Proc. Nat. Acad. Sci. USA
Vol. 68, No. 10, pp. 2319–2320, October 1971

The 24th Mersenne Prime

(computer/Lucas–Lehmer test/perfect number/computational speed)

BRYANT TUCKERMAN

IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598

Communicated by E. R. Piore, July 22, 1971

ABSTRACT The 24th Mersenne prime $M_p = 2^p - 1$, and currently the largest known prime, is $2^{19937} - 1$. Primality was shown by the Lucas-Lehmer test on an IBM 360/91 computer. The 24th even perfect number is $(2^{19937} - 1) \cdot 2^{19936}$.

A number (integer) n is called *perfect* if (like 6 or 28) it equals the sum of its divisors less than n. Let p always denote a prime. Euclid and Euler showed that an even n is perfect if, and only if, $n = (2^p - 1) \cdot 2^{p-1}$ for some p for which $2^p - 1$ is itself a prime. Thus, the determination of even perfect numbers is reduced to the determination of which *Mersenne numbers* $M_p = 2^p - 1$ are primes, i.e., *Mersenne primes.*

Mersenne (1644) gave, in effect, a list of p which he asserted (on unknown grounds) were those $p \leq 257$ that yield prime M_p. It was not until 1947 that the examination of this range was completed by others, after laborious hand and desk calculations; during these calculations, several errors were found in Mersenne's list.

The 23 Mersenne primes already determined are for the exponents $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 13$, $p_6 = 17$, $p_7 = 19$, $p_8 = 31$, $p_9 = 61$, $p_{10} = 89$, $p_{11} = 107$, $p_{12} = 127$, $p_{13} = 521$, $p_{14} = 607$, $p_{15} = 1279$, $p_{16} = 2203$, $p_{17} = 2281$, $p_{18} = 3217$, $p_{19} = 4253$, $p_{20} = 4423$, $p_{21} = 9689$, $p_{22} = 9941$, $p_{23} = 11213$. Those from p_{13} on have been determined by electronic computers, starting in 1952. The last three were found by Gillies (1).

As these Mersenne primes became successively known, their irregularity and increasing sparseness aroused interest and speculation. Somewhat surprising, and perhaps discouraging at times, were the large gaps, as between p_{12} and p_{13}, and between p_{20} and p_{21} (see ref. 1 for conjectures on their distribution).

The standard algorithm for testing the primality of an M_p is the Lucas-Lehmer test (2), described as follows: Let $u_1 = 4$ (an infinity of other values, such as $u_1 = 10$, are also acceptable). Let $u_{i+1} = u_i^2 - 2$, for $i = 1, 2, \ldots, p - 2$. Then M_p is prime if, and only if, $u_{p-1} \equiv 0 \pmod{M_p}$. The recursion equation can be replaced by the congruence

$$(*) \quad u_{i+1} \equiv u_i^2 - 2 \pmod{M_p}$$

where each u_i can be the least nonnegative residue mod M_p, and is thus a number of (at most) p binary digits, which can be represented by $l = \lceil p/w \rceil$ "digits" to a suitable base $\beta = 2^w$.

This primality-test is very cheap to perform on a computer compared to testing an arbitrary number of comparable size. Thus, the largest known Mersenne prime at any time is usually also the largest known prime. Nevertheless, the computational labor of this test increases rapidly with p, which fact, combined with the expected increasing rarity of Mersenne primes, makes the search for the next such number a substantial, but not hopeless, computational task.

An implementation of the above test was programmed for the IBM 360/91 computer after timing estimates indicated its reasonableness in this search. This test has been used for all $p < 20{,}000$, after confirming and excluding those M_p that have known factors (1, 3, 4).

The programs use single-length floating-point ($w = 24$) rather than fixed-point ($w = 31$) arithmetic, because the IBM 360 instruction repertoire and the design emphasis of the model 91 favor floating-point speed more than enough to counterbalance the larger mathematical operation count due to the smaller base.

The required squaring in (*) of a multiple-precision integer $u = \sum_{i=0}^{l-1} x_i \beta^i$ is accomplished by the formula

$$(**) \quad u^2 = \sum_{k=0}^{l-1} x_k^2 \beta^{2k} + \sum_{k=1}^{2l-3} \beta^k \cdot 2 \sum x_i x_j$$

(where the right-most summation runs over all (i, j) such that $i + j = k$ and $0 \leq i < j \leq l - 1$), with appropriate provisions for carries, unpacking partial results, etc. The heart of this calculation is the accumulation of double-length products in an inner looplet

$$(***) \quad s_k \leftarrow s_k + x_i \cdot x_j$$

executed a total of $l(l - 1)/2$ times, for various k, i, j. [Thus, the time for one squaring (**) is $O(p^2)$.] It was crucial to the practicality of this implementation to make this looplet (***) sufficiently fast. This proved to be possible due to two factors: (*a*) the inherent speed, and special facilities (such as pipelining, operand buffering, and parallelism) of the computer, and (*b*) special programming to take extra advantage of those facilities. The net effect is to perform one looplet (***) per 4 cycles of 60 nsec each, with some external overhead.

The other parts of (*), including the reduction mod M_p, and the checks to be described, contribute $O(p)$ to the time for a recursion (*), so that less care was originally expended in optimizing their speed.

As a control both on the programming and on the computer operation, each recursion was checked by residues, both mod $2^{24} - 1$ and mod $2^{24} - 3$. The first is very convenient, but it would not detect, for example, the addition of a correct quantity into a wrong digit position. The second check avoids this weakness. It requires more computation than the first check, but the penalty is tolerable. These modular checks never reported a computer error, and did detect synthetic errors. Consequently, I felt it was sufficient to run most of the cases of p only once.

This program, which we will denote by (a), was run first on the lower part of the known range of p, for confirmation, and then into the unknown range. Production was done chiefly during otherwise idle times at the end of the third shift. On the evening of March 4, 1971, a zero Lucas-Lehmer residue for $p = p_{24} = 19937$ was found. Hence, M_{19937} is the 24th Mersenne prime.

After the production was well under way, timing measurements showed that the times mentioned above as $O(p)$ per (*) were not negligible, so some of these parts of the program were rewritten. This faster program (b) has been used for checking the upper part of the known range of p, for confirming program (a) in a few cases, and is currently being used to search $p > 20{,}000$. The computation times for (a) and (b) for a few noteworthy values of p were: for M_{8191} (not a prime, but of historical interest), 3.78 and 3.10 min; for M_{11213}, 8.36 and 7.06 min; for M_{19937}, 39.44 and 35.01 min. These test times are about five times as fast as with equivalent nonoptimized programs that use the same principle to test M_p.

The values of the 24th Mersenne prime, $2^{19937} - 1$, and of the corresponding 24th even perfect number, $(2^{19937} - 1) \cdot 2^{19936}$, have been computed in decimal. The prime (the largest known) has 6,002 digits, and begins and ends 4315424797... 0968041471. The perfect number (the largest known) has 12,003 digits, and begins and ends 9311445590...0271942656.

The last five octal digits of the non-zero residues for the composite M_p, and some additional details of the computations, will be published elsewhere.

After my discovery of $p_{24} = 19937$ became known, I heard from Michael Speciner and Richard C. Schroeppel of MIT that they had been seeking p_{24} with a Lucas-Lehmer test by a multiplication whose computation time is $O(p^{1.57})$ (see ref 5, p. 258–9). The potential advantage in speed of that method did not suffice to overcome the more rapid calculating ability of the model 91 and its program over the range of numbers tested. The workers at MIT have kindly reported confirming the primality of M_p for my value of p_{24} with their program.

1. Gillies, D. B., *Math. Computation*, **18**, 93 (1964).
2. Lehmer, D. H., *Annals Math.*, **31**, 419 (1930).
3. Kravitz, S., *Math. Computation*, **15**, 292 (1961).
4. Brillhart, J., *Math. Computation*, **18**, 87 (1964).
5. Knuth, D. E., *The Art of Computer Programming*, Vol. 2. (Addison-Wesley Press, Reading, Mass., 1969).