Computational Thinking: The New Buzz

R RAMANUJAM

I. The Buzz

As someone working on problems in computer science in the 1980's, I often used to be asked: *What languages do you work in?* I would typically answer, deliberately misunderstanding the question: *Mostly in English, sometimes also in Tamil.* In those days, working with computers meant writing programs in Fortran or Cobol or C, and that was what the questioner was asking about. My answer was about the programming language being irrelevant, the underlying concepts being more important. In fact, a more precise but entirely obscure answer would have been *first order logic*, and to a lesser extent, algebra, these being the languages for abstract reasoning about computation.

All this is to point out that the public perception of computing and computer science may not reflect the *thinking* that underlies these disciplinary domains. (This is quite natural; the public perception of methods used by electrical engineers or archaeologists is unlikely to be accurate either.) The increasing impact of computers on modern living is not necessarily a reason to expect such understanding either: people consult doctors all the time but do not expect to understand medical diagnosis and prescription.

It is when there is advocacy of such "disciplinary thought" in school education that it becomes important to examine such thought, and when it comes to school education, public perception and engagement is critical. Over the last decade there has been an increasingly vocal demand in many countries that Computational Thinking (CT) be a part of the school curriculum. In India, the National Education Policy 2020 (NEP) has not only given importance to CT, but has also coupled it with

Keywords: School mathematics, computational thinking, problem solving, decomposition, pattern recognition, abstraction, algorithm

mathematical thinking. While this has generated quite a buzz in the country, it is quite unclear whether there is a clear perception among the community of educators and teachers what CT is about, why it is being coupled with mathematical thinking at all, and whether promoting CT in schools is necessary or even desirable.

It is in this context that this article seeks to raise and address the following questions:

- Is the omnipresence of computers in modern life sufficient reason to promote (a) learning computing in school, and (b) learning CT in school?
- What does CT have to do with the role of digital technology in the classroom? Given the current massive inequality in the country in the digital space, isn't promoting CT and digital technology going to further deepen the social divide?
- What does CT have to do with learning computer programming, and when should children start learning coding?
- What does CT have to do with school mathematics? With an already crowded mathematics curriculum, are we increasing the burden on students and teachers with additional themes?
- Is the Indian education system equipped to take up CT in schools?

Indeed, this list of questions is not exhaustive, and more questions will arise at all levels of implementation of the NEP in the years ahead. However, a national policy needs to answer these fundamental questions, and provide clarity of direction especially to the teaching community.

II. CT in the NEP

In India, computer science as a discipline is taught principally in the universities, with some preparation for it at the higher secondary level. In the first 10 years of school, so-called *computer classes* have tended to be mainly on usage of computers, platforms and the Internet. Even this is principally in urban private schools; the massive government school system typically introduces computer usage only at the secondary or higher secondary school level.

All this is set to change, with the advent of the NEP, which advocates computational thinking and coding throughout the school years. The relevant item from the NEP is worth quoting:

> 4.25. It is recognized that mathematics and mathematical thinking will be very important for India's future and India's leadership role in the numerous upcoming fields and professions that will involve artificial intelligence, machine learning, and data science, etc. Thus, mathematics and computational thinking will be given increased emphasis throughout the school years, starting with the foundational stage, through a variety of innovative methods, including the regular use of puzzles and games that make mathematical thinking more enjoyable and engaging. Activities involving coding will be introduced in Middle Stage.

The coupling of mathematical and computational thinking is significant, since this suggests completely doing away with the current model of "computer classes" and moving over to teaching the science underlying computing, the emphasis being on thinking. This has important implication for mathematics education as well, shifting the focus from learning "operations", formulas and procedures (to solve equations, etc.) to learning a way of thinking.

On the other hand, the justification for doing so, according to the passage, stems from the importance of mathematics and computational thinking for "upcoming fields and professions such as artificial intelligence, machine learning, and data science, etc." Moreover, coding is advocated from middle school onwards. One can then be pardoned for thinking that the advocacy of CT is merely pandering to fashion, to what is currently considered important in computer science, and has to do with coding. (Indeed, some people have already started advocating the teaching of artificial intelligence and data science at high school!)

The reference to "India's leadership role" further raises doubt: school curricula should be decided by the aims of school education, and children should not be burdened with the responsibility of nationalist priorities. Again, the promotion of CT seems instrumental rather than essential as an educational objective.

This is the only mention of the phrase "computational thinking" in the NEP document. On the other hand, the NEP strongly advocates the use of digital technology in classrooms, devoting entire sections to it.

Meanwhile, the country suffered a huge disruption in school education due to the pandemic, when online education (accessible only to the elite) brought educational technology into prominence. This has led to further confusion, conflating the role of digital technology in the classroom with CT and teaching programming.

All this has led a large section of teachers into believing that introduction of CT in schools means the use of digital tools in classrooms and teaching programming from an early age, with perhaps "data science" and "artificial intelligence" as new subjects.

There is a real *danger* that this might indeed be what the NEP's promotion of CT might end up as, in its implementation: promotion of digital tools and an emphasis on teaching coding from an early age. That would be a sorry consequence indeed.

III. What is CT?

Seymour Papert ([4]), an American computer scientist who pioneered computer programming activities for children, coined the term 'Computational Thinking' in 1980. The phrase "computational thinking" appears in Mindstorms (p.182) but without elaboration. We find a detailed exposition of the idea in his 1996 paper [5] - "An exploration in the space of mathematics education". In it he offers the principle of *Object before Operation:* giving mathematical ideas a "thing-like representation" (using the micro-world of programs) helps in thinking about them.

Jeannette Wing, another American computer scientist, popularised the term in 2006 ([6], [7]). She labels it an "attitude and skill set" that everyone can learn and use. Simply phrased, CT is a process that enables us to take a complex problem, understand it and develop possible solutions in a way that a computer, a human, or both, can understand how to implement the solution. If "mathematical thinking" is thinking like a mathematician, for Wing, "computational thinking" is thinking like a computer scientist.

What is critical here is the last part: why should solutions be developed in such a way that a computer can implement them? What do we mean by computer here? Which computer? What capabilities do we assume the computer to have? Asked in another way, how is it different from a human being implementing the solution?

It should be noted here that we do not mean any specific computer with specific capabilities here, but an idealised one. The important property of mechanical computation is that it does not get bored by repetition or start making mistakes. For a calculator adding five 3 digit numbers is no different an effort as adding 500 numbers, some in crores and some in thousands.

These are the two core elements to consider here: data size, and repetitiveness of algorithms. The essential property of computations is *scaling*. Once we devise such procedures, broken up into sufficiently simple steps, they scale up as necessary.

The characteristics of CT, according to Wing, are: *decomposition, pattern recognition, abstraction,* and *algorithms*.

- Decomposition lets us break up the complex tasks into subtasks, then each subtask into sub-subtasks, etc., until each is simple enough to carry out directly.
- While doing so we often find that some tasks come up again and again, perhaps with slight differences; then we consider them as instances of the same task, perhaps with a parameter

capturing the change. These are the processes of pattern recognition and abstraction.

- Algorithms are stepwise procedures that sequence the subtasks in the 'best possible' manner. CT includes not only such a methodology for problem solving but also ways of comparing solutions and evaluating them.

As an everyday example, consider preparing dinner for 4 persons. There is a significant amount of planning involved. Unless we decide a menu, we cannot shop for the ingredients. However, some of the ingredients may not be available, and hence the menu may need to be changed. Once we have the ingredients and the necessary kitchen appliances, if we have two persons cooking, we decide which tasks are to be done in what order, hopefully without one having to wait unduly for the other. Some tasks have to necessarily come earlier than others.

The recipes need to be clear and have to be followed carefully. Safety has to be maintained all along, and some "runtime checks" need to be done for both safety and taste. If, at the end of all this, you think you have a reliable procedure for making dinner for 4, consider how it would scale for a dinner for 20, and one for 100 guests (in a wedding). Further, compare the procedure with another person's recipes for the same menu, or for different menus. The processes involved in all these illustrate CT.

This example serves to illustrate that computational thinking is relevant in everyday contexts. The notion of "computer" here is abstract and does not refer to any particular machine. Likewise, a recipe is not a program written in a programming language but serves a communicative purpose. Indeed this view of CT does not need electronic devices at all and we can speak of *computer science "unplugged"!* ([1])

Thus the central element of computational thinking is **reasoning about procedures**. It is less important to know many algorithms and procedures, or to write specific code, than to be able to design procedures and algorithms, to be able to reason about how they work, assess their performance, to explore alternative methods and compare them. It is less important to know definitions of data structures than to understand how data can be organised in multiple ways, how this affects procedures that use them and which organisation suits which requirement.

From this viewpoint, computation assumes focus rather than computing devices, and developing thinking that *underlies* computation becomes the educational goal. Such thinking helps the student understand data organisation, scaling, assessment of procedures and their comparison, iteration and modular abstraction, independent of whether solutions are sought by computer or implemented on computing platforms. An example of this would be knowledge of multiple procedures for integer multiplication or division, and an understanding of which is best used when.

When we ask a child to perform the addition 53 + 28 + 47, it is quite alright to add from left to right (or from top to bottom, placing the numbers vertically, as children are used to). But it is computational intuition that suggests grouping as (53 + 47) + 28. This gives us the solution not only faster, but more conveniently, since we are used to the decimal system and multiples of 10 carry meaning for us. As "computers" this reformulation is easier on us. That commutativity of addition allows this is the mathematical understanding that underlies such computational thinking.

At the risk of belabouring the point, consider solving the equation: $2(x + 1) + 3(x + 1) = 10$. Again, there is no harm in expanding brackets, carrying like terms involving x to the left, other terms to the right, and then divide, as a standard technique. However, algebraic computation suggests to us that the equation can be rewritten as $5(x + 1) = 10$, giving the solution immediately. Once again, it is not about speed, but about facility: we are the ones computing here; so we consider different methods of computation and choose which one suits us best.

Consider a question such as: which grows faster, $2x^2 - 50$, or $x^2 + 100$? The awareness that as x increases, adding or subtracting a constant amount will not matter is essentially computational intuition, as also the fact that the function $2x^2$ always dominates x^2. The mathematical justification underlying such intuition can be provided by computing the derivatives and comparing them.

What is being stressed here is that computational thinking involves not only devising procedures and (re-)organizing data appropriately but also consideration of alternative procedures and choosing the "best" among them, articulated according to some criterion. Then **reasoning about procedures** assumes central importance.

While the foregoing may be articulated as the essential meaning of CT, the superficial meaning, relating to the use of computers, has relevance as well. These objectives of CT education relate to use of computing platforms, tools and devices, and giving the students not only mastery over their use, but also create in them a predisposition to identify and utilise educational contexts in which the use of computation can help, and employ computations accordingly. An example of this would be generation of data according to some distribution to examine a probabilistic proposition (such as in the Birthday Problem), or plotting the trajectory of a ball in accordance with Newton's laws. We are led to these objectives principally by "compulsions of the day": since such tools are prevalent and easy to use. Educators must concern themselves both with their "right" and safe use, and with use of such opportunities to enrich educational practices. If we are seen to lessen the emphasis on use of such digital tools in pedagogy, it is not because we consider them less important or irrelevant to CT. Set against the danger of CT being entirely reduced to digital tool use (which does not seem merely hypothetical right now), we have tended to stress on reasoning about procedures being the essential meaning of CT.

IV. Curricular components

What does this understanding of CT imply for school education? Teaching CT in school then would include the following components. These relate not only to CT education in itself but also enhanced educational opportunities provided by CT. (Admittedly, they are not fully independent components and admit some overlap.)

1. **Scaling:** Systematic listing and counting of relevant parameters and verifying that all have been counted is essential for the transition from additive reasoning to multiplicative reasoning. This also paves the way for functional variation, and use of symmetries for counting. Comprehending large magnitudes by scaling small ones is a good way of managing complexity.
2. **Iteration:** Looking for patterns, finding a mechanism for pattern generation and modification, and visualisation of new patterns are all not only pleasant processes but also provide a link between aesthetics and formal reasoning. Understanding the power of iterating simple rules creates a foundation, not only for computation, but also for understanding the dynamics of systems.
3. **Data organisation:** The term "data handling" is familiar in school education but ends up only as graphical depiction of data and computing numerical summaries. But data can be represented in multiple ways, and which one is to be chosen when depends on the use such data is to be put to. Moreover, storage and retrieval of data requires memory structures. Designing such data organisation is neatly coupled with understanding of scaling and iterative data access.
4. **Modelling:** Discrete modelling of problems from real-life situations is largely unfamiliar territory in schools. Discrete structures like lists, trees, maps, graphs, lattices and networks arise naturally and provide abstract problem spaces for computation. Working with concrete representations of such structures early on can help in creating mental models for later facility with such models for computational abstractions.

5. **Algorithms:** Starting from two-digit addition in arithmetic, school education provides a variety of procedures for students to learn – so much so that mathematics or science education often degenerates into a mere memorisation of pre-set procedures to be enacted on specific numerical data. *Following* algorithms with a view to understanding them is no doubt desirable, but *devising* procedures is at the heart of computational thinking. This requires a facility with procedures, reasoning about them, consideration of procedural alternatives and selection among them based on a clear rationale.
6. **Programming:** Concrete implementation of data organisation and algorithms on specified platforms to solve given problems is an essential skill. Coding, when accompanied by a feel for program structure, can be exhilarating while coding as translation of informal computing into a given formal language can be painful. Hence creating a good disposition for programming early on is essential for achieving eventual fluency.
7. **Devices:** Computers, smartphones and other devices provide platforms and tools for computation. Children need to learn purposive use of these tools, and develop mastery over them. At the same time, the challenges that such use may pose, due to physical, emotional and intellectual development of children need to be carefully considered. In a country of stark economic inequalities, access to such devices and platforms cannot be taken for granted either. Thus, the guiding principle in this regard has to be safe and nuanced usage based on need.
8. **Social connectivity:** CT provides a unique opportunity for viewing multiple social structures, their communicational infrastructure, identity formation, etc., and thus forges new links between mathematical formalism and society. As of now, even as students are engrossed in social media and lead parallel lives, their educational potential lies largely unaddressed. Importantly, the possibilities of student communities breaking language, regional and other barriers, working together on data creation and algorithm design need to be examined carefully.
9. **Simulation and visualisation:** This has perhaps been the oldest use of CT in school, learning to plot graphs of functions. However, once students understand the computational basis of visualisation and simulation tools, the tools can greatly expand their horizon of exploration across disciplines. (Consider students playing around with bonding structure of atoms in molecules.)

All these components would not carry equal weightage across the curriculum or across the stages, and it is the task of syllabus designers to spell out the weightage provided to each component at every stage.

V. Examples

Educational opportunities for CT already abound in the existing school curriculum, across the stages. In the primary and upper primary stages of schooling such opportunities are principally found in mathematics education, with a more expansive range across disciplines of study in the secondary stage.

We have already referred to reordering and regrouping techniques that often go under the rubric of "mental math." There are also many opportunities for reasoning about counting procedures, You are in a hall where a wedding is taking place with lots of people, anywhere between 100 and 150. How would you actually count how many there are? How would you know you have counted them all? How would you be able to verify whether your answer is correct?

For small children, counting out 20 seeds is sufficient to provide challenges. Counting silently is different from calling out, but why? We can also watch for bunching and grouping, providing for natural data organization. When we ask the child to move aside 15 of them, we can see if she needs to re-start from the beginning.

Consider the question: how many pairs of positive integers add up to 17? Surely there are multiple ways of listing such pairs, but reasoning involves employing some system of listing in order to be sure that all pairs are counted, and each pair counted only once. When a primary school child offers a 'quick' way to add 10, 100, etc., to a given number, and knows that this way is specific only to these numbers, he is employing CT in context. Finding multiple "reasonable" routes between two places on a map, or considering different arrangements of letters, or forming aesthetically pleasing patterns with beads of different colours, can be excellent exercises in CT as well.

There are many opportunities for data representation. Consider a village in Maharashtra where 61 families are Marathi speaking, 13 speak Kannada, 12 speak only Hindi, 8 are Tamilian, 5 are Gujarati and there is one lone Bengali family. One boring table will surely suffice. But suppose that we use one flag for each family, a colour representing a linguistic group. How would we depict this information? The flags can even be organized in a 10×10 grid, but a chaotic distribution of colours does not help. Once we group colours together in the grid, we suddenly get a great deal of visual information, not only about how numerous a group is, but also the relative size of groups in the village. The histogram, presented then, elucidates this structure further. The point here is not so much about using the histogram to give numerical answers to questions, but to consider alternative data representations and arrive at the one that is most felicitous for answering questions. This is at the heart of data structuring in computational thinking, and prepares students for wonderfully creative notions such as *codes* (and error correction) that can come later on.

Iteration, essential to CT, provides an excellent tool for explorations. Consider starting with a square. Join the mid-points. A new shape comes up, repeat the process. This simple recipe leads to beautiful figures. When the student realises that the procedure is abstract and can be applied to any polygonal figure as "input," she begins to have a taste for CT. We can then begin to see tessellations, *kolams* (or rangoli), and fractals as opportunities for creation of patterns by iterative procedure, reasoning about their variations and communicating such understanding in formal terms. In senior school, we can then describe the change of physical, biological and economic systems over time modelled by simple equations applied repetitively, and use these models to predict the long-term behaviour of such systems.

VI. Mathematics education and CT

One natural question that arises is whether computational thinking is actually different from mathematical thinking. This is really a question for foundational thought to be addressed by philosophers, One can narrow it down to the context of school education and assert that it helps pedagogically to meaningfully distinguish the two.

Before we explain this in detail, it is useful to consider university mathematics for a moment. Real analysis abounds with examples that distinguish mathematical thinking and CT. Bolzano's theorem asserts the existence of a root in an interval when a continuous function has values of opposite sign in that interval. The Newton - Raphson method of successive approximations provides a computational method for finding a root. The Brouwer fixed point theorem asserts that for any continuous function f mapping a compact convex set to itself there is a point x such that $f(x) = x$. Computing such a fixed point is a challenge and a general algorithm had to wait until recently to be formulated. Extracting the algorithmic (or constructive) content of mathematical statements and proofs is a greatly interesting challenge.

At primary school level, we consider that it is not especially useful to distinguish mathematics from CT, but it is relevant to highlight opportunities for CT within the mathematics classroom, as we did above. At secondary level, it does become useful to distinguish the two. For instance, consider solving a system of n linear equations (with integer coefficients) in n unknowns. We can learn an algorithm to do this, namely Gaussian

elimination. Nonetheless it is required to develop some intuition into when the system given does not have a solution, or has more than one solution. Indeed, in the latter case, one can ask whether the system must have infinitely many solutions. A further question relates to rational numbers arising at intermediate steps. Should we retain them as rationals and employ rational arithmetic as we proceed further, or convert them to their decimal representations? Does this matter? What do we gain by using a matrix representation for the system of equations? Raising and answering such questions is essential for CT.

CT has relevance not only for mathematics education but also for science and other subjects in school. Giving prominence to data, understanding data qualitatively and quantitatively, and interpreting data are essential skills not only for science but also for geography, and though less appreciated this way, for history as well. The fine arts provide many creative opportunities for CT, and conversely CT can greatly enhance educational contexts in the graphic arts as well as in music and dance.

VII. Revisiting the questions

While we have discussed what CT means and how it can enrich school education, we have not answered the question of *why* we should do so. One thing is clear. Advocacy of online education and the use of digital technology in classrooms arise from premises very different from what we have been discussing here. Our reasons are different.

- Developing *critical thinking* is a central aim of education, and a critical outlook on algorithms is the need of the twenty-first century. Algorithms increasingly run our lives and developing a mature understanding of how data is created and processed by algorithms requires a foundational knowledge of how algorithms work. Mastery over these processes is best developed slowly, over the school years.
- Developing *autonomy* in the learner is again a central aim of education, and computing provides a powerful new addition to the learner's toolkit for understanding the world. Not only is this new tool versatile, it adds capabilities as yet unexplored in school.
- *Resource consciousness* is a crucial need for modern life, and while this is an ecological imperative, instilling such consciousness needs to be attempted in ecologically sensitive practices; formal thinking on such lines is to be nurtured as well. Computing science provides a new such opportunity by bringing in a sensitivity to scaling and complexity of resource utilisation.
- Education embodies the spirit of *modern democracy* in preparing the citizen for participation in social development and directing the path of development. In the contemporary world, this is impossible without the citizen gaining *democratic control over data*, all data that involves her, and all data that is a determinant of her welfare. Educating the citizen on the relationship between data and democracy is thus a curricular imperative.

Viewed thus, the aims of CT education at school are about utilising the tremendous new potential brought by computation for autonomy and empowerment, and at the same time developing a critical outlook on data and algorithms, and a sensitivity to resource use as practices scale up.

These are broad statements of aims. What should be learnt at which school age is best decided on the basis of research on psychology of children's learning, not by availability of technological tools. Indeed, digital technology can be seductive in its glamorous manifestation and we need to be wary of children becoming enslaved by devices. Such considerations again suggest that the relationship between CT and mathematics education as we have discussed provides more safe and meaningful opportunities for CT than a technology-based understanding of CT.

Lastly, whatever the NEP may advocate, and however it gets implemented, we need to ask whether we have the capability for introduction

of CT in the education system at all levels. The teaching community, especially in mathematics, is alive to the possibilities and needs support by way of pedagogic resources. The experience of **CSpathshala**, a voluntary initiative of *ACM India*, providing a complete CT curriculum ([2]) for schools and reaching nearly a thousand schools in the country, offers a strong foundation from which many future initiatives can take off. We note here that the insights into CT presented in this article largely stem from the *cspathshala* experience.

The 2019 mathematics textbooks of Tamil Nadu State Board include an *information processing* track incorporating elements of a CT curriculum. The largely positive response from teachers to this initiative again offers hope.

References

[1] Bell, T., Alexander, J., Freeman, I., and Grimley, M. *Computer science unplugged: School students doing real computing without computers*. The New Zealand Journal of Applied Computing and Information Technology, 13(1), 20–29, 2009.

[2] The CSPathshala curriculum, https://cspathshala.org/curriculum/

[3] Denning, Peter J. "The science in computer science." *Communications of the ACM*, 56(5), 35–38, 2013.

[4] Papert, Seymour. *Mindstorms: Children, computers and powerful ideas*. New York, NY: Basic Books, 1980.

[5] Papert, Seymour. "An Exploration in the space of mathematics educations," *International Journal of Computers for Mathematical Learning*, Vol. 1, No. 1, pp. 95-123.

[6] Wing, Jeannette, "Computational thinking." *Communications of the ACM*, 49(3), 33–35, March 2006.

[7] Wing, Jeannette, "Computational thinking and thinking about computing." *Philosophical Transactions of the Royal Society*, 366 (1881), 3717–3725, 2008.

R RAMANUJAM is a researcher in mathematical logic and theoretical computer science at the Institute of Mathematical Sciences, Chennai. He has an active interest in science and mathematics popularization and education, through his association with the Tamil Nadu Science Forum. He was awarded the Indira Gandhi Prize for Popularisation of Science for the year 2020. He may be contacted at jam@imsc.res.in.