The Sieve of Eratosthenes

(*Eratosthenes of Cyrene ca 276-194 BC*)

Jochen Ziegenbalg

Each natural number (i.e. each positive integer) n has the "trivial" divisors 1 and n itself. Therefore, each natural number different from 1 has at least two divisors. Natural numbers having exactly these two divisors are called *prime numbers*. According to this definition, the number 1 is not considered a prime number. There are good reasons for this; one of them is that otherwise the *fundamental theorem of number theory* on the prime factor decomposition of the integers would not be true.

Prime numbers constitute one of the oldest and most interesting fields of research in mathematics. They are the building blocks from which all natural numbers are constructed by multiplication.

The **Fundamental Theorem of Number Theory** says that

> Each natural number n ($n > 1$) can be written as a product of prime numbers:
> $n = p_1 \cdot p_2 \cdot \ldots \cdot p_s$. Except for the order of the factors, this representation is unique.

For other number systems or algebraic systems, prime numbers, prime elements or accordingly modelled concepts are of fundamental importance. A fascinating property of prime numbers is the irregularity with which they occur. Finding patterns in the sequence of the prime numbers always was an important field of mathematical research.

Already in antiquity mathematicians tried to develop an understanding of prime numbers. *Euclid of Alexandria* (ca 325 BC - 265 BC) showed that infinitely many primes exist. The Greek mathematician *Eratosthenes of Cyrene* devised a procedure to determine all prime numbers up to a given number n. This "sieve" procedure is explained in the following example for $n = 20$.

1. List all natural numbers from 1 to 20:

   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2. Cross out the number 1 (it is not considered a prime number; see the discussion above):

   ~~1~~ 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

3. Underscore the number 2:

   ~~1~~ 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

4. Cross out all proper multiples of 2; i.e. the numbers 4, 6, 8, 10, 12, 14, 16, 18 and 20:

   ~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~

5. In the remaining numbers, underscore the first "free" number (i.e. the first number which is neither underscored nor crossed out); in this case underscore the number 3:

   ~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~

6. Cross out all the proper multiples of 3 (the new numbers to be crossed out in this case are the numbers 9 and 15):

   ~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~

7. Underscore the smallest free number; in this case underscore the number 5:

   ~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~

8. Cross out all the proper multiples of 5. Since all of the candidates for "crossing out" (i.e. the multiples 10, 15 and 20 of 5) are crossed out already, there is no change in this case (with maximum $n = 20$).
9. Continue with the procedure accordingly until each of the numbers is either underscored or crossed out.

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~

10. End of the procedure. The underscored numbers are the prime numbers between 1 and 20.

By this procedure, the prime numbers are "sieved out", so to speak. Therefore, the procedure is called the *Sieve of Eratosthenes* or the *Sieve*, for short.

Exercises:

(a) Run the *Sieve of Eratosthenes* manually for the upper limit $n = 200$.
(b) Give a general description of the Sieve procedure which is independent of the number 20 (the upper limit, in the general case, being n).
(c) Show that if the number n is composite, e.g. $n = x \cdot y$ (with factors x and y greater than 1), then one of the factors is less than or equal to $\sqrt{n}$.
(d) Show that if a number is crossed out after running the full Sieve procedure, it is already crossed out after $\sqrt{n}$ is reached.
(e) As a consequence of part (d), the Sieve procedure can be *considerably* abbreviated. Describe the Sieve algorithm in such a way that by taking this fact into account, the efficiency of the algorithm is considerably improved.

Remarks:

(1.) Occasionally, one can read that the goal of the Sieve of Eratosthenes is to find *the* prime numbers (i.e. *all* prime numbers). A glance at the algorithm, however, shows that it can only work if it is confined to a limited range of natural numbers (in the example: the numbers from 1 to 20). The Sieve of Eratosthenes, thus, produces primes only up to a given upper limit. This limit, however, can be pushed upward by allowing for more "runs" of the algorithm. The infinite set of the prime numbers, in the philosophical sense, is thus generated as a "potentially" infinite set (in the Aristotelian sense). This is a good place to quote Euclid's extremely farsighted statement: "*Prime numbers are more than any assigned multitude of prime numbers*" (see for instance: J. Ziegenbalg: Elementare Zahlentheorie - Beispiele, Geschichte, Algorithmen; Springer Spektrum, Wiesbaden, p. 51).

(2.) The above described first version of the Sieve algorithm is very slow. Due to this slowness it was used for some time to measure the speed of various computer systems (for fast algorithms may run so fast that a time measurement is practically impossible). For years, the renowned computer magazine BYTE used a procedure based on the Sieve of Eratosthenes for so-called benchmark tests, i.e. for testing the speed of hard- and software. Thus, the Sieve of Eratosthenes can be called "classic" in a twofold sense: First, in the original sense of Eratosthenes for generating primes and second, as a benchmark test. The program used by BYTE can easily be implemented in various programming languages. Since BYTE's version was slightly incorrect as far as the final printout is concerned, it was useful for nothing else but for measuring the run-time on various computer systems.

An interactive simulation can be found under the following address

https://jochen-ziegenbalg.github.io/root/Simulationen/Sieve-of-Eratosthenes/Sieve-of-Eratosthenes-Simulation.html

Programs for the Sieve of Eratosthenes

In the following two programs the "inefficient" version of the Sieve of Eratosthenes from the BYTE journal is implemented in the computeralgebra systems *Mathematica* and *Maxima*.

The activity of "listing all natural numbers …" is implemented by making a list of them (in the sense of the data type "list" in the programming language LISP and various computeralgebra systems). "Crossing out" a number is done by setting this entry to zero.

Implementation in the computeralgebra system *Mathematica*:

```
SieveOfEratosthenes[UpperLimit_] :=
  Module[{L = Table[t, {t, 1, UpperLimit}], i = 2, k},
    L = ReplacePart[L, 0, 1];
    While[i*i <= UpperLimit,
      k=i+i;
      While[k <= UpperLimit,
        L = ReplacePart[L, 0, k];
        k = k+i];
      i = i+1];
    Return[Select[L, Positive ] ] ]
```

Comment from the Help-Browser of Mathematica:
`ReplacePart[`expr, new, n`]` yields an expression in which the n-part of expr is replaced by new.

A concrete call of the above program (with the parameter UpperLimit = 80) yields:

```
SieveOfEratosthenes[80]
{2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79}
```

Implementation in the computeralgebra system *Maxima*:

```
Eratosthenes(UpperLimit) :=
 block([E, i, k],
   E : makelist(j, j, 1, UpperLimit),
   E[1] : 0,
   i : 2,
   while i*i <= UpperLimit do
    (k : i+i,
     while k <= UpperLimit do
       (E[k] : 0,
        k : k+i),
     i : i+1),
   E : delete(0, E),
   E);
```

A concrete call:

```
Eratosthenes(50);
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
```

References

Ziegenbalg J. / Ziegenbalg O. / Ziegenbalg B.: Algorithmen – von Hammurapi bis Gödel; Springer Spektrum, Wiesbaden 2016, Section 3.2.3

Ziegenbalg J.: Elementare Zahlentheorie - Beispiele, Geschichte, Algorithmen; Springer Spektrum, Wiesbaden 2015, Chapter 4

http://jochen-ziegenbalg.github.io/root/