Paul Stephenson and Ian Benson share highlights from their Computational Thinking in Mathematics session from ATM's virtual conference in October 2024

Number City

How do their prime factors appear when we take the natural numbers in order?

On a square grid, each number has its own column and each prime factor has its own row. Cubes are placed so the prime factorisation can be read down the column. Where the prime factor repeats, the cubes are piled on top of one another, so eight has three cubes in the 2 row and nine has two cubes in the 3 row. 'A teacher's guide to Number City,' explains how to build the model for up to 150. Whether or not you make a physical model, you can explore a GeoGebra applet by Ben Sparks (n.d.). In Figure 1, we show the model for one to ten in Multilink, Centicubes and 1 cm wooden cubes:

We call the representation Number City because it resembles an architect's model of a city laid out on a grid system, bordering a waterfront, which we shall assume runs west to east.

Already in this small corner you can see the diagonal line of primes and the same line multiplied by two and three.

In Number City each prime is located by its row. We read a number's prime factorisation down the column. We know from The Fundamental Theorem of Arithmetic that each column is unique.

One strand of the virtual conference on 5 October 2024, was Computational Thinking in Mathematics. Pedagogically, our jumping-off point was the 'qualitative arithmetic' of Caleb Gattegno. It's instructive to contrast how Gattegno (1974) represents the positive integers and how the actions of making figure 1 can be represented as an algorithm in the functional programming language Haskell.

In Gattegno's 'Mathematics with Numbers in Colour' a product of two numbers is modelled by setting the relevant rods at right angles in the form of a `cross' in the vertical plane. Gattegno extends the notation to show a power n of prime a by setting a stack of n a rods at right angles to one another to form a 'tower' (see Figure 2). A number represented by its prime factors is then shown by stacking towers, of which each number has a unique combination. But it is not Gattegno's main purpose to demon-

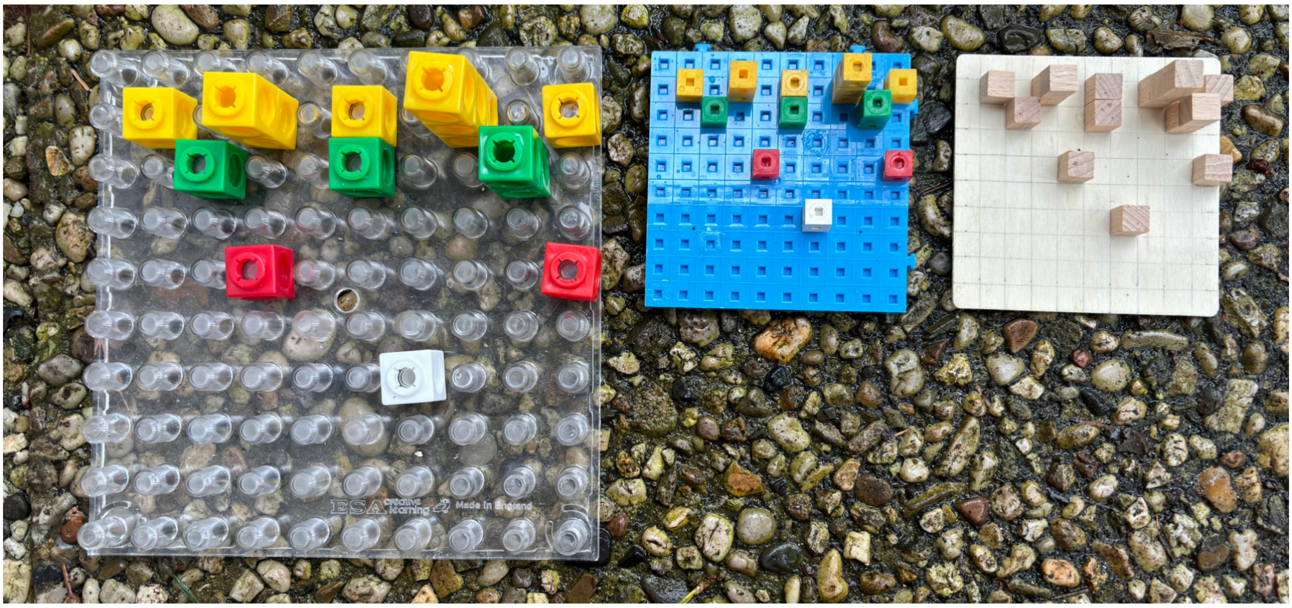

Figure 1. Number City representation for numbers 1 to 10.

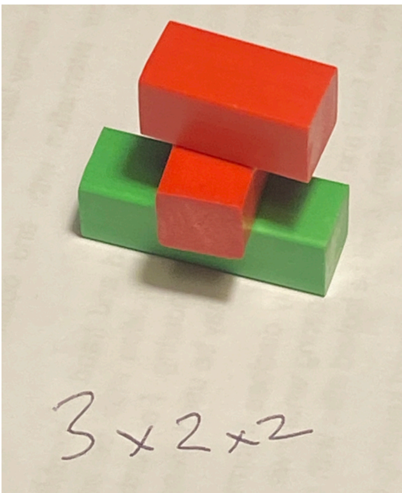

Figure 2. Representing 12 with a tower of Cuisenaire rods.

strate The Fundamental Theorem of Arithmetic. His aim is rather to show the many different ways a composite number can be factorised.

In the visualisation in Figure 1 and in the applet the primes are found in the ray extending at 45° from the horizontal. The process of removing the composite numbers, first by eliminating two and its multiples (the yellow cubes), then three and its multiples (the green cubes) and so on is known as the Sieve of Eratosthenes. The algorithm, written in Haskell, is:

```
-- Prime numbers generator using the
Sieve of Eratosthenes
primes :: [Int]
primes = sieve [2..]
  where
    sieve (p:xs) = p : sieve [x | x <-
xs, x `mod` p /= 0]
-- Display the first 50 prime numbers
take 50 primes
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,
47,53,59,61,67,71,73,79,83,89,97,101,
103,107,109,113,127,131,137,139,149,
151,157,163,167,173,179,181,191,193,
197,199,211,223,227,229]
```

The Haskell code reads that `primes` is an infinite list of integers, implemented by applying the function `sieve` to the infinite list of $n > 1$ where each new number in the sequence is reached by appending the first number in the input list to the result of applying the `sieve` function to the list of numbers that do not contain any of its multiples. That is, by eliminating each of the composite numbers whose multiples are found in the prior rows in Figure 1.

Haskell's lazy evaluation mechanism is employed in the illustration where 50 primes are taken from the front of the list. `primes` is an infinite mathematical object, but it is only executed until the 50th prime is found. The ability to represent infinite objects is a feature of lazy functional programming. It is not found in imperative languages, such as Scratch or Python, encountered in school computing.

The task has been designed to give learners the opportunity to become sensitive to the multiplicative relationships between numbers. Teachers who already have this awareness are introduced to a higher level of abstraction: recursive functions. This is an example of task design suggested by John Mason. He proposed that the critical feature for promoting algebraic thinking is not the tasks given to learners, but rather the opportunities noticed by teachers for calling upon learners' powers to express and manipulate generalities. He says that "this is enriched when teachers engage in similar tasks at their own level, so as to sensitize themselves to pedagogic opportunities when working with learners."

References

Gattegno, C. (1974). *Common Sense of Teaching Mathematics*, Educational Solutions Worldwide

Mason, J. (2018), How Early Is Too Early for Thinking Algebraically? In C. Kieran (ed.), *Teaching and Learning Algebraic Thinking with 5- to 12-Year-Olds, ICME-13*, *https://piazza.com/class_profile/get_resource/io2sufmixtxd4/m0gy3xoivmi6no*

Sparks, Ben (n.d.). *Number City GeoGebra Applet*, *www.geogebra.org/classic/jga6adzg*

Stephenson, P. (2024). *A Teacher's Guide to Number City*, *atm.org.uk/write/MediaUploads/Journals/PaStNumber_City_talk.pdf*

The Functional Programming and Computer Algebra Working Group is leading an ATM strategic initiative in computational thinking. We have developed a free self-study course, the Haskell Road to Mathematics, with Glasgow University School of Computing. For more information email ian.benson@cs.stanford.edu.