Back to basics in algorithms

The NEP 2020 must take a larger view of computational thinking and move beyond just data science and AI

S. Arun-Kumar Subhashis Banerjee*

Computer Science and Engineering
IIT Delhi

October 14, 2020

It is heartening to note that the National Education Policy 2020 (NEP) envisages putting greater emphasis on mathematical and computational thinking throughout the school years, starting right from the foundational stage of the learning process. Indeed, algorithmics – the abstract process of arriving at a post-condition through a sequential process of state changes – is among the earliest human intellectual endeavours that has become imperative for almost all organised thinking. However, the framing in the NEP appears to put it at the same level of distinction as the more instrumental 'coding', and almost as a mere tool towards the utilitarian goals of artificial intelligence (AI) and data science. We think this is misplaced.

The notions of computation and algorithms are as old as mathematics and date back to the early stages of representing numbers and geometrical figures and manipulating them for various uses. Their origins can be traced back almost to the beginning of the Mesolithic stone age, when the notions of counting and addition began to take form. All early learning of counting and arithmetic is method-based and hence algorithmic in nature, and all calculations involve computational processes encoded in algorithms.

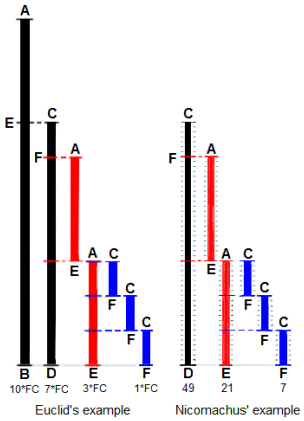

Figure 1: Euclid's method for finding the greatest common divisor (GCD) of two starting lengths BA and DC, both defined to be multiples of a common "unit" length. On the right is Nicomachus's example with the numbers 49 and 21 resulting in their GCD of 7. Source: Wikipedia.

*also associated with the School of Public Policy

Modern algorithms, almost in the same form as we know them today, began to appear from around 300 BC. Some of the earliest examples are the procedures to compute the greatest common divisor of a pair of numbers, or the factorial, which appeared in Euclid's books of *Elements*; and the descriptions of fast exponentiation and the Fibonacci sequence which appeared around 200 BC in the treatise of *Chandah-Sutra* by Acharya Pingala. The core algorithmic ideas of modern AI and machine learning are based on some seminal algorithmic ideas of Newton and Gauss, which date back a few hundred years. Though the form of expressions of algorithms – 'coding' – have been different across ages and cultures, the fundamental principles of classical algorithm design have remained invariant.

In the modern world, the use of algorithmic ideas is not limited only to computations with numbers, or even to digitization or communication or AI and data science. They play a crucial role in modelling and expressing ideas in diverse areas of human thinking including the basic sciences of biology, physics and chemistry; all branches of engineering; in understanding disease spread; in modelling social interactions and social graphs; in transportation networks, supply chains, commerce, banking and other business processes, and even in economic and political strategies and in design of social processes. Hence learning algorithmic thinking early in the education process is indeed crucial.

Coding however, is merely the act of encoding an algorithmic method in a particular programming language, which provides an interface such that the computational process can be invoked in a modern digital computer. It thus is less fundamental, and indeed great algorithms have been designed through the ages even without this facility. While coding certainly can provide excellent opportunities to the initiated for experimentation with algorithmic ideas, they are not central or indispensable to algorithmic thinking. After all, coding is merely one vehicle to achieve experiential learning of a computational process.

Rather than the intricacies of specific programming languages, it is more important at an early stage of education to develop an understanding of the basic algorithmic processes behind manipulating geometric figures, computing with numbers, solving systems of equations, modelling road networks and social graphs, and applying algorithmic ideas to everyday problems. In fact, an overemphasis on learning the nitty-gritty of specific programming languages prematurely - even from middle school - may actually distract from focussing on the development of algorithmic creativity. Indeed, this is a common outcome of the overly utilitarian 'skills training' based approaches evidenced throughout the country.

The devil lies in the details, and, while the NEP guideline of introducing algorithmic thinking early is a very welcome step, we advocate care in ensuring that it does not degenerate and get bogged down with mundane coding tricks at a budding stage in the education process.