A method to investigate primality.

(Mathematical tables and other aids to computation, 11(1957), p 195-196).

E.W. Dijkstra.)
A Method to Investigate Primality

The method determines the smallest odd prime factor of a number \(N\) by testing the remainders left after division by the successive odd numbers 3, 5, \(\ldots\) \(f_m - 2, f_m\): here, \(f_m\) is the largest odd number not exceeding \(N\). If none of these remainders vanishes, \(N\) is a prime number.

Let \(f\) be one of the odd trial divisors. Remainder \(r_0\) and quotient \(q_0\) are defined by the relations

\[
N = r_0 + f q_0, \quad 0 \leq r_0 < f.
\]

Now \(q_0\) is divided by \(f + 2\), giving

\[
q_0 = r_1 + (f + 2) q_1, \quad 0 \leq r_1 < f + 2.
\]

Then \(q_1\) is divided by \((f + 4)\), etc., and this process is continued till a quotient \((q_n, \text{say})\) equal to zero is found; \(r_n\) is the last remainder in the sequence unequal to zero. After elimination of the \(q_i\) we get the relations

\[
(1) \quad N = r_0 + f r_1 + f (f + 2) r_2 + f (f + 2) (f + 4) r_3 + \cdots
\]

\[
+ f (f + 2) \cdots (f + 2n - 2) r_n
\]

and

\[
(2) \quad 0 \leq r_i < f + 2i.
\]

Once the sequence \(r_i\) is known for a given value of \(f\), it is easy to compute the corresponding sequence \(r_i^*\), defined by the relations (1) and (2) with respect to \(f^* = f + 2\), as they are expressed in terms of the \(r_i\) by the recurrence relations

\[
(3) \quad b_0 = 0, \quad r_i^* = r_i - 2(i + 1) r_{i+1} - b_i + (f^* + 2i) b_{i+1}, \quad (i = 0, 1, \cdots, n).
\]

The relation corresponding to (1) is satisfied for arbitrary values of the numbers \(b_i\) with \(i \geq 1\); they are fixed, however, by the relations corresponding to (2)

\[
(2^*) \quad 0 \leq r_i^* < f^* + 2i.
\]

On account of the inequalities (2) and (2*)—and \(b_0 = 0\)—the \(b_i\) satisfy the inequalities

\[
(4) \quad 0 \leq b_i \leq 2i.
\]

We have chosen \(b_0 = 0\). Then the relations (3) and (2*) with \(i = 0\) determine \(r_0^*\) and \(b_1\); once \(b_1\) is known, (3) and (2*) with \(i = 1\) determine \(r_1^*\) and \(b_2\), etc. The process is easily programmed.

As \(r_{n+1} = 0\), and the inequalities (2*) with \(i = n\) are always satisfied with \(b_{n+1} = 0\), the process terminates with

\[
\begin{align*}
    r_n^* & = r_n - b_n. \\
\end{align*}
\]
As soon as \( r^*_n = 0 \) is found—in that case it can be proved that \( r^*_{n-1} \neq 0 \)—the index \( n \), marking the last \( r_i \neq 0 \) in the sequence, is lowered by 1.

In order to find the smallest odd prime factor of \( N \), the \( r_i \) defined by (2) and (3) and \( f = 3 \) are computed. Here the only divisions in the process are carried out. At the same time the initial value of \( n \) is found. If \( N \) is large, this value may be considerable: for instance \( n = 11 \) is found for \( N = 10^{10} \). The amount of work involved in each step is roughly proportional to \( n^2 \). Fortunately large initial values of \( n \) decrease very rapidly. As soon as \( f \cdot (f + 2) \cdot (f + 4) > N \), \( n \) takes the value 2. This is its minimum value: when \( r^*_n = 0 \) with \( n = 2 \) is found, \( (f^* + 2)^2 > N \) holds and \( N \) is a prime number. (If not, we should have found an \( r_0 = 0 \) earlier and should have stopped there.)

The process still may be speeded up. Let \( b_n' \) be the minimum of \( b_n \) for fixed \( n \) up till a certain moment: then it can be shown that the next \( b_n \) satisfies

\[
b_n \leq b_n' + 1.
\]

Let us apply this to the last stage \( n = 2 \). According to (4) \( b_2 \) satisfies \( 0 \leq b_2 \leq 4 \). According to (5), however, the only possible values for \( b_2 \) are 0 and 1 as soon as a value \( b_2 = 0 \) once has been found. This is bound to happen for \( f \) ranging (roughly) from \( (4N)^{1/4} \) to \( (8N)^{1/4} \). In the case \( b_2 = 0 \) it is apparently unnecessary to test whether \( r_2 = 0 \) is reached. (If \( N \geq 144 \), the case \( b_2 = 0 \) with \( n = 2 \) occurs, before \( r^*_n = 0 \) with \( n = 2 \) is found; prime numbers are then always detected in this last stage.)

The less efficient steps of the process for large \( n \) (i.e., small \( f \)) could be avoided by carrying out divisions for small values of \( f \) (see Alway [1]). However we strongly advise against doing this.

If the process described above is started at \( f = 3 \), the whole computation can be checked at the end by inserting the final values of \( f \) and \( r_i \) into (1). As all the intermediate results are used in the computation, this check seems satisfactory.

If a double-length number \( N \) is to be investigated, another argument can be added: division of \( N \) by small \( f \) may give a double-length quotient, i.e., two divisions (and two multiplications to check) are needed for each \( f \). In our case only part of the initial \( n \) divisions are double-length divisions.

The process described above has been programmed for the ARMAC (Automatische Rekenmachine van het Mathematisch Centrum). The speed of this machine is about 2400 operations per second. A twelve decimal number was identified as the square of a prime in less than 23 minutes.

E. W. Dijkstra

Mathematical Centre
Computation Department
Amsterdam, The Netherlands