Computational thinking in mathematics teaching in secondary school

Valérie Batteau and Jana Trgalová

University of Education, Lausanne, Switzerland; valerie.batteau@hepl.ch

Researchers acknowledge the importance of computational thinking for all, not only computer scientists. Strong links between computational thinking and mathematics led to the integration of computational thinking into mathematics curricula in many countries around the world. This ongoing research aims therefore to deepen our understanding of what students' computational thinking is while solving mathematics problems in secondary school. This contribution reports on an experimentation of the famous Sissa's chessboard problem in a Grade 8 class aiming at testing the operational character of a definition of computational thinking based on existing literature.

Keywords: Computational thinking, problem solving, mathematical problem.

Introduction

For almost twenty years, computational thinking (CT) draws attention of researchers and educators. Research points out the importance of CT as "universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" (Wing, 2006, p. 33). Because of the strong links between CT and solving problems in mathematics (Barcelos et al., 2018), CT is part of mathematics curricula in many countries around the world. Recently, the Swiss education policy (Conseil d'Etat, 2019) considers the teaching and learning of CT as a school priority and sets it as a transversal area in mathematics as well as in other disciplines. Our research aims at investigating proximities of CT with mathematical problem-solving skills.

The concept of CT is complex and various conceptualizations have been proposed from discipline-based, psychology-based and education-oriented perspectives (Li et al., 2020). While some of these perspectives view CT in close association with computers and programming, others take a position viewing CT "as a model of thinking that is more about thinking than computing" (idid., p. 4). Research inquires into the links between CT and mathematics (Kallia et al., 2021, Modeste, 2012).

The research study reported in this contribution is part of a project that aims to better understand what CT in students' mathematical activity is. To do this, we have chosen an initial definition of CT based on a literature review, briefly presented in the next section. This definition was then put to the test in a classroom experiment reported in the subsequent sections. The findings discussed in the concluding section highlight a strong intertwinement between CT and mathematical thinking and lead us to re-consider our initial definition of CT.

Computational thinking

In her seminal paper, Wing (2006) gives her viewpoint on what CT is:

> Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. (p. 33).

This definition emphasizes the connection of CT with computer science and highlights its importance with its uses in various fields. It has become a reference for further conceptualizations of CT. In the abundant literature, two main tendencies toward defining CT emerge. One views CT as a set of skills, among which the following are the most widely cited: abstraction, algorithmic thinking, automation, decomposition, debugging, and generalization[1] (Bocconi et al., 2016). The second one considers CT involving key dimensions, such as computational concepts, computational practices, and computational perspectives (Brennan & Resnick, 2012).

Close relations of CT with mathematics are acknowledged by many researchers. For example, Wing (2006, p. 35) claims that "Computer science inherently draws on mathematical thinking, given that, like all sciences, its formal foundations rest on mathematics". The proximity of these two disciplines is one of the reasons why, in many countries around the world, CT has been integrated to mathematics curricula. Yet, Stephens and Kadijevich (2020, p. 119) claim that "studies explicitly linking CT and learning mathematics are rather rare, mostly dealing with areas that are traditionally connected to programming, for example, numbers and operations, algebra, and geometry". Among the studies that attempt to define what characterizes CT in mathematics education, Kallia et al. (2021), drawing on an extensive literature review of 56 papers complemented by a Delphi study, highlight three key aspects of CT in mathematics education:

(1) problem-solving as a fundamental goal of mathematics education in which computational thinking is embedded;
(2) thinking processes that include (but not limited to) *abstraction, decomposition, pattern recognition, algorithmic thinking, modelling, logical and analytical thinking, generalisation and evaluation of solutions and strategies*;
(3) phrasing the solution of a mathematical problem in such a way that it can be transferred / outsourced to another person or a machine (transposition) (p. 179).

Based on this brief literature review, for our research purposes we consider CT in mathematics as a model of thinking that includes skills of abstraction, problem decomposition, data analysis and interpretation, pattern recognition, and modeling aiming at solving a mathematical problem by formalizing a solution in a way that it can be outsourced by a machine. We consider this definition of CT as our interpretative framework, and we question the extent to which it is operational.

Methodology

Our research draws on a qualitative research methodology supported by a collaborative and iterative process of design and implementation of mathematical tasks, involving researchers and teachers (Batteau & Clivaz, 2016; Psycharis et al., 2020).

Context. The project involves two researchers, the authors of this paper, and a mathematics and science teacher in a secondary school in the Lausanne region. The teacher finished her teacher training last year and defended a master thesis on CT in mathematics in secondary school. The first task chosen for the experimentation was designed based on the famous Sissa's chessboard problem. As we show in the next section, this problem has a potential to develop students' CT. To solve the task, the students will need to create a program with Scratch. As they have not programmed previously, a

[1] See (Bocconi et al., 2016, p.18) for definitions of these terms.

preliminary lesson was planned to introduce Scratch and programming to students (in particular the notion of variable, loop, and increment). During this lesson, the students were asked to program how to calculate different factorials (2!, 3!, 4!, 5! then 100!). This preliminary lesson was not recorded as it is not the focus of our research.

Data. The data we collected are of several types: videos of class sessions, students' written and digital productions, screen capture videos, and audio-recording of interviews with students. The data is thus collected with several instruments allowing a triangularization of the analyses, which helps to guarantee the scientific validity of the results obtained.

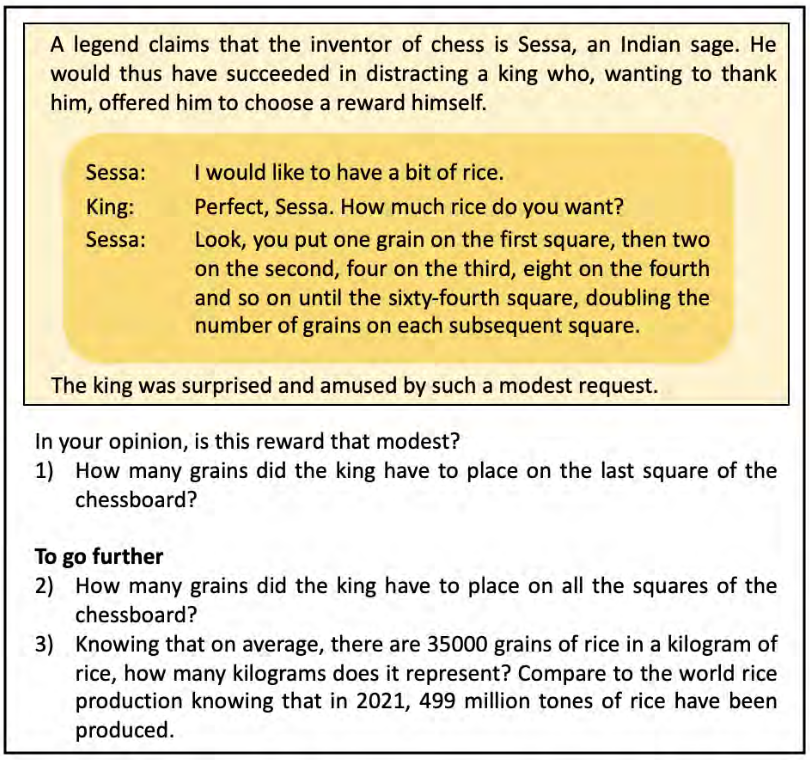
Method. Based on the initial definition of CT, we realize an *a priori* analysis of the Sissa's chessboard problem (see next section) to bring to the fore knowledge and reasoning at stake and identify where and which components of CT the students need to mobilize to solve the problem. Then, we confront this *a priori* analysis with an *a posteriori* (data) analysis aiming at identifying indicators of the components of the CT in the different data (written and digital productions of students, students' activities, students' discourses during interviews).

A priori and *a posteriori* analyses

In this section, we first present the task proposed to Grade 8 students (14-15 years) and its *a priori* analysis. Then we describe the implementation of the task and a few elements of the *a posteriori* analysis (still in progress).

The task and its *a priori* analysis. Figure 1 shows the task built around the well-known 'wheat and chessboard'[2] problem and adapted from the one present in the student's textbook (Livre 10e, Corminboeuf et al., 2012, p. 50). The original task can be found in the chapter related to real numbers, and more precisely powers. The students are thus expected to express the number of the grains of rice on the n-th square of the chessboard as 2^{n-1}; the expected answer to the question 1 is thus 2^{63}. This solution does not call for CT; indeed, the pattern recognition and generalization at stake are rather specific to algebraic thinking (Radford, 2018). To prompt the recourse to CT, we first ask students to find the exact number of grains of rice on the last chessboard square and then the total number of grains of rice on the whole chessboard (question 2), that is needed for the comparison with the world rice production (question 3). We also hypothesize that the exact number of grains of rice ($9\ 223\ 372\ 036\ 854\ 775\ 808$) is more striking than 2^{63} whose rough size might not be obvious for the students. Using a calculator, the students would find a number in a scientific notation ($9.223372037e + 18$). Using Scratch would thus appear as a necessity to calculate the exact number of the rice grains.

[2] https://en.wikipedia.org/w/index.php?title=Wheat_and_chessboard_problem&direction=next&oldid=775634591



A legend claims that the inventor of chess is Sessa, an Indian sage. He would thus have succeeded in distracting a king who, wanting to thank him, offered him to choose a reward himself.

Sessa:	I would like to have a bit of rice.
King:	Perfect, Sessa. How much rice do you want?
Sessa:	Look, you put one grain on the first square, then two on the second, four on the third, eight on the fourth and so on until the sixty-fourth square, doubling the number of grains on each subsequent square.

The king was surprised and amused by such a modest request.

In your opinion, is this reward that modest?

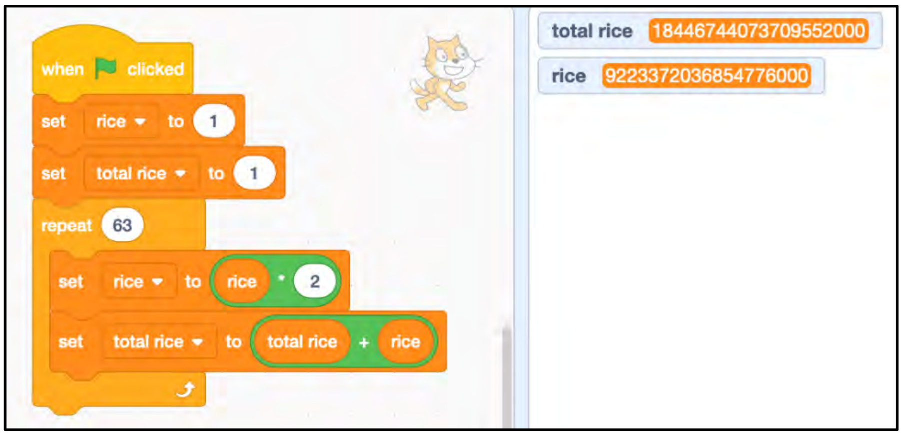
1) How many grains did the king have to place on the last square of the chessboard?

To go further

2) How many grains did the king have to place on all the squares of the chessboard?
3) Knowing that on average, there are 35000 grains of rice in a kilogram of rice, how many kilograms does it represent? Compare to the world rice production knowing that in 2021, 499 million tones of rice have been produced.

Figure 1. The task proposed to students

Since there is no power operator in Scratch, the students will need to (1) define a variable (e.g., rice) and set it to 1; (2) use a loop to double the value of the variable rice 63 times. To calculate the total number of grains of rice on the chessboard, using Scratch is also necessary. A new variable (e.g., total rice) can be defined for summing the amounts of rice grains. A possible code is shown in Figure 2:



```
when 🏴 clicked
set rice ▾ to 1
set total rice ▾ to 1
repeat 63
    set rice ▾ to rice * 2
    set total rice ▾ to total rice + rice
```

```
total rice 18446744073709552000
rice 9223372036854776000
```

Figure 2. Scratch program with the displayed results to questions 1 and 2

The solving process described above mobilizes several components of CT: *data analysis and interpretation* is required to notice that the sought number is obtained by doubling 63 times the previous one starting from 1, which leads to a *recognition of the pattern* that can be expressed in various ways, yielding different mathematical models (see Table 1). These models need to be *formalized in Scratch language* to obtain a solution.

	Procedural expression		Structural expression
Chessboard square	Additive model	Multiplicative model	Model with powers
1	1	1	$2^0 = 1$
2	$1 + 1 = 2$	$1 \times 2 = 2$	$2^1 = 2$
3	$2 + 2 = 4$	$2 \times 2 = 4$	$2^2 = 4$
…	…	…	…
64	$2^{62} + 2^{62}$	$2 \times 2 \times \ldots \times 2$	2^{63}

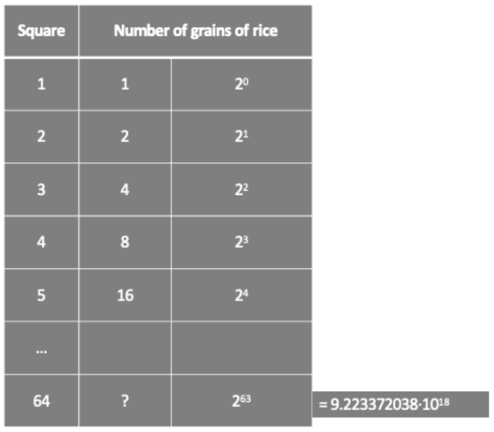
Table 1. Different mathematical models allowing to solve the task

Moreover, the *evaluation of the program* will be necessary to ensure the correctness of the obtained solution (e.g., by running the program for small square numbers for which the number of grains is known). *Debugging* might be required in the case that the obtained number is not correct. These are essential components of CT that remain implicit in our definition. To conclude, the *a priori* analysis of the task let us think that it has a potential to develop CT in students.

In what follows, we describe classroom implementation of the task and a preliminary data analysis.

Implementation of the task. The task was implemented in January 2023 in one Grade 8 class comprising 17 students. They worked in pairs (and one group of three students) and had at their disposal hand-held calculators and one computer per group of students with Scratch (and screen capture tool). They have previously been introduced to Scratch and programming, in particular the notions of variable, loop and incrementation, during one lesson in December 2022.

The lesson lasted over two sessions of 45 minutes each, with a 5-minute break in between. The students started by reading individually the text of the task (2'). During a short whole-class discussion (1'26''), the teacher made sure that all students understood what was expected from them and she insisted on the nature of the result – exact number of grains of rice on the last square of the chessboard. The students were solving the task first using paper and pencil and a calculator (14'20''). The teacher then organized a whole-class discussion about their findings that were summarized in a table drawn by the teacher herself on the blackboard (Fig. 3).

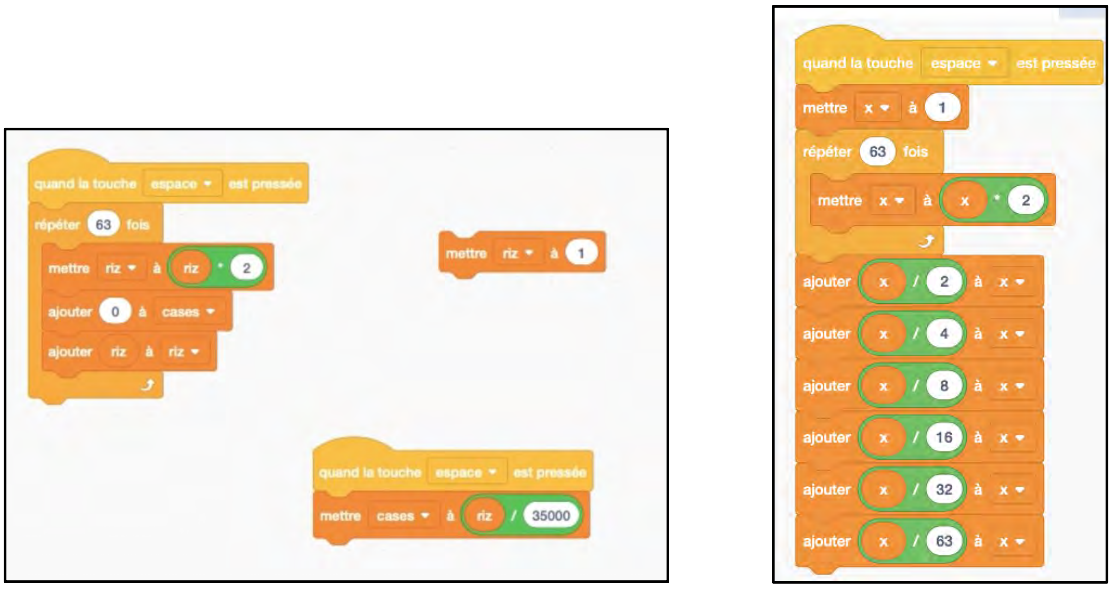


Square	Number of grains of rice		
1	1	2^0	
2	2	2^1	
3	4	2^2	
4	8	2^3	
5	16	2^4	
…			
64	?	2^{63}	$= 9.223372038 \cdot 10^{18}$

Figure 3. A transcript of the synthesis drawn by the teacher on the blackboard (our translation)

(Preliminary) *a posteriori* analysis. All eight groups of students succeeded in coding to obtain the correct answer to the question 1. Six groups used multiplication by 2 (using the command "set rice to rice ×2" – see Figure 1), and two groups used addition (using the command "change rice by rice"). Hence, all students succeeded in mobilizing the following components of CT: *data analysis and interpretation*, *modeling* with multiplicative or additive models, *pattern recognition* (multiplicative or additive) and *formalizing a solution* in a way that it can be outsourced by a machine.

Question 2 was correctly solved by four groups. Two other groups spent too much time on question 1 and had no time left for the next one. The two remaining groups failed in solving the question for different reasons. One group omitted the initialization of the variables, which, by default, are set to 0 (Fig. 4, left). This might indicate an insufficient knowledge of the programming environment and the interference of the mathematical meaning of a variable for which an initialization is not necessary.



```
quand la touche espace est pressée
répéter 63 fois
    mettre riz à (riz * 2)
    ajouter 0 à cases
    ajouter riz à riz

mettre riz à 1

quand la touche espace est pressée
mettre cases à (riz / 35000)
```

```
quand la touche espace est pressée
mettre x à 1
répéter 63 fois
    mettre x à (x * 2)
ajouter (x / 2) à x
ajouter (x / 4) à x
ajouter (x / 8) à x
ajouter (x / 16) à x
ajouter (x / 32) à x
ajouter (x / 63) à x
```

Figure 4. Codes produced by two groups of students

In the other group, the students did not take time to reflect on how to calculate the total amount of grains of rice before coding and kept working with Scratch. These students thus failed in *modeling* due to a lack of *data analysis* in the question 2. The code in Figure 4 (right) shows one of their unsuccessful trials. This might be an example of what Artigue (1997, p. 162) calls a "fishing phenomenon" that she observed in students working with digital technologies: they experiment, without worrying about their organization or their control, hoping that in a reasonable time, they will obtain something interesting.

Discussion and conclusion

Our findings show that all students succeeded solving the question 1, however some failed in solving the question 2. In the case of one group, the reason clearly was the omission of mathematical modelling before coding. Although we identified several components of CT in students' productions, it does not seem enough to conclude about its development in the students' activity. In particular, the failure in solving the problem (at least in one group) cannot be attributed to an insufficient CT. Mathematical and computational thinking seem to be too intertwined in our initial definition making it rather difficult to distinguish between the two modes of thinking when solving the problem. In addition, our definition relies on a set of skills, such as data analysis and interpretation, pattern recognition, abstraction, or generalization, that might have different meanings in mathematics and in computer science. Operationalizing the CT definition therefore requires further clarification of the skills involved in it, and of their relations with mathematics and computer science.

The difference between mathematics and computer science resides also in the meaning of some core concepts that are shared by both fields. The concept of variable is an example; its different meanings in mathematics and in computer science are highlighted by Heck (2002). Our findings show, in one group, a possible interference of the mathematical meaning of the concept of variable used in the digital environment: like in mathematics, the students do not think to assign an initial value to their variables.

These differences in mathematical and computational approaches to solving problems lead us to envision a characterization of the activity carried out in two work-spaces: a mathematical one, in the sense of (Kuzniak, 2011), and a computational one that would need to be introduced.

In our project, we plan to focus also on teachers’ practices that support students' development of computational thinking particularly during collective phases of the teaching, such as whole-class discussions. The course of the lesson and links between different lesson phases in mathematics and in the digital environment need to be explicit: it is necessary to ask students to model the problem in mathematics before moving on to the digital environment. The “fishing phenomenon” (Artigue, 1997) or a trial-and-error loop are signs of an over-investment in programming with respect to other tasks related to problem-solving (Chevalier et al., 2020), which can hinder the development of computational thinking. We will therefore search for elements of teachers’ practices that avoid these phenomena and that promote the development of students’ computational thinking.

References

Artigue, M. (1997). Le Logiciel ‘Dérive' comme révélateur de phénomènes didactiques liés à l'utilisation d'environnements informatiques pour l'apprentissage [‘Dérive' software as a developer of didactic phenomena linked to the use of computer environments for learning]. *Educational Studies in Mathematics, 33*(2), 133–169.

Barcelos, T. S., Munoz, R., Villarroel, R., Merino, E., & Silveira, I. F. (2018). Mathematics Learning through Computational Thinking Activities: A Systematic Literature Review. *Journal of Universal Computer Science, 24*(7), 815–845. https://doi.org/10.3217/jucs-024-07-0815

Batteau, V., & Clivaz, S. (2016). Le dispositif de lesson study: travail autour d'une leçon de numération [The lesson study process: a work about a lesson of numeration]. *Grand N, 98*, 27–48. http://www-irem.ujf-grenoble.fr/spip/spip.php?rubrique21&num=98

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education - Implications for policy and practice*. EUR 28295 EN. https://doi.org/https://op.europa.eu/o/opportal-service/download-handler?identifier=093eadcc-c820-11e6-a6db-01aa75ed71a1&format=pdf&language=en&productionSystem=cellar&part=

Brennan, K., & Resnick, M. (2012). New Frameworks for Studying and Assessing the Development of Computational Thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association* (Vol. 1). http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: a model for creative computational problem solving. *International Journal of STEM Education, 7*(1), 39. https://doi.org/10.1186/s40594-020-00238-z

Corminboeuf, I., Hostettler, T., Odiet, D., & Mante, M. (2012). *Mathématiques 10*. LEP Editions Loisirs et pédagogie et CIIP Conférence intercantonale de l'instruction publique de la Suisse romande et du Tessin.

Heck, A. (2001). Variables in Computer Algebra, Mathematics and Science. *International Journal of Computer Algebra in Mathematics Education*, *8*(3), 195–221.

Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education, 23*(2), 159–187.

Kuzniak, A. (2011). L'espace mathématique de travail et ses genèses [The mathematical workspace and its genesis]. *Annales de didactique des mathématiques et de sciences cognitives, 16*, 9–24.

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020, 2020/05/18). Computational Thinking Is More about Thinking than Computing. *Journal for STEM Education Research, 3*, 1–18. https://doi.org/10.1007/s41979-020-00030-2

Psycharis, G., Trgalová, J., Alturkmani, M. D., Kalogeria, E., Latsi, M., & Roubin, S. (2020). Studying primary and secondary teachers' collaborative design of resources for algebra. In H. Borko, & D. Potari (Eds.), *Teachers of Mathematics Working and Learning in Collaborative Groups. ICMI-Study 25 Conference Proceedings* (pp. 668-675). University of Lisbon.

Radford, L. (2018). The emergence of symbolic algebraic thinking in primary school. In C. Kieran (Ed.), *Teaching and learning algebraic thinking with 5- to 12-year-olds: The global evolution of an emerging field of research and practice* (pp. 3–25). Springer.

Stephens, M., & Kadijevich, D. M. (2020). Computational/Algorithmic Thinking. In S. Lerman (Ed.), *Encyclopedia of Mathematics Education* (pp. 117–127). Springer Nature. https://doi.org/https://doi.org/10.1007/978-3-030-15789-0

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215