The Mystery of “b := (b = false)”

Stuart Reges
University of Washington
Computer Science & Engineering
Seattle, WA 98195
+206.685.9138
reges@cs.washington.edu

ABSTRACT

This paper describes some unusual patterns that emerged from a statistical analysis of the 1988 Advanced Placement Exam in Computer Science. Most multiple-choice questions on the exam had few significant correlations with other parts of the exam. But a small set of five questions had a nontrivial correlation with many parts of the test. One question in particular demonstrated such correlations. It asked about the effect of the assignment statement “b := (b = false)” for a boolean variable b. One interpretation of this data is that these questions are testing general programming aptitude. The paper presents the analysis along with a discussion of the possible implications.

Categories and Subject Descriptors

K3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms: Human Factors, Algorithms.

Keywords: CS1, CS2, Mental Models.

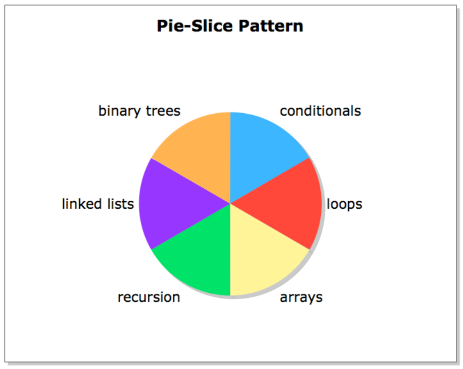
1. INTRODUCTION

The Advanced Placement Exam in Computer Science (AP/CS) is an exam developed by The College Board and administered by the Educational Testing Service to allow students to get credit for studying college-level computer science in high school [1]. The exam is divided into A and B sections that roughly correspond to the traditional CS1 and CS2 classes described by the ACM. The exam is offered in an AB format that tests both CS1 and CS2 material and in an A-only format that tests just CS1 material.

In 1988 the author served as the “Chief Reader” for AP/CS. In that role he was responsible for analyzing student performance on the exam and advising the development committee and ETS about how best to evaluate student performance. 1988 was a pivotal year because it was the first in which the A-only format was offered. The author was given access to the raw data for the more than 10 thousand students who took the exam and he performed an extensive analysis using SPSS to determine what could be learned from traditional statistical analyses.

A striking pattern emerged from this analysis. A small group of questions that the author dubbed the “powerhouse questions” showed up over and over again as being the most predictive of success on the exam. One question in particular stood out as having this predictive ability.

The author expected to find a pie-slice pattern in the set of correlations. It seems likely that questions will fall into various slices with loop questions correlating with loop questions, recursion questions correlating with recursion questions, and so on, as in the following diagram:



Instead what emerged was a donut-like shape with a small core of questions (the powerhouse questions) accounting for the vast majority of the nontrivial correlations with the rest of the exam.

This donut pattern suggests that the powerhouse questions are testing computer science ability in a more fundamental way than the other questions.

2. APTITUDE AND MENTAL MODELS

Computer Science educators have for years complained that introductory courses seem to be divided between a group of students who “get it” and a group of students who do not. Donald Knuth has written about this phenomenon:

> “Educators of computer science have repeatedly observed that only about 2 out of every 100 students enrolling in introductory programming classes really resonate with the subject and seem to be natural-born computer scientists…I conclude that roughly 2% of all people ‘think algorithmically,’ in the sense that they can reason rapidly about algorithmic processes.” [5]

Many computer science education researchers have been exploring the question of what factors predict success in CS1 and CS2 [2, 7]. In fact, Knuth was motivated by just such a study. It was an unpublished study conducted by Gerrit DeYoung in which he found that a measure of quantitative reasoning was not a predictor of success in a course for CS majors but was a reasonable predictor of success in a course for nonmajors. Knuth's tentative conclusion was that there is some kind of CS aptitude that is not measured by standard tests of quantitative reasoning and that students who lacked that ability were instead relying on general quantitative aptitude.

This theory has become more popular recently. Several researchers have been attempting to identify the mental models that students have in first-year programming courses. Dehnadi and Bornat [4] claim to have a test that can predict success or failure in a first-year programming course. There seems to be some question as to the effectiveness of their model with some obtaining similar results [6] and others failing to reproduce the results [3].

The pattern of correlations for the AP CS powerhouse questions won't settle the question of whether there is a special aptitude that computer scientists have or whether success in CS1 can be tested before a student takes the course. But the powerhouse questions at least might serve to point us in the direction of what to pay attention to in exploring these difficult questions.

3. STATISTICAL RESULTS

This section summarizes the statistical results for the 1988 exam. Data for all 7,374 students taking the full AB exam were used to derive Pearson's correlation coefficients. Any correlation below 0.2 was eliminated as being too low to consider. Even a correlation of 0.2 is not particularly strong, but it does seem to indicate some kind of underlying relationship.

The AB exam consisted of four parts that were completed separately:

- A multiple-choice: 35 questions
- A free-response: 3 questions
- B multiple-choice: 15 questions
- B free-response: 2 questions

The programming language in use at the time was Pascal.

Multiple-choice questions were scored as 1 for a correct answer, -0.25 for an incorrect answer (to account for random guessing) and 0 for a skipped question.

3.1 A Multiple-Choice Correlations

There were 35 multiple-choice questions on the A part of the test, which meant there were 595 correlations to compute for distinct pairs of questions. Of those, 56 were correlated at the level of 0.2 or higher. So, in general, there weren't many correlations between multiple-choice questions because this represents just 9.4% of the possible correlations. The number of correlations is not nearly as surprising as the pattern. As Table 1 indicates, 46 of the 56 correlations were accounted for by just 5 questions, which will be referred to as the "powerhouse questions" for the remainder of this paper.

If these five questions had been eliminated from the exam, the remaining 30 questions would have just 10 correlations of 0.2 or higher out of a potential 435 correlations, which is less than 2.3%. By contrast, question 23 correlated at 0.2 or higher with over half of the other A questions.

It is worth noting that half of the 10 correlations of 0.2 or higher that were not included in Table 1 came from a single question. The question is similar in nature to the other five, but it appeared late on the exam (question 32) and statistical evidence suggests that it was not reached by approximately one-third of the students. As a result, it might be a powerhouse question that wasn't quite able to prove itself because the test was too long.

Question	14	15	18	20	23
3					0.22
5		0.23	0.22	0.23	0.24
6	0.23	0.24			
10		0.22	0.21	0.21	0.24
11		0.21	0.21	0.21	0.22
12	0.20	0.21	0.20		0.23
13	0.27	0.28	0.24	0.25	0.29
14	-	0.55	0.26	0.26	0.34
15	0.55	-	0.29	0.29	0.38
16				0.20	0.21
17	0.24	0.25	0.29	0.30	0.25
18	0.26	0.29	-	0.30	0.31
19	0.24	0.25	0.26	0.25	0.27
20	0.26	0.29	0.30	-	0.25
22	0.21	0.21			0.22
23	0.34	0.34	0.31	0.35	-
24					0.25
25					0.24
26					0.22
28				0.21	0.24
32				0.21	0.24
total	10	13	11	13	19
%	29%	38%	32%	38%	56%

Table 1: Correlations of Powerhouse Questions against other A Multiple-Choice Questions

3.2 B Multiple-Choice Correlations

There were only 15 questions on the B multiple-choice section of the exam, so there is less data to work with. Of the 105 possible correlations, 14 of them (13.3%) were 0.2 or higher. A subset of three questions accounted for 8 of these 14 correlations, but no question had more than three questions that it correlated with at the level of 0.2 or higher. So there weren't any real powerhouse questions within the B questions.

A more interesting pattern emerges when comparing the group of 35 A multiple-choice questions against the group of 15 B multiple-choice questions. It is likely that there would be few correlations because they cover material from two different courses (CS1 and CS2), and, in general, there isn't much of a correlation. Only 46 of the 525 correlations between the two groups were at the level of 0.2 or higher, which is just 8.8% of the possible correlations. The striking statistic is that almost all of them come from the powerhouse questions, as indicated in Table 2.

Consider, for example, question 23. It has a nontrivial correlation with 56% of the A questions, but there is no particular reason to believe that it would correlate with the more advanced material on the B test. Yet, it correlates with even more of the B questions (67%). It correlates with two-thirds of the B questions even though the B questions don't tend to correlate with each other.

Question	14	15	18	20	23
36	0.21	0.21	0.21	0.25	0.24
39	0.20		0.22	0.24	0.23
40	0.21	0.21		0.21	0.23
41	0.21	0.21	0.41	0.22	0.23
42		0.21			0.23
43	0.21	0.20	0.21	0.21	0.23
45	0.22	0.23	0.22	0.24	0.26
47	0.22	0.23		0.24	0.23
49		0.20		0.22	0.21
50	0.23	0.26	0.22	0.22	0.26
total	8	9	6	9	10
%	53%	60%	40%	60%	67%

Table 2: Correlations of Powerhouse Questions against B Multiple-Choice Questions

As another example of how pronounced this effect is, consider the powerhouse questions separately from the rest of the A questions. The remaining 30 A questions have a mere 4 correlations with the 15 B questions (less than 1% of the 450 possible correlations). Yet with the powerhouse questions, 42 of the possible 75 correlations are 0.2 or higher (56%).

3.3 Free Response Correlations

The free-response questions involve writing substantial amounts of code. For example, one of the B free response questions involved reversing a linked list and the other involved constructing the mirror image of a binary tree. The free response questions are graded on a 10-point scale (0 to 9).

It is difficult to get a strong correlation between multiple-choice questions and free-response questions because the multiple choice questions have just three possible scores (1, 0 and -0.25) while the free response involves 10 possible scores. Even so, there were a number of multiple-choice questions that correlated at the level of 0.2 or higher with free-response questions and once again the powerhouse questions were at the top of the list.

The first A free response question was to write a function that would return the number of digits in its integer argument. Table 3 shows the top five correlations against all 50 multiple-choice questions (both A and B). These are exactly the powerhouse questions, with the now familiar question 23 topping the list.

Rank	Question	Correlation
1	23	0.36
2	20	0.34
3	15	0.34
4	18	0.32
5	14	0.30

Table 3: Top Five Correlations with Multiple-Choice for A Free-Response Question 1

The second A free-response question involved implementing a type for storing elapsed time. It had a nearly identical set of correlations with the powerhouse questions again taking the top five spots.

The third A free-response question involved manipulating a two-dimensional array with a "zoom" operation (replacing each integer with an n-by-m grid of that integer). It had a slightly different pattern in its top five, as shown in Table 4.

Rank	Question	Correlation
1	24	0.39
2	23	0.37
3	15	0.36
4	32	0.34
5	14	0.34

Table 4: Top Five Correlations with Multiple-Choice for A Free-Response Question 3

Question 24 was about graphics, which might explain why it correlated so highly with a question about two-dimensional arrays. After it we again see question 23 at the top of the list. We also see question 32, which was the question that almost qualified to be a powerhouse question. So in this case three of the five powerhouse questions are in the top five (the other two were ranked 7th and 8th).

It is again not that surprising that A multiple-choice questions are the ones that correlate with A free-response questions. The bigger surprise is that the powerhouse questions also top the lists for the B free-response. For example, the first B free-response question involved reversing a linked list. One would expect that the linked-list multiple-choice question would correlate with it, but not so. The five powerhouse questions were the top five correlations, with question 23 again with the highest correlation.

Similarly for the second B free-response question which involved constructing the mirror image of a binary tree, four of the five top correlated questions came from the powerhouse questions with question 23 again in the top position.

3.4 A-Only vs AB

A total of 3,344 students took the A-only version of the exam. As a result, they didn't answer any of the B questions. But they answered the exact same multiple-choice and free-response questions. Across the board the A-only students scored lower than the AB students, but it was interesting to note that the six questions that had the greatest difference between the two populations were the five powerhouse questions along with question 32, as indicated in Table 5.

Question	AB correct	A-Only correct	Delta
20	69.2%	38.7%	30.5
23	60.0%	35.3%	24.7
15	66.3%	43.8%	22.5
32	46.6%	25.7%	20.9
14	65.2%	46.0%	19.2
18	71.7%	52.6%	19.1

Table 5: Multiple-Choice Questions with Greatest Difference in Performance (A-only vs AB)

4. THE POWERHOUSE QUESTIONS

What exactly do the powerhouse questions look like? Let's explore question 23 in depth because it had the most nontrivial correlations. The exact text of the question is reproduced below:

23. If *b* is a Boolean variable, then the statement *b := (b = false)* has what effect?

(A) It causes a compile-time error message.
(B) It causes a run-time error message.
(C) It causes *b* to have value *false* regardless of its value just before the statement was executed.
(D) It always changes the value of *b*.
(E) It changes the value of *b* if and only if *b* had value *true* just before the statement was executed.

Only 5.4% of the students skipped the question. Of those who answered, 60% got it right. And getting this question right turned out to be a predictor of success on most of the rest of the exam, including solving complex problems like reversing a linked list.

To answer this question correctly, a student has to be able to read the code and simulate its execution. They also have to be able to identify the correct answer among the given choices.

Questions 18 and 20 both involved understanding recursive code. Question 18 asked students what output is produced by the call Wow(16) for the following procedure:

```
procedure Wow(n : integer);
begin
    if n > 1 then Wow(n div 2);
    write(n, ' ')
end;
```

Question 20 involved reasoning about necessary and sufficient conditions for the following function to return an answer:

```
function WhatIsIt(x, n: integer) : integer;
begin
    if n = 1 then
        WhatIsIt := x
    else
        WhatIsIt := x * WhatIsIt(x, n - 1)
end;
```

In both cases, students had to reason about execution and understand what a mysterious bit of code will do.

Questions 14 and 15 both involved the following code

```
const Size = 10;
type GridType = array[1..Size, 1..Size] of char;
function YesOrNo(Grid: GridType;
                 Row, Colm : integer;
                 Mark : char) : Boolean;
var i, Count : integer;
    OK : Boolean;
begin {YesOrNo }
    Count := 0;
    for i := 1 to Size do
        if Grid[i, Colm] = Mark then
            Count := Count + 1;
    OK := (Count = Size);
    Count := 0;
    for i := 1 to Size do
        if Grid[Row, i] = Mark then
            Count := Count + 1;
    YesOrNo := (OK or (Count = Size))
End; {YesOrNo }
```

Question 14 asks students to identify whether the function will return true for certain scenarios. Question 15 asks students to identify the description that best characterizes the computation performed by the function.

Question 32, which was almost a powerhouse question, also involved reasoning about recursive code. Students were shown a template for a recursive multiplication function and were asked to identify the pair of statements (base case and recursive case) that would allow the function to work as intended.

5. POWERHOUSE AS A TEST OF APTITUDE?

So what do the powerhouse questions have in common? They all involve reading and understanding code. They all test whether students have a proper mental model of program execution. And they involve some of the most central concepts from the first year programming course: logic, recursion and two-dimensional arrays.

The author had the opportunity to present these results to a group of Stanford faculty, including the late Bob Floyd. Floyd, who had taught introductory programming many times, commented that the greatest single predictor he had noticed for success was whether students had a mental model of program execution, whether they could "play computer" in their head. He commented that these questions seemed to be very good at measuring that ability.

So we are left with the question of whether the powerhouse questions are a kind of CS IQ test measuring Knuth's algorithmic thinking ability. It would help explain why they correlate so frequently with so much of the test.

For example, consider question 23. On its surface, it seems to test just simple assignment with boolean variables. And yet it was the best at predicting which students would be able to reverse a linked list or to recursively construct the mirror image of a binary tree. So it seems likely that it is testing something fundamental about a student's ability to approach computer science problems.

Knuth provides an intriguing intuition about this in talking about the difference between mathematical reasoning and algorithmic thinking:

> "The other missing concept that seems to separate mathematicians from computer scientists is related to the 'assignment operation' :=, which changes values of quantities. More precisely, I would say that the missing concept is the dynamic notion of the *state* of a process. 'How did I get here? What is true now? What should happen next if I'm going to get to the end?' Changing states of affairs, or snapshots of a computation, seem to be intimately related to algorithms and algorithmic thinking."[5]

Question 23 is about assignment for a boolean variable that requires thinking about its value before the assignment statement and what value it will have afterwards. Question 18 involves simulating the execution of a recursive procedure and question 20 involves reasoning about necessary and sufficient conditions for termination of a code segment, both of which involve reasoning about the changing "state of a process." Questions 14 and 15 involve reading code that manipulates a data structure and reasoning about what it does, which again seems to match Knuth's description of thinking about the dynamic state of a process.

The powerhouse questions also match the kind of questions that Dehnadi and Bornat claim are useful at predicting success in CS1 [4]. They say that assignment and recursion are two of the most difficult concepts for students to master and the sample questions they provide look very much like question 23.

6. IMPLICATIONS FOR TESTING

People often ask, "If these are such powerful questions, should we pack our exams with these kind of questions?" The answer depends on what you are trying to accomplish with your tests.

Most instructors include "program mystery" questions that students find annoying because they can be so confusing. The pattern of powerhouse correlations suggests that this is a useful kind of question to include. But should you include many such questions or assign them a particularly heavy weighting?

In terms of assessment, you would not want to overemphasize powerhouse questions. The powerhouse questions have a nontrivial correlation with each other, so including many of them may be unnecessary. When the author ran a regression analysis to pick a set of multiple choice questions that best predicted overall success on the AP exam, question 23 was picked first. But once it had been included, the other powerhouse questions weren't particularly strong predictors. The regression picked questions that seemed to be measuring mastery of specific skills and coverage of specific topics which are also important to include in any assessment of student performance.

But the powerhouse type of question can serve a different role. If these questions do indeed test a student's mental model of computation, then it is possible that they would make good practice problems. They help students to deduce flaws in their mental model of computation and they might serve as a kind of mental calisthenics for programming. There is some evidence for this in that the powerhouse questions had the greatest disparity between A-only and AB test takers. It is possible that this ability improves with practice, which could explain why for these particular questions students who took the longer AB course outperformed the students who took just an A course.

As an example, the author has designed a specific kind of question in an attempt to test Knuth's "snapshots of a computation." Students are shown a code fragment with particular points indicated with comments, as in the following Java method:

```
public static int mystery(int n) {
    int x = 0;
    while (n % 2 == 0) {
        // Point A
        n = n / 2;
        x++;
        // Point B
    }
    // Point C
    return x;
}
```

Students are asked to identify at each point whether certain assertions are always true, never true or sometimes true and sometimes false. For example, the assertion `(n % 2 == 0)` is always true at point A, sometimes true at point B and never true at point C. The author's anecdotal experience is that students often struggle with this style of question when they first encounter it, but they do fairly well in answering this type of question if they are given the opportunity to practice.

7. CONCLUSIONS

This paper has demonstrated a rather startling set of correlations that a small set of questions from the 1988 AP/CS exam have with the rest of the exam. The high number of nontrivial correlations with the entire exam suggest that there is something special about these questions.

Unfortunately, the analysis leaves us with more questions than answers. The powerhouse questions may point us in the direction of what kind of question to ask, but do they measure a fundamental ability that some people have more than others? If so, can that ability be effectively tested before a student takes a course? Can we add components to our courses like specially designed test questions that help students to hone this ability?

The author hopes that others will undertake similar studies to identify questions that are particularly good predictors of success in CS1 and CS2 and to attempt to answer some of these important questions.

REFERENCES

[1] http://www.apcentral.collegeboard.com.

[2] Bergin, S. and Reilly, R. 2005. *Programming: factors that influence success*. Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education

[3] Caspersen, M. E., Larsen, K. D., and Bennedsen, J. 2007. *Mental models and programming aptitude*. Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education.

[4] Dehnadi, S., and Bornat, R. 2006. *The Camel Has Two Humps*. Middlesex University Working Paper, www.cs.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf

[5] D. Knuth. 2004. *Selected Papers on Computer Science*. CSLI.

[6] Ma, L., Ferguson, J., Roper, M., and Wood, M. 2007. *Investigating the viability of mental models held by novice programmers*. Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education.

[7] Simon, Fincher, S., Robbins, A., Baker, B., Box, I., Cutts, Q., De Raadt, M., Haden, P., Hamer, J., Hamilton, M., Lister, R., Petre, M., Sutton, K., Tolhurst, D., and Tutty, J. 2006. *Predictors of success in a first programming course*. Proceedings of the Eighth Australasian Computing Education Conference.