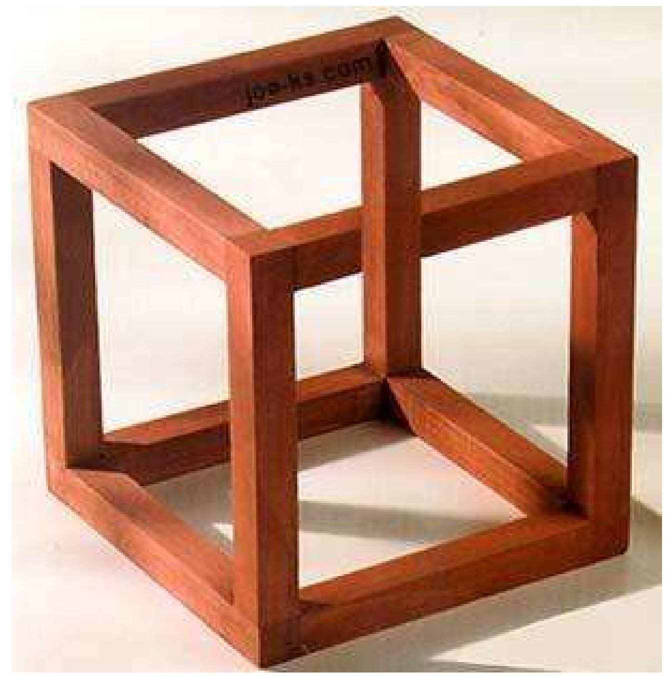
Easy, Hard, Impossible!

A Lecture in CE Freshman Seminar Series:
Puzzling Problems in Computer Engineering

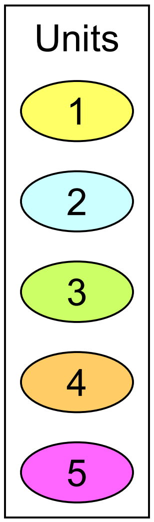


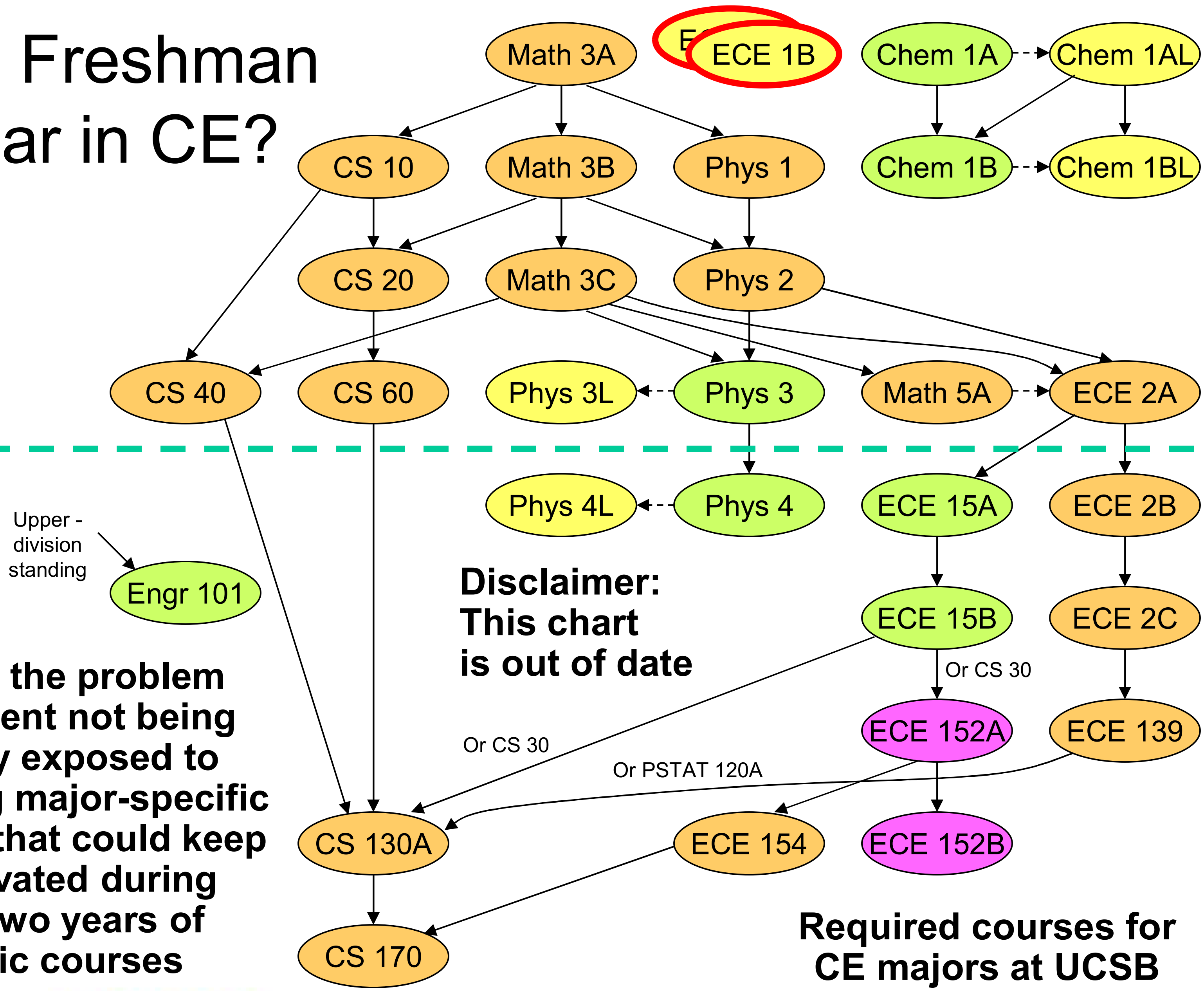
About This Presentation

This presentation belongs to the lecture series entitled "Ten Puzzling Problems in Computer Engineering," devised for a ten-week, one-unit, freshman seminar course by Behrooz Parhami, Professor of Computer Engineering at University of California, Santa Barbara. The material can be used freely in teaching and other educational settings. Unauthorized uses, including any use for financial gain, are prohibited. © Behrooz Parhami

Edition	Released	Revised	Revised	Revised	Revised
First	Apr. 2007	Apr. 2008	Apr. 2009	Apr. 2010	Mar. 2011
		Apr. 2012	Mar. 2015	Mar. 2016	Mar. 2020

Why a Freshman Seminar in CE?





Disclaimer: This chart is out of date

Required courses for CE majors at UCSB

To remedy the problem of CE student not being sufficiently exposed to interesting major-specific problems that could keep them motivated during their first two years of taking basic courses

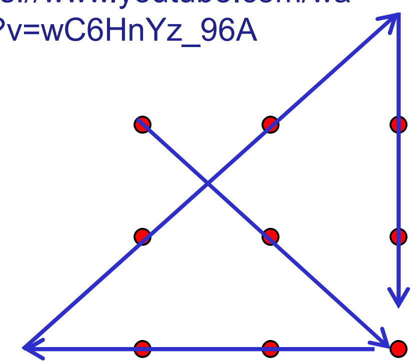
Ten Puzzling Problems in Computer Engineering

What is a puzzling problem?

☞ looks deceptively simple, but ...

☞ appears very difficult, or even impossible, but is readily tamed with the correct insight

Rotate faces until each face is single-colored
https://www.youtube.com/watch?v=wC6HnYz_96A



Connect all dots using four straight lines, without lifting your pen

Many engineering problems are puzzle-like (especially in CE)

Because of a long-standing interest in mathematical puzzles, I designed this course that combines my personal and professional passions

Each lecture starts with one or more puzzles
We will try to solve the puzzles and discuss possible solution methods
I introduce you to CE problems that are related to the puzzles

Course Expectations and Resources

Grading: Pass/Not-Pass, by attendance and class participation

0 absence: Automatic "Pass"

1 absence: "Pass" if you submit a written explanation for the absence; any explanation will do

2 absences: Can earn a "Pass" by taking a final oral exam covering the missed lectures

3 or more absences: Automatic "Not Pass"

Attendance slips distributed at the beginning of class and to those arriving no more than 10 minutes late. Complete and turn in at the end (The next slide lists the special arrangements for spring 2020 offering)

Course website: http://www.ece.ucsb.edu/~parhami/ece_001.htm (PowerPoint and PDF presentations, addresses of relevant websites)

Instructor's office hours for s'20: M 3:00-4:00, W 3:00-5:00, HFH 5155

Course Evaluation During Spring 2020

Grading: Pass/Not-Pass, by attendance and class participation

0 absence: Automatic "Pass"

1 absence: "Pass" if you submit a written explanation for the absence; any explanation will do

2 absences: Can earn a "Pass" by taking a final oral exam covering the missed lectures

3 or more absences: Automatic "Not Pass"

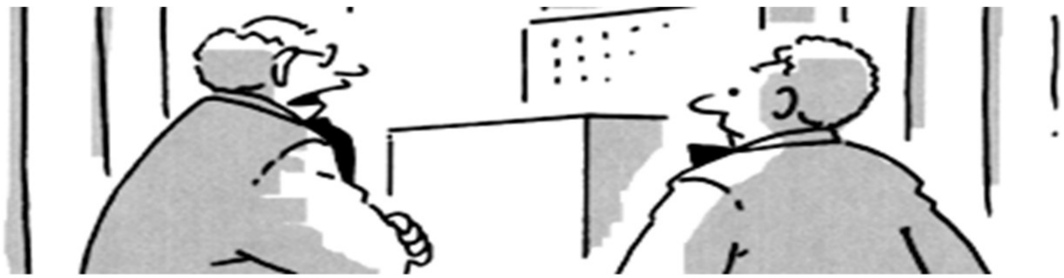
During spring 2020 offering of ECE 1B, "attendance" means e-mailing the answers to 4 questions, Q1-Q4, scattered among the lecture slides (absence means either not sending an e-mail or doing a very poor job)

Send your answers via e-mail: parhami(at)ece(dot)ucsb(dot)edu
Use the subject line: ECE 1B, Spring 2020: Attendance mm/dd

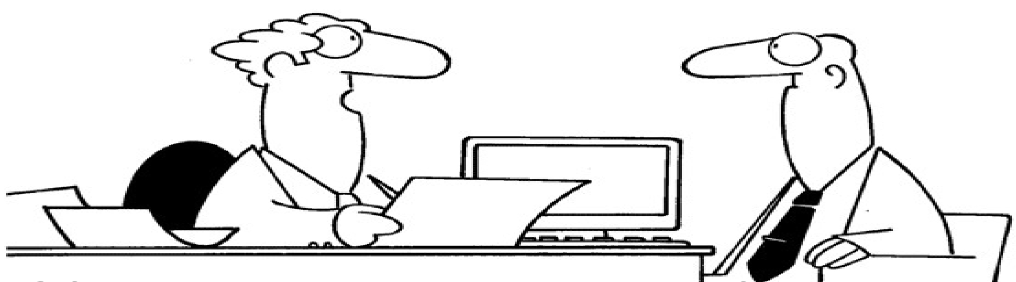
Include A1- A4 to Q1-Q4 in the body of e-mail and attach any needed diagrams as jpg files: Deadline for answers is 48 hrs after each lecture

Now, on to Today's Topic: Easy, Hard, Impossible

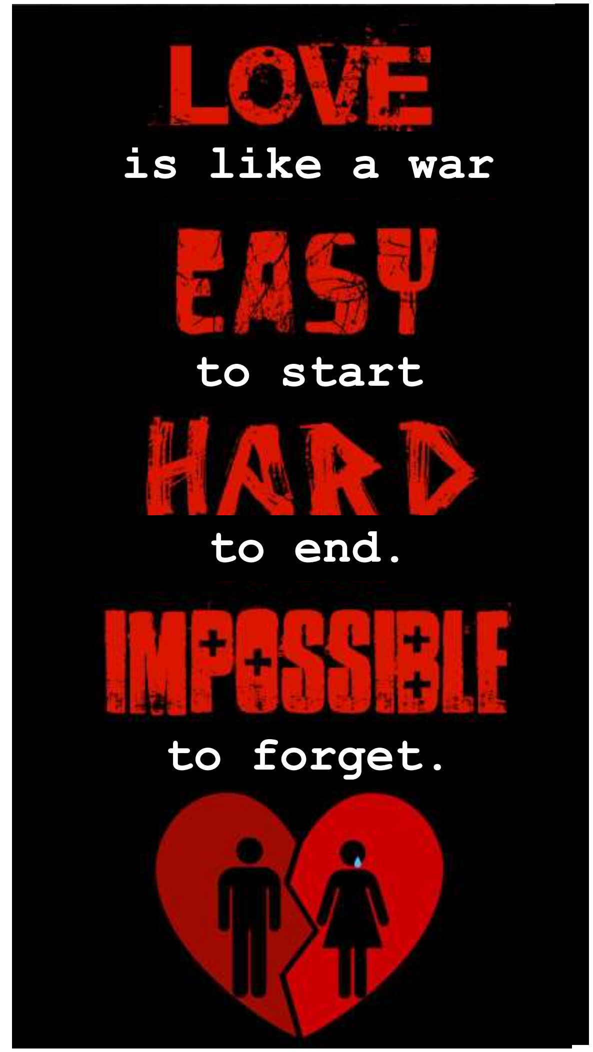
"It's not as easy as it looks!"



"The hardest things to deal with in this job are feelings and lawyers."



"Of course it's impossible to figure out. Why else would they call it the tax *code*?"



Easy: Euclid Sequences

Form a sequence of number pairs (integers) as follows:
Begin with any two positive numbers as the first pair
In each step, the next number pair consists of
(1) the smaller of the current pair of values, and (2) their difference
Stop when the two numbers in the pair become equal

(10, 15) (10, 5) (5, 5)

(22, 6) (6, 16) (6, 10) (6, 4) (4, 2) (2, 2)

(9, 23) (9, 14) (9, 5) (5, 4) (4, 1) (1, 3) (1, 2) (1, 1)

Q1: Why is the process outlined above guaranteed to end?

Challenge: Repeat this process for a few more starting number pairs and see if you can discover something about how the final number pair is related to the starting values

Not So Easy: Fibonacci Sequences

Form a sequence of numbers (integers) as follows:
Begin with any two numbers as the first two elements
In each step, the next number is
the sum of the last two numbers already in the sequence
Stop when you have generated the *j* th number (*j* is given)

5 16 21 37 — $j = 4$

2 0 2 2 4 6 10 16 26 — $j = 9$

1	1	2	3	5	8	13	21	34	55	89	144	$j = 12$
Fib(1)	Fib(2)	Fib(3)	Fib(4)	Fib(5)	Fib(6)	Fib(7)						

Q2: The ratio Fib($j + 1$)/Fib(j) tends to a limit as $j \to \infty$. What is the limit?

Challenge: See if you can find a formula that yields Fib(*j*) directly, that is, without forming the sequence Fib(1) Fib(2) . . . Fib(*j*)

Very Hard: Collatz Sequences

Form a sequence of numbers (integers) as follows:
Begin with a given number
To find the next number in each step,
halve the current number if it is even or triple it and add 1 if it is odd

5 16 8 4 2 1

The pattern 4 2 1 repeats
(5 steps to reach the end)

22 11 34 17 52 26 13 40 20 10 5 . . . 1 (15 steps)

9 28 14 7 22 11 . . . 1 (19 steps)

Challenge: Repeat this process for 27 and some other starting values. See if you can discover something about how various sequences end; i.e., do all sequences end in the same way, fall into several categories, or do not show any overall pattern at all?

Reference: http://en.wikipedia.org/wiki/Collatz_conjecture

Collatz Sequence for 27

Form a sequence of numbers (integers) as follows:
Begin with a given number
To find the next number in each step,
halve the current number if it is even or triple it and add 1 if it is odd

27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, **7288**, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, **9232**, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

111 steps See Wikipedia for a plot of the values vs. step number.
Reference: http://en.wikipedia.org/wiki/Collatz_conjecture

Q3: Find the largest integer whose Collatz sequence has *n* steps.

Easy, Not So Easy, Very Hard

Euclid: Form a sequence of number pairs (integers) as follows:
Begin with any two positive numbers as the first pair
In each step, the next number pair consists of
(1) the smaller of the current pair of values, and (2) their difference
Stop when the two numbers in the pair become equal

Fibonacci: Form a sequence of numbers (integers) as follows:
Begin with any two numbers as the first two elements
In each step, the next number is
the sum of the last two numbers already in the sequence
Stop when you have generated the j th number (j is given)

Collatz: Form a sequence of numbers (integers) as follows:
Begin with a given number
To find the next number in each step,
halve the current number if it is even or triple it and add 1 if it is odd

What Makes a Computational Problem Easy?

Euclid: Form a sequence of number pairs (integers) as follows:
Begin with any two positive numbers as the first pair
In each step, the next number pair consists of
(1) the smaller of the current pair of values, and (2) their difference
Stop when the two numbers in the pair become equal

Algorithm: *gcd*(*x* , *y*), greatest common divisor of *x* and *y*
If $x = y$, then output x and stop
Otherwise, compute $gcd(min(x, y), |x - y|)$

Analysis: Number of steps in the worst case
$(1, n)$ $(1, n - 1)$ $(1, n - 2)$ . . . $(1, 1)$ ($n - 1$ steps)

When the number of steps is a *polynomial function* of the problem size, the problem is considered computationally easy or *tractable*

The idea is that modern computers can perform many billions of computations per second, so even n^2 or n^5 steps may be manageable

Easy Problems: Degrees of Ease

Consider the computation of x mod y (remainder of dividing x by y), where x and y are positive integers

Algorithm: *rem*(*x* , *y*), remainder of dividing *x* by *y*
If $x < y$, then output x and stop
Otherwise, compute *rem*($x - y$, y)

Analysis: Number of steps in the worst case
(n, 1)   (n – 1, 1)   (n – 2, 1)   . . .   (0, 1)

(n steps)

For example, if $n \approx 10^{16}$ (i.e., a 16-digit decimal number), and if each computation step takes 1 ns (10^{-9} s), then the execution time of this algorithm would be $10^{16}/10^{9}$ s = 10^{7} s ≈ 2778 hr ≈ 116 days ≈ 4 mo

Long division would yield the answer much quicker, in time that depends on the number of digits in n (i.e., log n) and not on n itself

Computer scientists and engineers pursue more efficient algorithms

Some Difficult-Looking Problems Are Easy

Fibonacci: Form a sequence of numbers (integers) as follows:
Begin with any two numbers as the first two elements
In each step, the next number is
the sum of the last two numbers already in the sequence
Stop when you have generated the *j* th number (*j* is given)

Algorithm: *Fib*(*j*), the *j*th Fibonacci number

Set $x := 1$ and $y := 1$

Repeat $j - 2$ times: Set $z = x + y$; $x := y$; $y := z$

($j - 2$ steps)

Quicker solution for large *j*

The golden ratio $\left(\frac{1+\sqrt{5}}{2}\right) = 1.61803\ldots$

$$Fib(j) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^j - \left(\frac{1-\sqrt{5}}{2}\right)^j}{\sqrt{5}}$$

In the field of computational complexity, degrees of ease or difficulty are measured with reference to the best known algorithm for the problem

Q4: Fib(*j*) is an integer but its formula contains √5. How is this possible?

Some Easy-Looking Problems Are Difficult

Collatz: Form a sequence of numbers (integers) as follows:
Begin with a given number
To find the next number in each step,
halve the current number if it is even or triple it and add 1 if it is odd

Algorithm: ***Collatz*(*n*), Collatz sequence for *n***
Print n and set $x := n$
While $x > 1$ repeat
If x is even then $x := x/2$ else $x := 3x + 1$
In either case, print the new value of x

Number of steps, or even whether it is always finite, is unknown

Collatz's conjecture: For any starting value, the sequence reaches 1

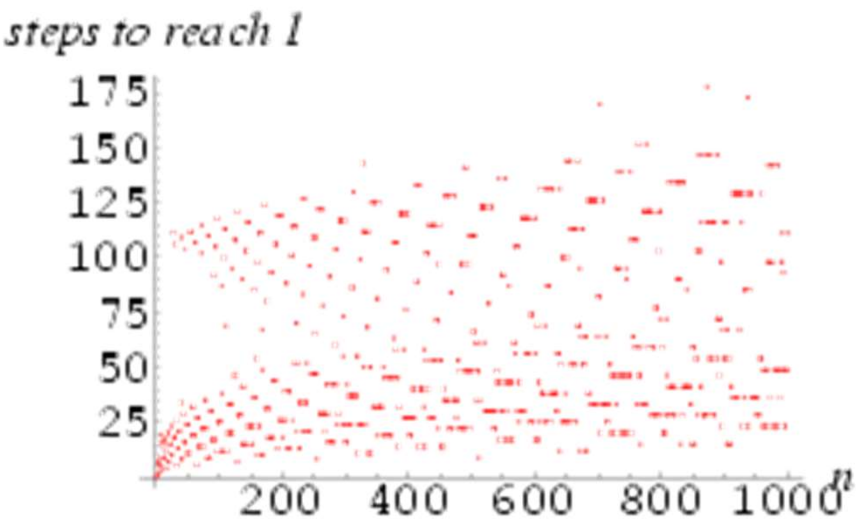
The truth of this conjecture has not yet been verified or contradicted

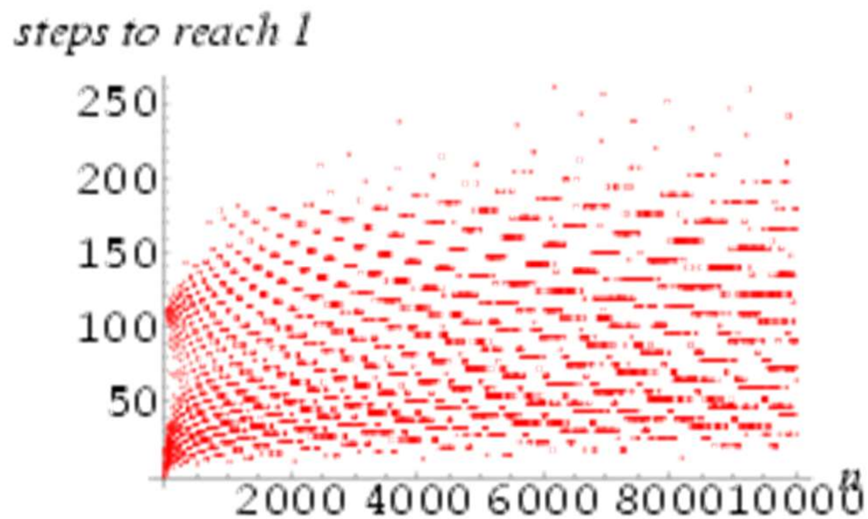
Evidence suggesting truth: experimentally verified for n up to 5.76×10^{18}; next odd number in sequence is on average about ¾ of previous one

Getting a Handle on Difficult Problems

Use the computer to experiment and find solutions for many instances; then, try to generalize from the results and patterns observed

Number of steps for the Collatz sequence to reach 1, as a function of *n*





Diagrams from: Weisstein, Eric W., “Collatz Problem,” *MathWorld* (http://mathworld.wolfram.com/CollatzProblem.html)

In CE, the computer is both an object of study and a tool to help the study

Are Collatz Sequences Useful?

Not directly, but they help us understand the nature of difficult problems

Decision problem: Is there an algorithm to decide whether a given program computing *f*(*n*) will eventually stop for every input *n*?

Program in question → **Decision Algorithm** → Yes/No

Yes → Collatz's conjecture is true

No → Collatz's conjecture is false

Program: *Collatz*(*n*), Collatz sequence for *n*
Print *n* and set *x* := *n*
While *x* > 1 repeat
    If *x* is even then *x* := *x*/2 else *x* := 3*x* + 1
    In either case, print the new value of *x*

Suppose such a decision algorithm exists

One More Easy-Looking Hard Problem

The subset sum problem: Given a set of *n* numbers, determine whether there is a subset whose sum is a given value *x*

S = {3, –4, 32, –25, 6, 10, –9, 50} $x = 22$

Can't do fundamentally better than simply trying all 2^n subsets (exponential time, intractable for even moderately large *n*)

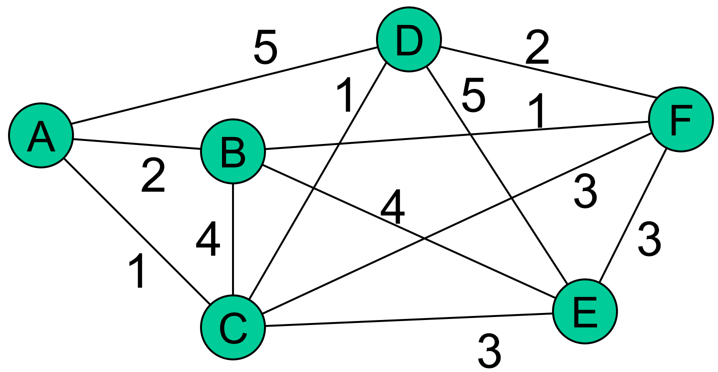
Such problems are known as NP-hard: No one knows how to solve them efficiently, yet there is no proof that they are inherently hard

A specific instance may be readily solvable
The difficulty lies in finding a general solution method

For this instance: Must include 32 or 50 (10 + 6 + 3 = 19 < 22)
Assume 32 is included. Then we need –10 from other terns
32 + (–10) = 32 + (3 – 9 – 4)

Yet Another Easy-Looking Hard Problem

The traveling salesperson problem: Given a set of n cities with known travel cost c_{ij} between cities i and j, find a path of least cost that would take a salesperson through all cities, returning to the starting city



In the worst case, must examine nearly all the $(n - 1)!$ cycles, which would require exponential time

A B F D E C A : 2 + 1 + 2 + 5 + 3 + 1 = 14

A B E F D C A : 2 + 4 + 3 + 2 + 1 + 1 = 13

A C D E F B A : 1 + 1 + 5 + 3 + 1 + 2 = 13

Again, it is finding a general algorithm (solution method) that is difficult, not solving a specific instance of the problem