MATHEMATICS OF COMPUTATION
VOLUME 55, NUMBER 192
OCTOBER 1990, PAGES 815–820

EXTENDING WARING'S CONJECTURE TO 471,600,000

JEFFREY M. KUBINA AND MARVIN C. WUNDERLICH

ABSTRACT. Two computer techniques are described that the authors used to confirm Waring's Conjecture to 471,600,000, thus extending R. M. Stemmler's result of 200,000 computed in 1964. M. C. Wunderlich increased the result to 175,600,000 in August 1988, and in the following October, J. M. Kubina, using a method suggested by Jean-Marc Deshouillers, raised it to 471,600,000.

In 1770 Edward Waring asserted in his *Meditationes Algebraicae* that every positive integer can be written as the sum of four squares, as the sum of nine nonnegative cubes, and as the sum of nineteen fourth powers. Later that year Lagrange proved the case for the sums of four squares. The function $g(k)$ is usually defined as the smallest integer n such that every positive integer can be written as the sum of n kth powers of nonnegative integers. In 1772 Euler obtained a lower bound for $g(k)$ by noting that if

$$3^k = q2^k + r, \quad \text{with } 0 \le r < 2^k, \tag{1}$$

then the number

$$q2^k - 1 = (q-1)\cdot 2^k + (2^k - 1)\cdot 1^k$$

requires $q - 1 + 2^k - 1 = q + 2^k - 2$ kth powers. Thus, with q defined in equation (1),

$$g(k) \ge q + 2^k - 2.$$

In 1909 Hilbert proved that $g(k)$ is finite for all k. In 1935 Dickson and Pillai independently proved virtually the same result, that for $g^*(k) = q + 2^k - 2$ and $k \ge 7$, $g(k) = g^*(k)$, whenever

$$q + r \le 2^k, \tag{2}$$

for q and r defined in equation (1). Here, for clarity, the authors will bend the interpretation of history and refer to the statement $g(k) = g^*(k)$ as Waring's conjecture. In 1909 Wieferich proved $g(k) = g^*(k)$ for $k = 3$, in 1964 Chen proved it for $k = 5$, and in 1940 Pillai proved it for $k = 6$. An excellent exposition of Waring's problem is given by Small [4].

Received January 30, 1989; revised April 11, 1989 and June 26, 1989.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 11–04, 11P05.
The authors would like to thank Bruce Nemnich and John Thorp of Thinking Machines whose patient tutelage was extremely valuable to them.

To test condition (2), for $k \geq 2$, let $(b_{L(k)}b_{L(k)-1}\cdots b_1)_2$ be the binary expansion of 3^k, where $L(x) = \lceil \log_2(3^x + 1) \rceil$. The bits $(b_{L(k)}b_{L(k)-1}\cdots b_{k+1})_2$ and $(b_k b_{k-1}\cdots b_1)_2$ represent the binary expansion of q and r, respectively. When $k \geq 2$, it can be shown that $k \geq L(k)-k+1$. The largest carry that could result from the sum of the first k bits of q and r is one. Therefore, since q consists of $L(k)-k$ bits, and r of k bits, $q+r < 2^k$ if $(b_k b_{k-1}\cdots b_{L(k)-k+1})_2$ contains a zero bit.

Inequality (2) was verified for $401 \leq k \leq 200{,}000$ by R. M. Stemmler [5] in 1964 on an IBM 7090. In August 1988 M. C. Wunderlich completed 240 hours of computing on the Connection Machine at the Supercomputing Research Center in Lanham, Maryland, which verified (2) for $7 \leq k \leq 175{,}600{,}000$. Table 1 displays the values of k which produce a new largest string of ones. Note that the longest string yet discovered occurs for $k = 92{,}600{,}006$ and consists of 29 bits, far shorter than the approximately 38,432,000 bits required to violate condition (2).

TABLE 1

k	One bits	k	One bits
5	1	8,093	13
14	3	28,277	14
46	4	49,304	17
58	5	164,000	19
105	6	835,999	21
157	7	2,242,294	22
455	8	10,406,357	23
1,060	9	25,380,333	26
1,256	10	92,600,006	29
2.677	11		

The program to test condition (2) was implemented on the Connection Machine, which is a computer that executes single instructions on multiple data. Its maximum configuration consists of 65,536 bit-manipulative processors, each having an available memory of 64,995 bits. A host computer broadcasts instructions to all processors which are executed in parallel. A bit mask allows instructions to affect a selected set of processors. This Connection Machine was set to run at 7 megahertz, so that each 142 nanoseconds, 65,536 single bit operations are simultaneously executed. Low-level software exists, so that arithmetic operations can be performed on contiguous segments of bits in each of the 64K processors in parallel. In terms of raw power and speed, the Connection Machine performs about eight times as fast as a CRAY 2 if all the 64K processors are counted. The total memory available (about $.43 \times 10^{10}$ bits)

is somewhat smaller than the memory of the CRAY 2 but much larger than most other supercomputers. The very large memory, together with the fact that integer arithmetic can be performed at the hardware level, made this an ideal computer for this application.

For this program, a word size $R = 32$ was chosen for the program, and the number 3^k was stored in the $65536R$ bits that were partitioned over all the processors, with R bits per processor. The program computed 3^{k+1} by multiplying every R-bit word by three, using a parallel shift and add instruction. Then, in parallel, the carry produced in each processor was added into the neighboring processor. This carry propagation was repeated until all the carries were zero. Next, the bits $b_k, b_{k-1}, \ldots$ were examined until a zero bit was found. This procedure continued until 3^k comprised more than $65{,}536R$ bits, at which point R was incremented by 16 and the bits representing 3^k were redistributed throughout the processors. To permit the program to be interrupted, 3^k was stored on disk every two hours. The integrity of the number 3^k was guaranteed by a residue check every 10,000 steps. This was done by computing 3^k modulo the primes 10,007 and 11,003 on the host computer, then computing the value of the number in the Connection Machine modulo the primes, and comparing the results. With these primes a false positive would occur with a probability less than 10^{-8}. A failure did occur on average every 50 hours of computer time. When one occurred, the program restarted from the smaller number stored on disk. A failure never occurred twice for the same number. The program printed each value of k whose associated bit string exceeded 15 ones, which may be useful for anyone who wishes to verify these results.

At a July 1988 conference in Bowdoin College, where one of us (M.C.W.) was present, Deshouillers discussed the following speed-up that he described in [2]. John Selfridge had mentioned a similar method to the second author independently, and this was incorporated in Kubina's later October calculations which we now describe. First notice that the string of ones in the number below, indicated by the bar, is reduced by at most two bits with each multiplication of three:

$$3^0 x_0 = (\ldots 0\overline{111111111}010\ldots)_2$$
$$3^1 x_0 = (\ldots 0\overline{11111110}111\ldots)_2$$
$$3^2 x_0 = (\ldots 0\overline{1111}10010\ldots)_2$$
$$3^3 x_0 = (\ldots 0\overline{11}101011\ldots)_2.$$

This can be easily proven in general. Now suppose 3^k and 3^{k+m} are computed and 3^k satisfies condition (2); an upper bound is placed on m below. If 3^{k+i} failed the test for some $1 \le i \le m$, then its binary expansion would contain a string of ones from k to $L(k+i) - (k+i) + 1$. The results above imply that the expansion of 3^{k+m} would contain a string of ones from bit k to $L(k+i) - (k+i) + 1 + 2(m-i)$. For all possible i, the largest value of the

latter expression is $L(k+1)-(k+1)+1+2(m-1)$, when $i=1$. If m was small enough so that

$$k \geq L(k+1)-(k+1)+1+2(m-1) \tag{3}$$

and the bits k through $L(k+1)-(k+1)+1+2(m-1)$ contain a zero, then each 3^{k+i} would satisfy condition (2). It can be shown that if $m < .20k$, then inequality (3) holds.

To test condition (2) using this technique, J. M. Kubina implemented on the Connection Machine a fast Fourier transform (FFT) integer multiplication program. The Connection Machine available to the authors had no high-speed floating-point hardware. Therefore, a version of the Schönhage-Strassen multiplication algorithm in Aho, Hopcroft, and Ullman [1], which uses integer arithmetic, was programmed, rather than the version in Knuth [3], which uses floating-point arithmetic. A brief description of the program follows. Let $x = (x_{N-1}\cdots x_0)_2$ and $y = (y_{N-1}\cdots y_0)_2$, with their binary representations, be the integers to multiply, where $N = l\cdot 2^c$, for $l, c > 0$ and $x\cdot y < 2^N$. Define $X_i = (x_{(i+1)l-1}\cdots x_{i\cdot l})_2$, $Y_i = (y_{(i+1)l-1}\cdots y_{i\cdot l})_2$, $\mathbf{X} = \langle X_i\rangle_{i=0}^{2^c-1}$, and $\mathbf{Y} = \langle Y_i\rangle_{i=0}^{2^c-1}$. If $W_{r,i} = \sum_{j=0}^{i} X_j \cdot Y_{i-j}$ for $i = 0, 1, \ldots, 2^c-1$ and $\mathbf{W}_r = \langle W_{r,i}\rangle_{i=0}^{2^c-1}$, then

$$w = x\cdot y = \sum_{i=0}^{2^c-1} W_{r,i}\cdot 2^{l\cdot i}. \tag{4}$$

Let $\odot$ denote the componentwise product of vectors over $\mathbf{Z}_{p_s}$, with each p_s a prime, and define $\mathbf{FFT}_{p_s}$ and $\mathbf{IFFT}_{p_s}$ to return the FFT and inverse FFT of a vector over $\mathbf{Z}_{p_s}$. Then the convolution of $\mathbf{X}$ and $\mathbf{Y}$, $\mathbf{W}_r$, is calculated as

$$\mathbf{Z} \leftarrow \mathbf{IFFT}_{p_s}(\mathbf{FFT}_{p_s}(\mathbf{X}) \odot \mathbf{FFT}_{p_s}(\mathbf{Y})),$$
$$\mathbf{W}_s \leftarrow \mathbf{CRT}(\mathbf{Z}, p_s, \mathbf{W}_{s-1}, p_0p_1\cdots p_{s-1})$$

for $s = 0, 1, \ldots, r$, where $p_0p_1\cdots p_r > 2^{c-1}\cdot(2^{l-1}-1)^2$ and **CRT** calculates the elements $W_{s,j}$ of $\mathbf{W}_s$, whose existence is guaranteed by the Chinese Remainder Theorem, such that $W_{s-1,j} = W_{s,j} \bmod p_0p_1\cdots p_{s-1}$ and $Z_j = W_{s,j} \bmod p_s$, for $j = 0, \ldots, 2^c-1$. The binary representation of the product is computed from equation (4) by recalculating the $W_{r,j}$ so that $0 \leq W_{r,j} \leq 2^l-1$. Using 16-bit primes p, the correctness of the product is checked by verifying if $w \bmod p = ((x \bmod p)(y \bmod p)) \bmod p$.

The routine computed the sequence $\langle v_i\rangle_{i=1}^{10}$, where $v_0 = 3^{175,600,000}$ and $v_i = 3^{29,600,000}v_{i-1}$, in one hour and 50 minutes; each multiplication took eight minutes 41 seconds. Both numbers were obtained from Wunderlich's previous calculations. The routine checked each multiplication with nine 16-bit primes,

and the residue check described above (near the end of paragraph five) was done on each v_i with another set of nine 16-bit primes. The probability of an undetected error is less than 10^{-86}. Every check returned positive results in every program run. Also, the last number computed, $3^{471,600,000}$, was checked with all primes of 16 bits or fewer, pushing the probability that it is incorrect to $10^{-28,304}$. The number of consecutive one bits beginning at the kth bit position are given in Table 2, along with the bits k through $k-24$. Since the smallest number of consecutive ones beginning at the k_ith bit position for the test to fail is greater than 13,680,000 and the longest string of consecutive ones beginning at the k_ith position is three, Waring's conjecture is true up to 471,600,000.

TABLE 2

i	k_i	One bits	$b_{k_{i-1}} \cdots b_{k_{i-1}-24}$ for $i \geq 1$
0	175,600,000		
1	205,200,000	3	111000011011000001101110
2	234,800,000	0	010000110000011100011001
3	264,400,000	0	000001110011011001000110
4	294,000,000	1	100100000000110011010100
5	323,600,000	1	100010001100000100001010 0
6	353,200,000	1	101100101001001001011110 1
7	382,800,000	0	010000101001001100001110 1
8	412,400,000	0	010001110010010010100011 1
9	442,000,000	1	101010000000010001001110 1
10	471,600,000	0	010100100100110101111000 1

ACKNOWLEDGMENTS

The authors would like to thank the consultants, the systems staff, and the management of the Supercomputing Research Center for the machine time, their office facilities, and countless hours of technical assistance while conducting this research project.

BIBLIOGRAPHY

1. Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman, *The design of computer algorithms*, Addison-Wesley, Reading, Mass., 1974.
2. Francine Delmer and Jean-Marc Deshouillers, *On the computation of $g(k)$ in Waring's problem*, Math. Comp. **54** (1990), 885–893.
3. Donald E. Knuth, *The art of computer programming*, Vol. 2, Seminumerical Algorithms, 2nd ed., Addison-Wesley, Reading, Mass., 1981.

4. Charles Small, *Waring's problem*, Math. Mag. **1** (1977), 12–16.
5. R. M. Stemmler, *The ideal Waring theorem for exponents* 401–200,000, Math. Comp. **18** (1964), 144–146.

MATHEMATICAL SCIENCES PROGRAM, NATIONAL SECURITY AGENCY, CENTRAL SECURITY SERVICE, FORT GEORGE G. MEADE, MARYLAND 20755-6000