" limited
0.73222744	0.546052632	0.520809062	-1.468595307	Functions (MC, cardinality, injection/bijection/surjection, pigeonhole)
0.6836933	0.476911977	0.79708724	-1.439631886	P vs. NP (show prob in P, show prob in NP, reduction from IS)
0.683545973	0.466569767	0.624281229	-1.383294823	Working Computer (newer, open-ended, somewhat similar to 2010W2.9
0.705661765	0.508235294	0.353694098	-1.375329253	Sorts/BSTs/Heaps (open-ended asymptotic analysis in interesting situat
0.700276696	0.494318182	0.646238294	-1.374351207	Memory management/Sorting algorithm (C++, space complexity)
0.653105503	0.458815029	0.56689642	-1.273761376	Predicate Logic Proof (open-ended, direct, EEA, big-O like)
0.659589036	0.476293103	0.450958138	-1.232959284	Sorting Algorithms (selection and quick comparison)
0.653105503	0.469291908	0.67632922	-1.205075335	Working Computer (newer, open-ended)
0.683545973	0.497416021	0.587305738	-1.186639792	Induction Proof (MC, strategy, nearly identical to 2010W2.10)
0.683545973	0.50503876	0.718495299	-1.138042319	Induction Proof (open-ended, strong, triangulation, nearly identical (bu
0.688011564	0.548821549	0.791351467	-1.106911058	Working Computer/Seq'l Circuit (open-ended + MC, similar to parts of se
0.73222744	0.597744361	0.600299496	-1.060827218	Loop Invariant/Induction Proof (theorem given: "for any n/m exists q/r s
0.6836933	0.532125769	0.724361238	-1.055228043	DP (broken down step-by-step, # of ways to make change, 1-D but not gu
0.628239728	0.439393939	0.7652708	-1.04917734	Induction Proof (strong, open-ended, dividing stack of a+b scores a*b bu
0.628239728	0.446969697	0.644091386	-1.007088431	DFA <-> Seq'l Circuit (open-ended)
0.659589036	0.510344828	0.43274854	-1.003906796	External Memory Tree (B+ tree, min # nodes given height/M/L, semi-con
0.653105503	0.500963391	0.499601425	-0.997438225	Induction Proof (MC, strategy)
0.700276696	0.551515152	0.641368177	-0.992678593	Recursion (call tree, tail recursion, just these parts of 2009W1.16)
0.653105503	0.502023121	0.779417012	-0.990490672	Induction Proof (open-ended, strong, base case given, triangulation, ver

Students have done terribly on strong induction but weak induction is sometimes quite bad and sometimes fairly good. Clearly, both are worth assessing to understand how changing instructional strategies affect students' performance.

Besides induction, various topics show up below the mean, such as Working computer-related problems, predicate logic proof, predicate logic translation, and DFA <-> sequential circuit problems.

Almost anything concrete shows up well above the mean, e.g., set and function problems, propositional logic evaluation or straightforward equivalence, number rep, etc.

# Low Correlation Problems

Unweighted Mean	Problem Mean	Problem Correlation	Adjusted Mean	Topic
0.659589036	0.747126437	0.003087377	0.588829493	Hash Table (simulation, double hashing)
0.661328656	0.994285714	0.046197893	1.813520218	DFA (MC, simulation)
0.661328656	0.992380952	0.057182762	1.803145534	DFA (MC, formal rep)
0.700276696	0.958441558	0.075378499	1.722721652	Graphs (draw, isomorphism, nearly identical to 2009W1.12)
0.661328656	0.52	0.079397651	-0.769776071	Predicate Logic Translation (MC, Logic -> English)
0.628239728	0.746212121	0.144438716	0.655423467	Prop Logic Proof (open-ended, standard proof, similar to 2009S.3)
0.661328656	0.660952381	0.167216546	-0.002049463	Predicate Logic Proof (MC, proof strategy)
0.661328656	0.946666667	0.198819734	1.55415312	DFA (MC, formal rep)
0.700276696	0.936363636	0.205014669	1.575396745	Binary Tree (Parson's Puzzle version of 2009W1.9, C++, implementation,
0.659589036	0.931034483	0.206140413	1.825906223	Graphs (draw, isomorphism)
0.661328656	0.485714286	0.2138816	-0.956520381	DFA (MC, design)
0.661328656	0.931428571	0.230638812	1.471155649	DFA (MC, formal rep)
0.688011564	0.828703704	0.232346015	1.118856729	Prop Logic Equivalence (MC, similar to 2010W2.1)
0.73222744	0.927631579	0.236079092	1.541474586	Hash Tables (two advantages of double hashing over other hash table te
0.661328656	0.923809524	0.245451057	1.429656914	DFA (MC, understanding)
0.73222744	0.934210526	0.261527449	1.593372343	ADTs (advantages of BST over open addressing hashing)
0.661328656	0.628571429	0.273738986	-0.178419089	DFA (MC, design)
0.73222744	0.747807018	0.274421515	0.122935901	Time Complexity/Space Complexity (MC, select multiple grab bag, sorts,
0.705661765	0.791176471	0.274963678	0.595719896	Hash Table (simulation, double hashing, 4 inserts into already-populated
0.705661765	0.941176471	0.27816474	1.640662792	Graphs (given adj list, draw graph, give adj matrix)
0.661328656	0.965714286	0.279325075	1.657899959	DFA (MC, formal rep)
0.705661765	0.788235294	0.279336058	0.57523082	Heaps (MC, similar to CI question from 2011W2.4,
0.659589036	0.551724138	0.290536214	-0.725564532	Balanced Trees (MC, AVL, algorithm analysis)
0.6836933	0.867694805	0.300459885	1.28103655	Amortized Analysis (critique)

Many problems that are weakly correlated are also those students have done quite well on. Sometimes they're MC questions (guessing?). Concrete problems like number representation and concrete set/function problems also show up often.

Perhaps the most surprising is 2012S2's predicate logic proof problem #10. Students did poorer than average on this problem, yet it is relatively weakly correlated (0.52). Maybe this is high enough anyway to be uninteresting? (As it turns out, it's the highest-correlated of the MC questions. Bear in mind that these did NOT benefit as much from the equal-weighting process, either.)

Let's refocus on low correlates, but those on which students did fairly poorly (below mean).

Irritatingly, yet another functions MC problem "tops" this list.

Several MC problems on which it looks like students may have just guessed rank highly (including the induction strategy planning MC questions

Perhaps tellingly, of ALL low correlates, the lowest four (0.14 up to 0.35, next is 0.40) are all early-term questions: prop logic proof/equivalence, number rep, circuit design.

# High Correlation Problems

Unweighted Mean	Problem Mean	Problem Correlation	Adjusted Mean	Topic
0.595117845	0.433333333	0.811163781	-0.785871568	Induction Proof (strong, open-ended, dividing stack of a+b scores a*b bu
0.6836933	0.476911977	0.79708724	-1.439631886	P vs. NP (show prob in P, show prob in NP, reduction from IS)
0.595117845	0.569444444	0.796360036	-0.124709068	Functions (closed-ended MC-ish, pre-image/image/injective/surjective,
0.661328656	0.618253968	0.796029275	-0.234615294	Miscellaneous (MC)
0.688011564	0.548821549	0.791351467	-1.106911058	Working Computer/Seq'l Circuit (open-ended + MC, similar to parts of se
0.661328656	0.708333333	0.781922156	0.256020799	Functions (MC + brief justification, injective/bijective/surjective)
0.653105503	0.502023121	0.779417012	-0.990490672	Induction Proof (open-ended, strong, base case given, triangulation, ver
0.617412418	0.541843972	0.770617916	-0.488066015	Parallel Algorithms/ADTs (really about linked lists (no random access) an
0.628239728	0.439393939	0.7652708	-1.04917734	Induction Proof (strong, open-ended, dividing stack of a+b scores a*b bu
0.628239728	0.686363636	0.747386088	0.322921087	Functions Proof (open-ended, prove or disprove, somewhat similar to 20
0.595117845	0.5	0.744593975	-0.462036874	DFA <-> Seq'l Circuit (open-ended)
0.659589036	0.540708812	0.740485281	-0.799660273	Induction Proof/Recursion (pseudocode to C++, call tree, tail recursion)
0.756680889	0.700374532	0.733247625	-0.402221532	DFA <-> Seq'l Circuit (MC + open-ended)
0.661328656	0.552698413	0.733101616	-0.591677331	Induction Proof (critique + open-ended, strong, code-based)
0.617412418	0.466058764	0.729702886	-0.977532002	Loop Invariant/Induction Proof (cons subsequence sum, complete invari
0.661328656	0.676785714	0.729542795	0.084190097	DFAs <-> Seq'l Circuit (open-ended)
0.6836933	0.532125769	0.724361238	-1.055228043	DP (broken down step-by-step, # of ways to make change, 1-D but not gi
0.6836933	0.666048237	0.724008271	-0.122846662	Divide-and-Conquer Algorithm ("most distant pair of duplicates", must b
0.756680889	0.623907615	0.723494285	-0.948458969	Induction Proof (strong, odd/even cases, open-ended, recursive reverse
0.661328656	0.586666667	0.722469058	-0.406662135	Predicate Logic Proof (Parson's Puzzle, proof strategy)
0.688011564	0.566666667	0.722079775	-0.964997443	Predicate Logic Translation (open-ended, negation, English -> Pred Logic
0.595117845	0.62962963	0.71906796	0.167641698	Predicate Logic Translation (open-ended, English <-> Logic)
0.683545973	0.50503876	0.718495299	-1.138042319	Induction Proof (open-ended, strong, triangulation, nearly identical (but
0.628239728	0.553429027	0.717591137	-0.415628502	Predicate Logic Translation (open-ended, English <-> Logic, similar to 200

Of the high correlation problems, one is unsurprising: the aggregate of the many MC questions in 2009W1.

Other strong correlates include:

- + some of the strong induction problems
- + the functions problem (perhaps b/c they're end-of-term material?? perhaps b/c they integrate many concepts? surprising!)

Filtering to high correlates on which students did reasonably well (better than mean on that exam), we get:

- + functions MC
- + functions proof
- + DFAs <-> Seq'l Circuits
- + Predicate logic proof
- + an induction proof (weak)
- + set proof
- + predicate logic translation

Not sure what to say from this. MANY of the high correlates are those on which students did poorly, not well.

## Some MC Questions

Unweighted Mean	Problem Mean	Problem Correlation	Adjusted Mean	Topic
0.661328656	0.994285714	0.046197893	1.813520218	DFA (MC, simulation)
0.661328656	0.992380952	0.057182762	1.803145534	DFA (MC, formal rep)
0.661328656	0.52	0.079397651	-0.769776071	Predicate Logic Translation (MC, Logic-> English)
0.661328656	0.660952381	0.167216546	-0.002049463	Predicate Logic Proof (MC, proof strategy)
0.661328656	0.485714286	0.2138816	-0.956520381	DFA (MC, design)
0.661328656	0.923809524	0.226328817	1.471165646	DFA (MC, formal rep)
0.661328656	0.923809524	0.245451057	1.429656914	DFA (MC, understanding)
0.661328656	0.628571429	0.273738986	-0.178419089	DFA (MC, design)
0.661328656	0.965714286	0.279325075	1.657899959	DFA (MC, formal rep)
0.661328656	0.836190476	0.302046296	0.952421455	Prop Logic Equivalence (MC)
0.661328656	0.813333333	0.315282183	0.827925248	Sets (MC, empty set)
0.661328656	0.792380952	0.341282625	0.713803725	DFA (MC, design)
0.661328656	0.914285714	0.35730583	1.377783494	Sets (MC, cardinality, concrete)
0.661328656	0.853333333	0.387707432	1.04579361	Sets (MC, comprehension-style, subset)
0.661328656	0.693333333	0.424294577	0.174320163	Number representation/Prop logic (MC, dec -> bin, evaluation of prop)
0.661328656	0.902857143	0.445772731	1.315535391	Sets (MC, concrete, member, cartesian product)
0.661328656	0.866666667	0.453583056	1.118416397	Sets/Predicate Logic (MC)
0.661328656	0.740952381	0.467990155	0.43368726	Functions (MC, terminology)
0.661328656	0.851428571	0.499356622	1.035418926	Functions (MC, terminology)
0.661328656	0.866666667	0.529261409	0.612424247	Sets (MC, power sets, subsets)
0.661328656	0.866666667	0.549126117	1.118416397	DFA (MC, design)

MC/closed-ended questions from one exam; note the high correlation problems.

The two really interesting ones are the two highlighted “DFA (MC, design)” questions, one of which is the highest correlation problem and the other is the lowest performance problem.

Looking at the small MC questions on which I have lots of detail (2009W1's problem #1 parts), most are not strongly correlated with exam results (no surprise!). One DFA design problem is surprisingly well-correlated (among a handful near or above 0.5)

WHY did some people make different choices? We cannot tell.

The other question with significant correlation is a sets question in which one correct answer (of two) is trivial (a set is a subset of itself) and the other is very tricky (because the two sets' intersection is non-empty, they share an element; so,  $(a,a)$  is a member of  $P(A \times B)$  for some  $a$  in  $A$  and so  $\{(a,a)\}$  is a subset of  $P(A \times B)$ ).

Perhaps could ask whether and why  $\{(a,a)\}$  is a subset of  $P(A \times B)$  given the conditions on  $A$  and  $B$ ?

In terms of the problems people did worst on, they are:

- + Besides the two discussed in the previous problems [referring to the DFA described above], how many more arcs still need to be drawn from the existing states in the diagram? (This /should/ be easy! Not sure if many of the wrong answers are 4s, indicating perhaps that they didn't understand the question.)
- + Problem 1.2, a rather messy predicate logic question translating "everybody loves somebody that loves them back". This is really about negating predicate logic statements and implications from my perspective.
- + You are building a DFA to recognize the language: "words that end in a vowel". Should the DFA's start state be accepting or rejecting? (Again, this /should/ be easy. It boils down to "is an empty string a word ending in a vowel?".)
- + The set problem from above about  $P(A \times B)$ . This is the only one that students did poorly on that also had reasonably high correlation with the overall exam. (But note that the average on this problem was already up to 66%; indeed, only the first two described were near or below 50%).
- + A proof strategy problem (#1.4) that I recall feeling was poorly written during grading.
- + A somewhat messy propositional logic evaluation problem (#1.8, but this one was the first that was above the mean).



## Easy MC: Number Representation

You're given a number in hexadecimal representing a 32-bit signed integer. How do you tell whether it's negative? Here are some examples of positive and negative hexadecimal values (mixed together).

0x8A040010

0x1

0x1000

0xF10

0x12345678

0xFFFA3882

0xFF

0xFFFFFFFF

## Easy MC: Balanced BST Performance

Suppose it takes 100 nanoseconds on average to find a random key in a balanced binary search tree with 1,000 keys. Estimate the average time it will take to find a random key in a well-balanced binary search tree with 1,000,000 keys.

- $\log(1000) = 100$
- $100 \lg 1000$
- ???

Imagine you were creating a dance. Here's an algorithm you could use to describe the dance:

```
Dance(n) :  
  if n = 1:  
    walk forward 1 meter  
  else:  
    turn 45 degrees to the left,  
    do the steps in Dance(n - 1),  
    turn 90 degrees to the right,  
    do the steps in Dance(n - 1),  
    turn 45 degrees to the left
```

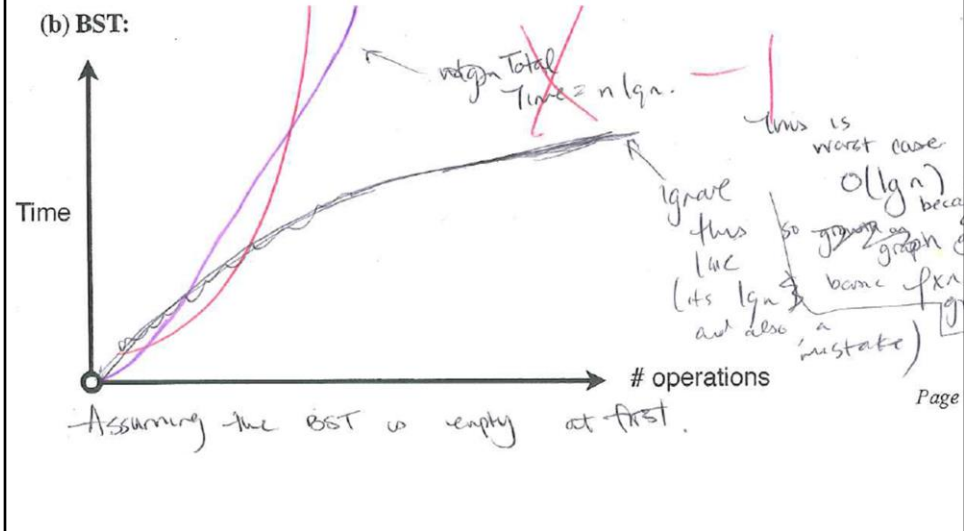
Let  $M(n)$  be the number of meters of walking you have to do in the dance. So,  $M(1) = 1$ .

Give a formula for  $M(n)$  that is correct for all  $n \geq 2$ . Your formula can and should be in terms of  $M(\cdot)$ . For example the following is a formula for  $M(n)$  in terms of  $M(\cdot)$ , although it is **not** a correct formula for this problem:  $M(n) = M(n/3) - 2$  for  $n \geq 2$ .

**Exam analysis lesson: communicating clearly is hard.**

Give a formula for  $M(n)$  that is correct for all  $n \geq 2$ . (That is, for  $n = 2, 3, 4$ , and all larger values.) Your formula **can and should be in terms of  $M(\cdot)$** .

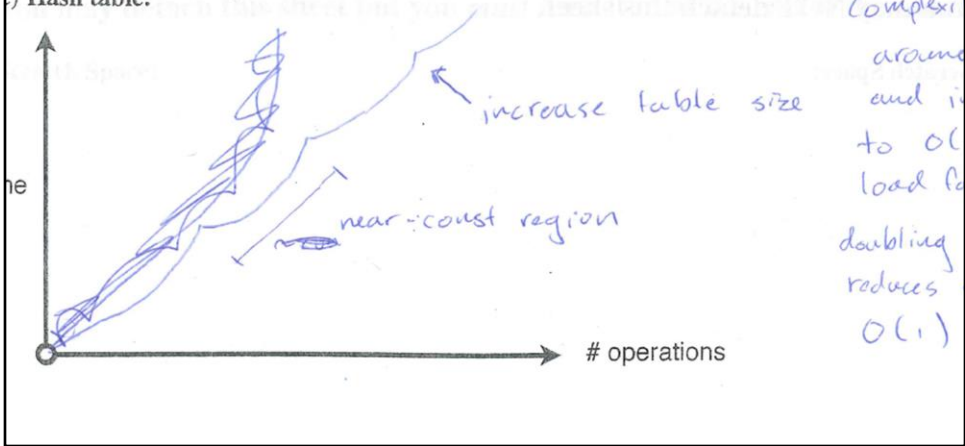
Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).



"worst case", "stick-like", and yet  $O(\lg n)$ . Why?

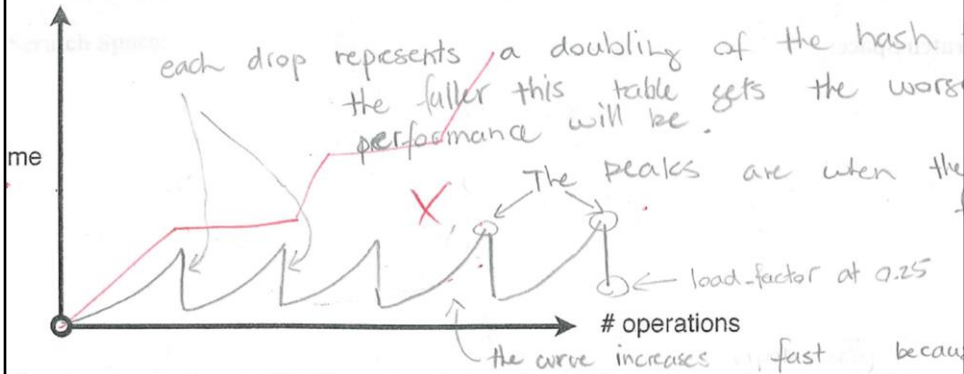
Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

b) Hash table:



Roughly sketch the dictionaries' behaviour responding to the commands "INSERT 100000, INSERT 99999, ..., INSERT 3, INSERT 2, INSERT 1" on the axes below. The vertical axis should be the **total time** for all operations completed to that point (as in the programming project). Label and briefly explain key elements of the sketches (e.g., unusual properties, asymptotic behaviour, and relative runtimes between graphs).

(c) Hash table:



Now imagine the keys 1–4,000 have already been inserted in random order into an initially empty dictionary. You get no timing or other information from the insertion process, but you get high-precision timing after every 100 operations from any further operations.

## George, SIGCSE 2000

gave incorrect responses due to several exogenous factors. These especially included: misconceptions concerning variable updating and computer memory storage; difficulty with evaluating conditional statements; and idiosyncratic notions about the behaviour of some program elements. Misconceptions concerning variable updating were generally related to the use of an 'immediate invocation update' evaluation strategy or a 'delayed invocation update' strategy. The former concerned the misconception that the value of a variable was explicitly changed by an argument expression. The latter concerned the misconception that on return of control to a suspended process, the value of a variable was changed to correspond to the parameter value in a previous invocation. Misconceptions about memory storage concerned the attempt to understand aspects of the recursive execution process (e.g. variable storage) in terms of the technical workings of computers. Difficulty with conditional statements stemmed from an inability to rationalise certain expressions, such as 'IF 1<1 THEN... ELSE'. Some students seemed to have first evaluated '1>1' which was false and then proceeded to think that '1<1' was true. Idiosyncratic notions regarding the behaviour of

# Götschi, Sanders, and Galpin, SIGCSE 2003

and *return-value* models. Some non-viable models, such as the *copy* model, are considered to be precursors of the *active* model. Had these models been used by lecturers in order to construct solutions, such as the *step* and *return-value* models, students had many misconceptions about recursion and a non-viable model.

**Active model (Ac):** Although many students did show evidence of understanding the active flow of control and the instantiations of the recursive functions with smaller argument values, as well as reaching the base case correctly, they did not show the passive flow and simply calculated the solution at the base case. In the case of some recursive programs (such as Q1), the correct solution *can* be evaluated at the base case, but this is not always possible. For example, in Q2, because of the order and precedence of the operations, the base case must be evaluated and the solution passed back to previous invocations before a solution can be calculated. It is possible that students use the active model when it is viable and the copy model otherwise.

**Step model (S):** Students with these models have no concept of any recursive flow of control. With this model, students simply evaluate the IF-THEN-ELSE and execute either the recursive condition once (one-step), both the recursive condition and the base case (two-step).

**Return value model (R):** This model stems from misconceptions about when return values from a function call are evaluated. Many students hold the misconception that at each instantiation a value is evaluated, before the next instantiation is complete. These values are stored and all combined into a solution.

**Looping model (L):** The looping model procedure is viewed as a sequence of instantiations at each iteration. The solution is reached, thus the base case is reached, thus the base case of the loop.

Note the value of a FASI that can be used to track student “trajectory” through responses!