3-1-2021

Disputing Dijkstra, and birthdays in base 2

Mark Guzdial
University of Michigan

Joel C. Adams
Calvin University

BLOG@CACM

The *Communications* Web site, http://cacm.acm.org, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter
Follow us on Twitter at http://twitter.com/blogCACM

DOI:10.1145/3446806 http://cacm.acm.org/blogs/blog-cacm

Disputing Dijkstra, and Birthdays in Base 2

Mark Guzdial takes issue with Dijkstra's metaphors, while Joel C. Adams considers how birthdays might differ if based on binary numbers.

Mark Guzdial Dijkstra Was Wrong About 'Radical Novelty': Metaphors in CS Education

http://bit.ly/35dg21S

November 30, 2020

Edsger Dijkstra's 1988 paper "On the Cruelty of Really Teaching Computer Science" (in plain text form at https://bit.ly/3b6bFto) is one of the most well-cited papers on computer science (CS) education. It is also wrong. A growing body of recent research explores the very topic that Dijkstra tried to warn us away from—how we learn and teach computer science with metaphor.

According to Google Scholar, Dijkstra's paper has been cited 571 times. In contrast, the most-cited paper in all of the ACM Digital Library papers related to SIGCSE has 412 citations (see data at https://bit.ly/3bae0Ub). Dijkstra's paper has been cited more than any peer-reviewed CS education research. Many of these citations might be citing the "cruelty" paper as a foil, like Owen Astrachan's "On the Cruelty of Really Teaching Computer Science redux" (https://bit.ly/3pSXSKI).

Dijkstra's argument is that computers represent "*radical novelty*." There's nothing like them in human experience, and we cannot use our past experience to understand them. In particular, we shouldn't use metaphors.

> "It is the most common way of trying to cope with novelty: by means of metaphors and analogies we try to link the new to the old, the novel to the familiar. Under sufficiently slow and gradual change, it works reasonably well; in the case of a sharp discontinuity, however, the method breaks down: though we may glorify it with the name 'common sense,' our past experience is no longer relevant, the analogies become too shallow, and the metaphors become more misleading than illuminating. This is the situation that is characteristic for the "radical" novelty.
>
> "Coping with radical novelty requires an orthogonal method. One must consider one's own past, the experiences collected, and the habits formed in it as an unfortunate accident of history, and one has to approach the radical novelty with a blank mind, consciously refusing to try to link it with what is already familiar, because the familiar is hopelessly inadequate."

We now know this is likely impossible. The learning sciences tell us all learning is based on connecting new experiences to previous, through a process called *constructivism* developed by Jean Piaget (see a nice explanation at http://bit.ly/3oiCZZ8). Trying to learn something *without* connection to prior experience inhibits learning. It leads to a phenomenon called *inert knowledge* (http://bit.ly/3oiCZZ8) where you have memorized stuff to pass the test, but you don't really understand and can't really use the knowledge.

I never really thought much about the metaphors we use to learn and teach computer science until the SIGCSE 2014 paper "Metaphors we teach by" (https://bit.ly/3pQ9bn1). CS teachers and students have been ignoring Dijkstra's admonitions all along. They teach with a variety of metaphors, and though all of them have limitations (Dijkstra was right about that), this paper explored how teachers dealt with the breaking point.

The 2019 paper "Identifying embodied metaphors for computing education" (https://bit.ly/3od9uI9) goes a step further to focus on the metaphors that are based on physicality. From a "radical novelty" perspective, this may seem ridiculous—nothing could be less physical than ideas like "arrays" and "control flow." But from a "constructivism" perspective, nothing could be more natural. The basis for all our

experiences are being physical beings in a physical world. When we're dealing with new ideas, we will likely relate them to physical processes.

I am working with Ph.D. student Amber Solomon, who has been studying how teachers teach recursion and how students learn it. She had a paper last summer at the 2020 International Conference of the Learning Sciences about the embodied metaphors that teachers use when teaching recursion (see summary at http://bit.ly/3ogO9xq). Teachers gesture and point, but it's not clear to what. They talk about being "here" and "going." They use language that suggests metaphors like the program "*says*" something.

Solomon is co-advised by Betsy DiSalvo and myself. The three of us have been spending time coding her videos of CS students understanding and modifying programs that use recursion. These are absolutely fascinating, and once you start looking for metaphors and uses of embodiment, you see it everywhere. I particularly like how students shift metaphors, such as talking about the recursive function "going" and then being "stopped" by the base case, then talking about "going down" the stack and execution being different "on the way back up." We know that there is no "down," "back," or "up" in a computer process —these are examples of using concepts from our everyday physical world to understand computational processes.

In 1988 when Dijkstra wrote this piece, cognitive science journals were only about a decade old, and learning sciences was not established until the 1990s. It is understandable that Dijkstra might not have known about constructivism. Today, we know constructivism as the most widely-accepted theory of how humans learn. Using a constructivist lens on learning about computing, we can better understand how to help students use their everyday knowledge as metaphors to learn computer science.

Joel C. Adams Birthday Bit Boundaries

http://bit.ly/38gYp3p

December 1, 2020

My family and I recently celebrated my 63rd birthday. As we were eating dinner that night, one of my sons asked if I had anything special planned for this upcoming year. I hadn't given next year much thought, but since 63_{10} is 111111_2, it occurred to me that this was my last birthday for which my age can be represented in six bits, as it will take seven bits (1000000_2) to represent my age when I turn 64. When I mentioned this, it triggered a surprisingly long and whimsical discussion. (My sons have both graduated with CS degrees and my wife teaches statistics, so...) Some of the points raised during that discussion included:

- We might define a *birthday bit boundary* as a birthday that requires an additional bit to represent one's new age. On my next birthday, I will cross a birthday bit boundary when my age changes from 111111_2 (63) to 1000000_2 (64).
- After birthday #64, my next possible birthday bit boundary would be #128. According to the *Guinness Book of World Records*, the most long-lived person on record was Jean Calment of France, who was 122 when she died in 1997. With no intention of being morbid, barring a medical longevity breakthrough, #64 will almost certainly be the last time I cross a birthday bit boundary.
- Our culture places special emphasis on some birthdays. Often these are multiples of 10 (like 30, 40, 50, 60, ...), presumably because our culture primarily uses decimal numbers. What birthdays would be deemed special if we used a different number system, such as base 12?
- A few other birthdays also receive special attention, such as #12 in some cultures, or "Sweet Sixteen" in popular U.S. culture.
- My previous birthday bit boundary—#32—is quite close to 30, which is commonly regarded as the threshold-age separating youth from non-youth (as in, "*never trust anyone over 30*"). Why not use 32 instead of 30 as that threshold?
- Each birthday bit boundary—#2, #4, #8, #16, #32, #64—is reasonably close to a key threshold in one's life stages. If our culture were based on binary numbers instead of decimal numbers, might we celebrate these birthdays as having special significance?

If we were to celebrate birthday bit boundaries as the entry points to new life stages, the table here shows the result.

Decimal Age	Binary Age	Life Stage
0	0	Infant
1	1	
2	10	Toddler
3	11	
4	100	Child
7	111	
8	1000	Adolescent
15	1111	
16	10000	Adult
31	11111	
32	100000	Middle Age
63	111111	
64	1000000	Senior Citizen
127	1111111	

In this table, the bit-boundary ages map surprisingly well to the start of significant life-stage transitions. For example, the start of adolescence is often associated with the onset of puberty, which can occur anytime in the age-range 8–14. In many U.S. states, teenagers can get their driver licenses at 16, marking their transition to adulthood.

Likewise, in the U.S., 60–65 is commonly thought of as the age at which one becomes a senior citizen, and 65 has long been thought of as the typical "retirement" age. However, 65 seems fairly arbitrary; 64 is obviously close by and might be used instead.

As a result of our family discussion, I've decided to: (i) declare my next birthday (#64) to be one of extra-special significance, and (ii) hold a special party to celebrate my crossing of this final birthday bit boundary. Assuming, of course, that I am still around.

If you have read this far, you may well be thinking that this seems like an especially geeky idea. You may even think this seems like evidence of encroaching elderly eccentricity. This would be difficult to dispute.

However, before you render a final judgment, it is worth noting there is a well-known Beatles song about reaching old age, and the title of that song is not "When I'm Sixty Five," but rather "When I'm Sixty *Four*"!

Mark Guzdial is professor of electrical engineering and computer science in the College of Engineering, and professor of information in the School of Information, of the University of Michigan. **Joel C. Adams** is a professor of computer science at Calvin University.