Motivating Computer Engineering Freshmen Through Mathematical and Logical Puzzles

Behrooz Parhami, *Fellow, IEEE*

***Abstract*—As in many other fields of science and technology, college students in computer engineering do not come into full contact with the key ideas and challenges of their chosen discipline until the third year of their studies. This situation poses a problem in terms of keeping the students motivated as they labor through their foundational, basic science, and general education coursework. At the University of California, Santa Barbara, the Computer Engineering Program has sought to remedy this problem by offering a required freshman seminar, entitled "Ten Puzzling Problems in Computer Engineering." This pass/not-pass seminar, which meets once a week and is graded based solely on attendance, introduces the students to some of the most challenging problems faced by computer engineers in their daily professional endeavors and at the frontiers of research. To accomplish this feat in a nontechnical way, these problems are related to popular mathematical and logical puzzles. Each class session (60–75 min) begins by introducing the students to puzzles of a particular kind and letting them participate in formulating solutions. General solution methods for the puzzles are then discussed by the instructor, who then proceeds to demonstrate how the puzzles and their solution strategies are related to real technical challenges in computer engineering. The new one-unit course was well received during its first two offerings in spring 2007 and spring 2008, and its continuation is planned.**

***Index Terms*—Computer engineering education, freshman seminar, problem complexity, problem solving, puzzling problem, teaching-research links.**

I. Introduction

FRESHMAN seminars, which are being offered by many universities, have at least four different objectives. Some are aimed at helping students with the transition from high school to college. Others strive to create excitement and challenge by allowing the students to work in a small setting with a professor and fellow students on topics of special interest. A third category aims at introducing college freshmen to general challenges of college-level learning, resources available to students, and important study skills. Still others aim to offer a sampling of milestones, achievements, and societal impacts of a particular field, such as computer and information sciences, serving in effect as an overview of the discipline. Freshman seminars of all four kinds tend to forego the formal setting and requirements of a regular college course, focusing instead on stimulating curiosity, generating excitement, promoting interaction, and engendering self-improvement.

Anecdotal evidence on the usefulness and effectiveness of freshman seminars has led to a proliferation of such offerings in many institutions. This paper reports on the objectives and contents for a fifth kind of freshman seminar: one meant to introduce the students to the most challenging problems faced in their selected field of study by practicing engineers who design state-of-the-art products or processes, and by researchers at the forefront of technology. Lecture topics are chosen such that they can be related to interesting mathematical or logical puzzles, which lead off the discussion in each class session. The role of puzzles in instruction is further elaborated upon in Section II of the paper.

A complete syllabus for the freshman seminar, "Ten Puzzling Problems in Computer Engineering," and its lecture slides can be found at the course's website [1]. The teaching material for each lecture include presentation files in PowerPoint and PDF formats, a 4-slide handout (printable on two sides of a single sheet, with two slides per page) that allows the students to participate in solving puzzles in the early part of the lecture, and URLs of websites with additional information. The PowerPoint presentations contain a great deal of animation used to reveal information gradually, or to overlay different components of a diagram. Because these animations are lost in the PDF files, the PowerPoint versions should be used whenever possible.

II. The Role of Puzzles in Instruction

Many engineering problems are puzzle-like. Pieces of the puzzle are provided to engineers in the form of user/customer requirements, technological constraints, professional or industrial codes, and market realities. The engineer must then craft a product or process that either meets all of these (often conflicting) demands or else provide partial solutions, with clear justification of the tradeoffs made, when meeting all of the specifications is not possible. Even though all engineers encounter puzzle-like tasks, computer engineers seem to face a much greater share of such problems. Puzzling problems are, of course, plentiful in the research arena, regardless of the discipline.

So, what exactly is a puzzling problem? In a very interesting resource for the teaching of physics [2], the authors use the term "puzzling problem" as an antithesis for problems that "can be solved only through long, complex calculations, which tend to be mechanical and boring, and often drudgery for students." According to this view, a puzzling problem may ask, "How high could the tallest mountain on Mars be?" Such a problem can often be solved quite elegantly, via appropriate choices of

Manuscript received June 26, 2007; revised July 10, 2008. First published May 05, 2009; current version published August 05, 2009.

The author is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: parhami@ece.ucsb.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TE.2008.930087

variables or coordinates, once an enabling notion, appropriate analogy, or "eureka moment" has occurred to the student. Others have used the term to refer to trick questions, or riddles, such as, "Is it against the law for a man to marry his widow's sister?" or, "How much dirt is there in a 3-m by 3-m square hole that is 2 m deep?"

In the context of the freshman seminar course described in this paper, a puzzling problem is either an innocent-looking problem that reveals its depth and degree of difficulty when one begins to formulate a solution for it (e.g., rotate faces of a Rubik's cube until each of the six faces is single-colored), or a tough-looking problem that is readily tamed with the appropriate insight (e.g., draw four straight lines that connect all nine points in a 3×3 grid, without lifting your pen).

The use of puzzles has a long tradition in the teaching of mathematics [3]. The author's own experience, and relative success, in using puzzles to teach mathematics topics to elementary school students [4] was instrumental in imagining and designing the freshman seminar course described in this paper. Puzzles have also been advocated as teaching tools in other disciplines: computer science [5], operations research [6], and biology [7], to name a few examples. However, in the references just cited, and in all other examples known to the author, puzzles are used as pedagogical tools for keeping students interested and engaged in standard courses, rather than as bases for a motivating gateway course. While there is no dearth of effort to inspire students to pursue computer science and engineering careers [8], to the best of the author's knowledge, maintaining student motivation once they have enrolled in a computer science or engineering program has not been addressed via an informal entry-level course.

The puzzling nature of computer engineering problems is best captured by the following two expository articles. In an informative technical news column [9], Lauren Aaronson demonstrates the relationship between the immensely popular Sudoku puzzles and the NP-complete class of computational problems, which arise in practically important application domains such as scheduling, network routing, and gene sequencing. In a similar vein, a recent "Computing Science" column, by Brian Hayes, deals with rearranging train cars in railroad yards [10]. He shows, with great clarity, how the railway mechanisms known as stubs, sidings, and wyes, relate to stacks, queues, and other data structures, and how certain algorithms for sorting data are intimately related to procedures for rearranging train cars. These two sources were used as bases for two of the 10 lectures in the course described herein (see Lectures 7 and 9 in Table I).

TABLE I
LIST OF DISCUSSION TOPICS AND THE ASSOCIATED PUZZLES

	Lecture Title	Lead Puzzle
Lecture 1	Easy, hard, impossible!	Collatz's conjecture
Lecture 2	Placement and routing	Houses and utilities
Lecture 3	Satisfiability	Making change
Lecture 4	Cryptography	Secret messages
Lecture 5	Byzantine generals	Liars and truth tellers
Lecture 6	Binary search	Counterfeit coin
Lecture 7	Task scheduling	Sudoku
Lecture 8	String matching	Word search
Lecture 9	Sorting networks	Rearranging trains
Lecture 10	Malfunction diagnosis	Logical reasoning

III. COURSE PHILOSOPHY AND DESIGN

The short catalog description of the course ECE 1, Ten Puzzling Problems in Computer Engineering, at the University of California, Santa Barbara, reads: "Gaining familiarity with, and motivation to study, the field of computer engineering, through puzzle-like problems that represent a range of challenges facing computer engineers in their daily problem-solving efforts and at the frontiers of research." The course was not conceived as a forum for imparting knowledge of specific facts or methods (things on which the students can later be tested), nor was it meant as a vehicle for honing problem-solving skills [11]. The primary purpose of the course is to motivate and excite. To this end, it was decided to forego homework and exams, and to assign pass/not-pass grades based on attendance. During the spring 2007 and spring 2008 offerings of the course, one absence (in 10 class sessions) was allowed with no consequences to students, while two or three absences required instructor consent for a pass grade, with approval granted based on active class participation or a special oral "exam." The intention was to encourage full class participation, while also allowing for some flexibility to accommodate personal circumstances.

Clearly, there are always some students who view a course that has no formal evaluation as an easy credit, and who come to class with no intention of getting involved. The flip side of this coin is that many highly motivated students can become discouraged by long hours spent on doing homework and by preparation for exams. This tradeoff was made deliberately and with some unease. In the end, it was thought that fun activities in a relaxed classroom setting could contribute more to the course goals of exposing the students to challenging problems, than would formal course requirements. This approach is particularly in tune with the tradition of undergraduate research at the University of California, Santa Barbara (UCSB). It is hoped that the introduction of this freshman seminar will help increase participation in undergraduate research projects by computer engineering students and faculty, beyond the already significant level which is a hallmark of UCSB.

The course handouts are designed to be simple and uncluttered. Each 2-sided single-sheet handout contains the image of four of the lecture slides. The students use the handout as a worksheet during the lecture and can take it away with them to work on the more challenging puzzles outside class. Put together, the ten handout sheets, along with the course outline/schedule, which is handed out during the first lecture and is also available on-line [1], provide a capsule summary of the topics discussed during the quarter and serve as pointers for students to go back to the course's website, during the quarter or afterwards, to pursue the topics at leisure.

IV. The Ten Course Lectures

The format of lectures is as follows. At the beginning of each class session, the instructor introduces a simple puzzle and asks the students to try to solve it. Students are called upon to explain their answers and solution strategies. Next, somewhat more difficult or elaborate versions of the puzzle are introduced and students are again invited to solve them. As the students work on these harder puzzles, hints are revealed on the screen to help them along. The puzzles segment constitutes roughly one-third of the class duration.

In the second segment of the lecture, the instructor provides some background (origin, historical context, variations) on the puzzles and why they are deemed interesting. She or he then introduces general solution methods for the puzzles. The third and final segment of each lecture (roughly one-third of the class duration) is spent discussing why the puzzles are relevant or similar to computer engineering problems and how solution methods for the puzzles can be adapted to these engineering and research endeavors.

Part of the first class session is devoted to introducing the course, describing its requirements, and defining the nature of puzzling problems. The discussion segment of this first lecture is thus shorter than a typical session. Similarly, the tenth lecture is a bit shorter to accommodate the administration of instructor and course evaluation surveys, which is a requirement for every course taught at University of California, Santa Barbara.

A complete list of lecture titles, and the associated lead puzzles, is presented in Table I. Collatz's conjecture, which has as yet neither been proven nor contradicted, is used as the starting point. The conjecture states that the sequence of numbers $x_1, x_2, x_3, \ldots$ always leads to the repeating pattern 4, 2, 1 in a finite number of steps, where x_1 is a chosen initial number and x_{i+1} is obtained from x_i by halving it, when x_i is even, and by tripling it and adding 1, when x_i is odd. In other words, if $e_i = x_i \bmod 2$, then the sequence defined by $x_{i+1} = x_i(1 + 5e_i)/2 + e_i$ eventually leads to 1, regardless of the first number x_1. This problem, when contrasted with other sequences that are quite easily analyzed, shows that appearances can be deceiving and that the brevity or simplicity of a problem's statement does not always foretell an easy solution. The rest of the lectures, all of which are available on-line [1], cover both hardware and software topics, in accordance with the modern meaning of "computer engineering," which entails more than mere hardware design.

V. Outline of a Sample Lecture

Imagine two utility companies and five houses. Is it possible to connect each utility to every house in a manner that the ten lines drawn do not intersect? Students quickly discover that a solution does exist, not only with five houses but also with any number of houses that require connection to two utilities. They also readily determine that the puzzle is symmetrical, that is, it makes no difference whether there are five houses and two utilities, or two houses and five utilities. In graph theoretic terms, the complete bipartite graph $K_{n,2}$ or $K_{2,n}$ is planar (Fig. 1). What about the case of three houses and three utilities? A few minutes of experimentation convinces the students that this classic form of the puzzle (Fig. 2) has no solution, although they may

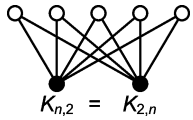

Fig. 1. The houses-and-utilities puzzle, such as this version with 5 houses and 2 utilities, is modeled by a complete bipartite graph.

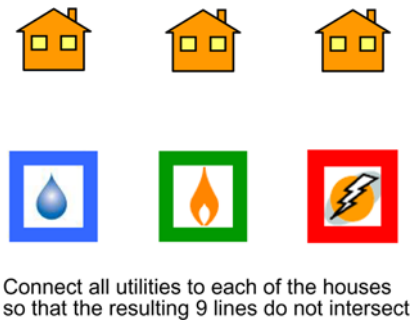

Fig. 2. The classic version of the houses-and-utilities puzzle.

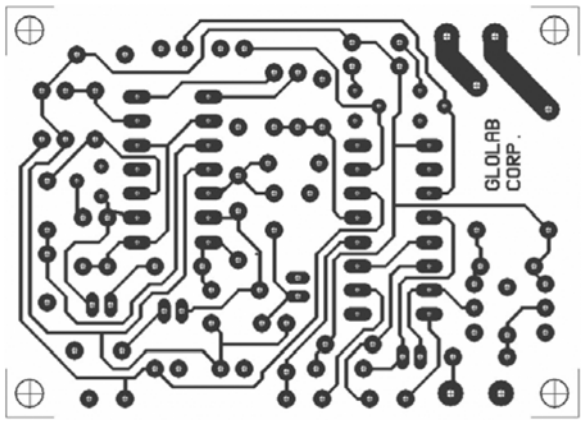

Fig. 3. Connecting utilities to houses with lines that do not cross is akin to connecting points on a circuit board with wires that do not overlap. Modern ICs and circuit boards have multiple layers of metal wiring and are thus not limited by the nonintersection requirement.

not be able to provide a convincing proof; they just know that they keep getting stuck when they try to draw the ninth and final line.

Presentation of a simple and elegant proof for the nonplanarity of $K_{3,3}$ [12], using Euler's formula $v - e + f = 2$ for planar graphs, confirms the intuition that the problem is insoluble. The formula, which relates the number of vertices, edges, and faces in a planar graph, implies that if $K_{3,3}$ (with $v = 6$ and $e = 9$) is planar, it must have five faces. However, each face in a bipartite graph needs at least four edges, thus implying that there must be at least ten edges, where the number 10 is obtained from $4 \times 5/2$, given that each edge is shared by two faces. This clearly contradicts the fact that $K_{3,3}$ has only nine edges.

The last third of the lecture begins with the observation that nonintersecting connections are also required for circuits that must be deposited or "printed" on a surface (Fig. 3). If the 6-vertex graph $K_{3,3}$ is nonplanar, then there is little hope that the very intricate connections of a typical electronic circuit can be drawn without intersecting each other. This observation leads to a discussion of multilayer printed circuits (such as a two-sided

printed-circuit board) and the flexibility provided by more than two layers on modern boards and microchips. Displaying and explaining drawings and photographs of multiple metal layers in microchips and printed-circuit boards concludes the lecture.

VI. Priliminary Evaluation

During its first offering at UCSB in Spring 2007, the course "ECE 1: Ten Puzzling Problems in Computer Engineering" had 41 enrolled students, along with a dozen or so students who just sat in. The enrolled students were primarily freshmen and sophomores, although a few juniors and seniors were also allowed to take the class by special petition. The second offering in Spring 2008 had 36 enrolled students and half a dozen or so auditors. Nearly all enrollees in this second offering were first-year students, although many were classified as sophomores due to transfer credits from advanced high-school courses.

During both offerings, students indicated that they enjoyed the course, particularly because it imposed no homework pressure, deadline, or exam preparation. The students were encouraged to participate in class and to communicate with the instructor outside the class during open office hours and via electronic mail. Class participation exceeded the instructor's expectations based on his past experience with other lower-division courses. This high level of participation is probably a direct result of the puzzle-based approach. Interaction outside the classroom was very limited, possibly due the lack of homework assignments and exams.

The written student comments at the end of the quarter were overwhelmingly positive. Nearly all students showed appreciation for the carefully prepared PowerPoint presentations that were made available on-line. A handful of negative comments concerned the pace of some lectures (too much material, skipping through slides), the long and narrow lecture hall making interaction somewhat difficult, and a dearth of humor.

Even though handouts were distributed in all ten lectures, these might have been even more effective had they been collected in a booklet for distribution at the outset. This booklet would provide the "big picture" needed for greater enjoyment of the course. This big picture is currently provided by the course's website, but having a hard-copy document to carry around is still the preferred mode for most students.

VII. Future Plans

The next offering of this course is scheduled for Spring 2009, when, in addition to updating and fine-tuning the presentation slides, effort will be made to reduce the number of topics covered in each lecture, so as to create greater focus and to increase student participation.

Puzzling problems in computer engineering are not limited to those ten presented in Table I and described in detail on the course's website [1]. Thus, the following additional topics are being considered, either to replace some of the current topics, or for use in creating a pool of lectures from which the instructor can pick ten topics for any given offering of the course. These augmentations would create added flexibility for the instructor, allowing him or her to tailor the course material to the specific focal points of a particular educational program. In the following list, possible mathematical and logical puzzles are listed in square brackets.

- Computational geometry: What becomes of lines, circles, and other shapes when rendered with pixels and how to recover the original forms from digitized versions
- Loss of precision: Seemingly simple computations that produce utterly wrong results
- Secret sharing: How to give parts of a secret to n people so that any m of them can cooperate to discover the secret, whereas any group of $m-1$ cannot do so
- Amdahl's law: Why improving only one aspect of a system's behavior may not lead to significant improvement in its overall performance [River and bridge crossing]
- Predicting the future: How a system can learn from past events in order to make good guesses about the future [Determining the next term in a series]
- Circuit-value problem: Example of a problem that has polynomial-time solution but that cannot be efficiently parallelized [Cooperation in adding 1000-digit numbers]
- Maps and graphs: Problems in digital representation and manipulation of maps of various kinds [Map/graph coloring and graph properties]

Comments and suggestions from instructors who teach similar freshman seminar courses and other interested parties would be most welcome by the author. A formal evaluation of the freshman seminar course described in this paper, along with its outcomes and curricular impacts, will be conducted in future work.

References

[1] B. Parhami, "Ten puzzling problems in computer engineering," in *Website for ECE 1 Freshman Seminar, Dep. Electr. Comput. Eng., Univ. Calif.*, Santa Barbara [Online]. Available: http://www.ece.ucsb.edu/~parhami/ece_001.htm

[2] P. Gnädic, G. Honyek, and K. Riley, *200 Puzzling Physics Problems, With Hints and Solutions*. Cambridge, U.K.: Cambridge Univ. Press, 2001.

[3] J. Parker, "The use of puzzles in teaching mathematics," *Mathematics Teacher*, vol. 48, pp. 218–227, Apr. 1955.

[4] B. Parhami, "Math + fun!," in *Website Containing Presentations on Mathematics and Technology Topics* [Online]. Available: http://www.ece.ucsb.edu/~parhami/math-plus-fun.htm

[5] A. Levitin and M.-A. Papalaskari, "Using puzzles in teaching algorithms," in *Proc. ACM SIGCSE Conf., Comput. Sci. Educ.*, 2002, pp. 292–296.

[6] H. Müller-Merbach, "The role of puzzles in teaching combinatorial programming," in *Combinatorial Programming: Methods and Applications*, B. Roy, Ed. Dordrecht, The Netherlands: D. Reidel, 1975, pp. 379–386.

[7] S. Franklin, M. Peat, and A. Lewis, "Non-traditional interventions to stimulate discussion: The use of games and puzzles," *J. Biolog. Educ.*, vol. 37, no. 2, pp. 79–84, 2003.

[8] A. Y. Akbulut and C. A. Looney, "Inspiring students to pursue computing degrees," *Commun. ACM*, vol. 50, no. 10, pp. 67–71, Oct. 2007.

[9] L. Aaronson, "Sudoku science: A popular puzzle helps researchers dig into deep math," *IEEE Spectrum*, vol. 43, no. 2, pp. 16–17, Feb. 2006.

[10] B. Hayes, "Trains of thought: Computing with locomotives and box cars takes a one-track mind," *Amer. Scient.*, vol. 95, no. 2, pp. 108–113, Mar.-Apr. 2007.

[11] B. Averback, *Problem Solving Through Recreational Mathematics*. New York: Dover, 2000.

[12] J. A. Brady and U. S. R. Murty, *Graph Theory With Applications*. New York: Elsevier, 1976.

Behrooz Parhami (S'70–M'73–SM'78–F'97) received the Ph.D. degree in computer science from the University of California, Los Angeles, in 1973.

Currently, he is a Professor with the Department of Electrical and Computer Engineering, University of California, Santa Barbara. His research deals with parallel architectures and algorithms, computer arithmetic, and reliable computing. In his previous position with Sharif University of Technology, Tehran, Iran from 1974 to 1988, he was also involved in the areas of educational planning, curriculum development, standardization efforts, and technology transfer. His technical publications include more than 250 papers in journals and international conferences, a Persian-language textbook, and an English/Persian glossary of computing terms. Among his latest publications are two graduate-level textbooks on parallel processing (New York: Plenum, 1999) and computer arithmetic (New York: Oxford, 2000), and an introductory textbook on computer architecture (New York: Oxford, 2005).

Prof. Parhami is a Chartered Fellow of the British Computer Society, a Member of the ACM, and a Distinguished Member of the Informatics Society of Iran, for which he served as a founding member and President during 1979–1984. He has held various editorial responsibilities, including a five-year term as Editor of *Computer Report*, a Persian-language computing periodical, and is currently an Associate Editor for both IEEE TRANSACTIONS ON COMPUTERS and IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He also served as Chairman of the IEEE Iran Section (1977–1986) and received the IEEE Centennial Medal in 1984.