MATHEMATICS OF COMPUTATION
VOLUME 54, NUMBER 190
APRIL 1990, PAGES 885–893

# ON THE COMPUTATION OF $g(k)$ IN WARING'S PROBLEM

FRANCINE DELMER AND JEAN-MARC DESHOUILLERS

ABSTRACT. Over two hundred years ago, Waring raised the question of representing natural integers as sums of integral $k$th powers. At the beginning of this century, Hilbert proved that, for any fixed $k$, the minimal number of summands needed in the representation of any integer can be uniformly bounded. The least such bound is denoted by $g(k)$.

A first goal is to show that our knowledge of $g(k)$ is rather satisfactory:

—Writing down all the numbers $g(2), g(3), \dots, g(K)$ may be performed in $O(K^2)$ bit operations, which is best possible, since $g(k)$ has $(k+1)$ digits in its binary expansion.

—Writing down $g(k)$ may be performed in $O(k \log k . \log\log k)$ bit operations, which we expect to be fairly close to the actual complexity.

A second aim is to discuss the complexity of checking the validity of the conjectured Diophantine inequality

$$\{(3/2)^k\} \leq 1 - (3/4)^k;$$

the underlying idea has led J. M. Kubina and M. C. Wunderlich to check this up to 471,600,000. This inequality is related to Waring's problem in that it would imply the formula

$$g(k) = 2^k + [(3/2)^k] - 2;$$

however, the knowledge of this relation would not improve our knowledge on the complexity of computing $g(k)$, neither on average, nor for individual values of $k$.

## 1. INTRODUCTION

The original statement of E. Waring (1770), according to which every positive integer is a sum of at most 4 squares, 9 cubes, and 19 biquadrates has now been completely proven through the works of J. Lagrange (1770), A. Wieferich and A. J. Kempner (1909–1912), and R. Balasubramanian, J.-M. Deshouillers, and F. Dress (1986). In a subsequent edition of his *Meditationes Algebraicae*, Waring (1782) raised the same question for higher powers. It is a tradition to denote by $g(k)$ the least integer $s$ such that every positive integer can be expressed as a sum of at most $s$ positive $k$th powers; thus, the original statement

Received February 7, 1989; revised May 19, 1989.

1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 11P05, 11Y16; Secondary 11J25.

*Key words and phrases.* Computational number theory, Waring's problem, Diophantine inequalities.

The authors acknowledge the support of the P.R.C. Mathématiques et Informatique.


0025-5718/90 $1.00 + $.25 per page

may be formulated as $g(2) = 4$, $g(3) = 9$, $g(4) = 19$. The finiteness of $g(k)$ is not obvious; the first proof was given by D. Hilbert in 1909; a simpler one is given in [3]. The consideration of the integer $[(3/2)^k] \cdot 2^k - 1$ readily leads to the lower bound

$$g(k) \geq 2^k + [(3/2)^k] - 2,$$

as was noticed by Euler in 1772 (quoted in [2, p. 717]), and it is quite possible that, on the faith of numerical evidence, he had in mind that, for all $k$, equality should hold in the previous relation; we shall refer to the statement

$$(*) \qquad g(k) = 2^k + [(3/2)^k] - 2$$

as "Euler's conjecture" for exponent $k$.

The main step towards this conjecture was accomplished in 1936 by S. S. Pillai and L. E. Dickson, when they independently obtained the precise value of $g(k)$ for ranges containing all $k$ between 7 and 100, proving Euler's conjecture for those exponents. Their results depend on a method—known as the circle-method—introduced in 1917 by G. H. Hardy and S. Ramanujan and developed by G. H. Hardy and J. E. Littlewood around 1920. The combined efforts of L. E. Dickson, S. S. Pillai, R. K. Rubugunday, and I. Niven led to the following solution of Waring's problem, around 1944.

**Theorem A.** *Let* $k \geq 6$, *and define* $X_k$, $Y_k$, $\xi_k$, *and* $\eta_k$ *by*

$$X_k = [(3/2)^k] + 1 = (3/2)^k + \xi_k,$$
$$Y_k = [(4/3)^k] + 1 = (4/3)^k + \eta_k.$$

**Either** *we have*

$$(**) \qquad \xi_k \geq (3/4)^k,$$

*and then Euler's conjecture* $g(k) = 2^k + X_k - 3$ *holds,*
**or** *we have*

**either** $X_k Y_k > 2^k + 1$, *and then* $g(k) = 2^k + X_k + Y_k - 5$
**or** $X_k Y_k > 2^k + 1$, *and then* $g(k) = 2^k + X_k + Y_k - 4$.

It is clear from Theorem A that Euler's conjecture for exponent $k$ is equivalent to the validity of the Diophantine problem $(**)$ for the same exponent.

The following theoretical and computational results concerning $(**)$ will enhance our belief in the validity of Euler's conjecture.

**Theorem B** (K. Mahler, 1957). *The relation* $(**)$ *holds when* $k$ *is sufficiently large* (*and so does Euler's conjecture*).

Unfortunately, Mahler's proof is based on the Ridout $p$-adic extension of Roth's theorem (cf. Lemma 2 below), and so it is not effective, since it just provides us with a bound for the number of exceptions to $(**)$.

**Theorem C** (R.-M. Stemmler, 1964). *The relation* $(**)$ *holds for* $2 \leq k \leq 200,000$ *(and so does Euler's conjecture for* $6 \leq k \leq 200,000$*).*

**Theorem D** (J. M. Kubina and M. C. Wunderlich, 1989). *The relation* $(**)$ *holds for* $2 \leq k \leq 471,600,000$ *and so does Euler's conjecture in the same range.*

In this paper, we pursue a double aim:

(1) to present two algorithms which lead respectively to the determination of $g(k)$ for a single value $k$ and to the determination of all values of $g(k)$ in the range $k \leq K$;
(2) to present a practical method for testing $(**)$ in an interval $k \leq K$ and discuss its complexity.

**Theorem 1.** *There exists an algorithm, the input of which is an integer* $K$ *and the output of which is the set of values* $(g(k))_{1 \leq k \leq K}$*, which runs in* $O(K^2)$ *bit operations.*

Since $g(k) \geq 2^k$, it is clear that Theorem 1 is best possible, up to improvement of the implied constant.

**Theorem 2.** *There exists an algorithm, the input of which is an integer* $k$ *and the output of which is the value* $g(k)$*, which requires* $O(k \log k \log\log k)$ *bit operations.*

Since $g(k) \geq 2^k$, the running time cannot be $o(k)$. However, we would like to stress that we do not know any faster algorithm (up to the implied constant) to compute $2^k + [(3/2)^k] - 2$.

The next result presents a "nondeterministic algorithm", or what could be even described as a "diplomatic algorithm", the output of which is YES or MAYBE. Its interest relies on the fact that, in practice, we expect the output to be always YES, i.e., it will show the validity of Euler's conjecture in a certain range.

**Theorem 3.** *There exists an algorithm which has the following properties*:

(i) *For an input* $K$*, the output is either YES (Euler's conjecture holds for* $K \leq k \leq 2K$*), or MAYBE (there might be some* $k \leq 2K$ *for which Euler's conjecture fails).*
(ii) *For an input* $K$*, the algorithm runs in* $O(K \log K \log\log K)$ *bit operations.*
(iii) *There exists an integer* $K_0$ *such that the answer is YES for* $K \geq K_0$.

The underlying algorithm uses FFT; in practice, it might well be more efficient to use the method of Karacuba and Toom, which easily gives a complexity $O(K^{1.58\ldots})$, a low cost improvement overe the standard $O(K^2)$ complexity.

The practical interest of Theorem 3 is that it may indeed be used for actual computation: the underlying basic idea is the one that is used for obtaining

Theorem D. In this light, one may notice that Proposition 1 would permit us to check Euler's conjecture for the range [175,600,000, 471,600,000] in five steps, Proposition 2 in seven steps, whereas Theorem D was proved in ten steps.

The theoretical interest of Theorem 3 is to tell us, that, to our knowledge, checking the validity of Euler's conjecture for all $k \leq K$ is not more complicated than writing down the single value $2^K + [(3/2)^K] - 2$.

One should notice that (iii) is neither completely trivial nor a direct consequence of Mahler's result, although it shares with it the dependence on the ineffective result of Ridout. Indeed, (iii) is a statement about the algorithm asserted to exist in the theorem; as such it proves again Mahler's theorem.

Theorems 1, 2, and 3 formalize a discussion held in Banff in May 1988 between Professor D. Shanks and the second-named author: Theorem 3 partly represents the point of view of Professor Shanks for whom the state of the Diophantine problem (**) will be satisfactory only when we know a bounded algorithm to settle it. On the other hand, Theorems 1 and 2 represent the point of view of the second-named author for whom the actual determination of $g(k)$ is as easy (or as difficult) to perform whatever the computation is based upon: Theorem A, or Euler's conjecture. The second-named author is thankful to Professor Shanks for that stimulating exchange.

The authors are grateful to the organizers of the NSF Computational Number Theory meeting held in Bowdoin College in July 1988, among other, for giving them this opportunity to meet.

## 2. Proof of Theorems 1 and 2

The value of $g(k)$ is trivial for $k = 1$, and, as we mentioned in the introduction, is already known for $k = 2$, 3, and 4. In 1964, Chen Jing-run proved that $g(5) = 37$. One can now check directly that Theorem A also applies for $k \leq 5$.

The proofs of Theorems 1 and 2 rely on Theorem A. Since the evaluation of the product $X_k \cdot Y_k$ may be cumbersome, we start with a more convenient reformulation of Theorem A.

### 2.1. Mathematical analysis of Theorem A.

**Lemma 1.** *In the notation of Theorem* A, *and assuming that* $\xi_k < (3/4)^k$, *we have the following equivalences*:

$$3^k \cdot \eta_k \geq 2^k - 1 \Leftrightarrow X_k \cdot Y_k > 2^k + 1,$$
$$3^k \cdot \eta_k < 2^k - 1 \Leftrightarrow X_k \cdot Y_k = 2^k + 1.$$

*Proof.* We begin with two remarks:

(i) The integer $X_k \cdot Y_k = ((3/2)^k + \xi_k) \cdot ((4/3)^k + \eta_k)$ is strictly larger than $2^k$, so it is either $2^k + 1$ or at least $2^k + 2$.

(ii) The integer $2^k([(3/2)^k] + 1) - 3^k$ is strictly positive, thus it is at least 1, and so $2^k \cdot \xi_k$ is at least 1, so that $(4/3)^k \cdot \xi_k \geq (2/3)^k$.

Assume that $3^k \cdot \eta_k \geq 2^k - 1$; then

$$X_k \cdot Y_k = ((3/2)^k + \xi_k) \cdot ((4/3)^k + \eta_k)$$
$$\geq 2^k + (2/3)^k + 1 - 1/2^k + \xi_k \cdot \eta_k > 2^k + 1.$$

Assume next that $3^k \cdot \eta_k < 2^k - 1$; then

$$X_k \cdot Y_k = ((3/2)^k + \xi_k) \cdot ((4/3)^k + \eta_k)$$
$$< 2^k + (3/4)^k (4/3)^k + 1 - (1/2)^k + (3/4)^k (2/3)^k$$
$$= 2^k + 1 + 1 - (1/2)^k + (1/2)^k = 2^k + 2,$$

so that $X_k \cdot Y_k = 2^k + 1$. □

**2.2. Proof of Theorem 1.** The proof is based on Lemma 1 and Theorem A. We are going to name a few "boxes" in the memory of a binary computer, leaving unnamed the auxiliary buffers. The main boxes will be $G_1, \ldots, G_K, A, B, C$.

We initialize the data by storing 3 in $A$ and 1 in $G_1, B$, and $C$.

At the end of the $(k-1)$st stage, we assume that we have filled the following boxes:

$G_1, \ldots, G_{k-1}$ (with $g(1), \ldots, g(k-1)$, respectively),

$A$ with $3^{k-1}$,

$B$ with $q_{k-1} := [(4/3)^{k-1}]$,

$C$ with $r_{k-1} := 3^{k-1} \cdot \{(4/3)^{k-1}\}$.

We now proceed from stage $(k-1)$ to stage $k$ (for $k \leq K$).

(i) compute $3^k$ and store it in $A$

(ii) compute $p_{k-1}$ and $s_{k-1}$ such that $q_{k-1} = 3p_{k-1} + s_{k-1}$ with $0 \leq s_{k-1} \leq 2$

(iii) compute $s_{k-1} \cdot 3^{k-1} + 4 \cdot r_{k-1} - 3^k$

- if it is negative, then store $q_k := q_{k-1} + p_{k-1}$ in $B$ and $r_k := s_{k-1} \cdot 3^{k-1} + 4 \cdot r_{k-1}$ in $C$, and go to (iv)
- if it is nonnegative, then store $q_k := q_{k-1} + p_{k-1} + 1$ in $B$ and $r_k := s_{k-1} \cdot 3^{k-1} + 4 \cdot r_{k-1} - 3^k$ in $C$, and go to (iv)

(iv) compute $u_k := [(3/2)^k]$ and $v_k = 4^k \cdot \{(3/2)^k\}$

- if $4^k - v_k - 3^k$ is positive, then store $2^k + u_k - 2$ in $G_k$, and go to next stage
- if $4^k - v_k - 3^k$ is negative or 0, then go to (v)

(v) compute $3^k - r_k - 2^k + 1$

- if it is nonnegative, store $2^k + u_k + q_k - 5$ in $G_k$, and go to next stage
- if it is negative, store $2^k + u_k + q_k - 4$ in $G_k$, and go to next stage

It is easy to verify that at the end of the $K$th stage, the boxes $G_1, \ldots, G_K$ are filled with $g(1), \ldots, g(K)$, respectively. We must now check that each step

requires only $O(K)$ bit operations:

(i) $3^k = 3^{k-1} + 2 \cdot 3^{k-1}$ (shift and add), then store $O(k)$ bits

(ii) Euclidean division by 3 can be performed in $O(K)$ bit operations, exactly in the same way as one usually performs a division (by 3, or 11, ... ) by hand

(iv) if one writes $3^k = \sum_j \varepsilon_j 2^j$, then $[(3/2)^k] = \sum_{j \geq k} \varepsilon_j 2^{j-k}$, and $4^k \cdot \{(3/2)^k\} = \sum_{j<k} \varepsilon_j 2^{j+k}$ with $\varepsilon_j \in \{0, 1\}$. □

**2.3. Proof of Theorem 2.** The $k$th step of the previous algorithm naturally splits into two phases:

- the computation of $3^k \cdot [(4/3)^k]$, and $3^k \cdot \{(4/3)^k\}$, which is performed in steps (i) to (iii)
- the determination of $g(k)$ from those values, in steps (iv) and (v).

To prove Theorem 2, it is thus sufficient to prove that one may get $3^k$, $[(4/3)^k]$, and $3^k\{(4/3)^k\}$, ab initio, in $O(k \log k \log\log k)$ bit operations.

Fast exponentiation is used to get $3^k$: one computes $t_l = 3^{2^l}$ for $2^l \leq k$ and then the product $t_{l_1} \cdots t_{l_s}$, where $k = \sum 2^{l_i}$; products are obtained by the Schönhage/Strassen FFT method in which the product of two integers of length $n$ requires at most $O(n \log n \log\log n)$ bit operations. If one is clever enough to compute the product of the $t_j$'s by first multiplying the smallest ones, the total computing time for the determination of $3^k$ does not exceed $O((k + k/2 + k/4 + \cdots) \log k \log\log k)$ bit operations.

We now write down $4^k$, which can be performed in $O(k)$ bit operations; it is then possible (cf. [6]) to determine integers $a$ and $b$ such that $4^k = a3^k + b$ with $0 \leq b < 3^k$ in $O(k \log k \log\log k)$ bit operations. We then notice that $a = [(4/3)^k]$ and $b = 3^k \cdot \{(4/3)^k\}$. □

**2.4. A second proof of Theorem 1.** It is possible to prove Theorem 1 by noticing that the Ridout-Mahler way of proving Theorem B gives indeed an effective bound for the number of exceptions to (**). Thus, one can follow the Stemmler route (i.e., the steps (i) and (iv) in the algorithm described in §2.2); if an appeal to step (v) is needed, we just compute $g(k)$ by the method described in §2.3. The total cost is $O(K^2) + O(1) \cdot O(K \log K \log\log K) = O(K^2)$. □

## 3. Proof of Theorem 3

**3.1. Statement of auxiliary results.** If the Diophantine condition (**) is not fulfilled for some $k$, then $3^k$ contains a long string of consecutive 1's in its binary expansion; but then $3^{k+1}$ will also contain a long (but maybe slightly shorter) string of consecutive 1's, and so on. However, on probabilistic grounds, $3^l$ should not contain a long string of consecutive 1's; thus, one has a good chance to verify this fact for some integer $l$, and then the condition (**)

will be fulfilled for many integers $k$ less than $l$; in Proposition 1, we give a precise formulation of this fact.

**Proposition 1.** *Let $m$ be a positive integer, and assume that $3^m$ does not contain, in its binary expansion, a block of $h$ consecutive 1's; then the Diophantine relation $(**)$ holds for*

$$m(\log 3/\log 4) + h/2 + 1/2 \leq k \leq m$$

*(and so does Euler's conjecture).*

It is now easy to deduce from Proposition 1 the following (the proof is left to the reader):

**Proposition 2.** *Let $N$ be an integer greater than 11 and consider the assertion*

$(\mathscr{P}_N)$ *None of the integers $3^{2^N+j\cdot 2^{N-2}}$ $(j = 1, 2, 3, 4)$ contains, in its binary expansion, a block of $(.8-(\log 3/\log 4))2^N-3$ consecutive 1's.*

*If $(\mathscr{P}_N)$ is true, then $(**)$ holds for $2^N < k \leq 2^{N+1}$.*

By using fast exponentiation and FFT, it is readily seen that one has the following

**Proposition 3.** *Assertion $(\mathscr{P}_N)$ can be checked in $O(2^N \cdot N \cdot \log N)$ bit operations.*

Our belief in the validity of $(\mathscr{P}_N)$ for $N \geq 12$ relies on a computation performed by Ch. Batut, which shows that a slightly weaker result holds for $12 \leq N \leq 18$. The work of J. M. Kubina and M. C. Wunderlich shows that the range $19 \leq N \leq 29$ lies within the range of present computations.

**Proposition 4.** *There exists an integer $N_0$ such that $(\mathscr{P}_N)$ is valid whenever $N$ is larger than $N_0$.*

All these results clearly imply Theorem 3.

**3.2. Proof of Proposition 1.** Let $k$ be an integer from the interval

$$[m(\log 3/\log 4) + (h+1)/2, m]$$

such that $(**)$ does not hold; with the notation of Theorem A, one has

$$X_k = (3/2)^k + \xi_k \quad \text{with } 0 < \xi_k < (3/4)^k,$$

and so there exist integers $M$ and $\alpha$ such that

$$3^m = 2^k M - \alpha \quad \text{with } 0 < \alpha < 3^m 2^{-k}.$$

Since $m(\log 3/\log 4) \leq k - (h+1)/2$, one has

$$0 < \alpha < 4^{k-(h/2)-(1/2)} \cdot 2^{-k} \leq 2^{k-h-1};$$

it follows from this that the $h$ digits of $3^m$ corresponding to $2^{k-h}, 2^{k-h+1}, \ldots, 2^{k-1}$ are all equal to 1. □

**3.3. Proof of Proposition 4.** We are indeed going to prove a stronger result, namely: *the length of the longest block of digits* 1 *in the binary expansion of* $3^m$ *is* $o(m)$.

This is a corollary to Ridout's approximation theorem, a special case of which reads:

**Lemma 2.** *Let* $\lambda$, $\mu$, *and* $c$ *be real numbers such that* $0 \le \mu \le 1$, $\lambda < 1-\mu$, *and* $c > 0$; *let* $p$ *be restricted to integers of the form* $p = 2^u M$ *with* $0 < M \le cp^\mu$; *then the inequality*

$$0 < |p - 3^v| < 3^{\lambda v}$$

*has only a finite number of solutions.*

We now prove our assertion. Let $\varepsilon > 0$ be given; without loss of generality, we may assume that $\varepsilon < 1$, and we have to prove that only finitely many integers $m$ are such that $3^m$ contains a block of $\varepsilon m$ consecutive 1's. We choose an integer $q > 5/\varepsilon$, and, for any $p$ with $0 \le p < q(\log 3/\log 2)$, we denote by $\mathcal{N}_p$ the set of all integers $m$ for which $3^m = \sum e_i 2^i$ with $e_i = 1$ for $pm/q \le i \le (p+2)m/q$. It is clear that each integer $m$ for which $3^m$ contains a block of $\varepsilon m$ consecutive 1's belongs to at least one set $\mathcal{N}_p$; it is thus sufficient to prove the finiteness of $\mathcal{N}_p$ for each given $p$.

For $m$ in $\mathcal{N}_p$, we have

$$3^m = \sum_{i<(pm/q)} e_i 2^i + \sum_{pm/q \le i \le (p+2)m/q} 2^i + \sum_{(p+2)m/q < i} e_i 2^i,$$

so that there exists $M_m$ such that

$$|2^{[(p+2)m/q]} M_m - 3^m| < 2^{[pm/q]+1}$$

(and for obvious arithmetical reasons, the left-hand side is different from 0). If we let

$$\mu = 1 - \frac{p+2}{q} \cdot \frac{\log 2}{\log 3} \quad \text{and} \quad \lambda = \frac{p+1}{q} \cdot \frac{\log 2}{\log 3},$$

we see that Ridout's result can be applied, at least for $m$ sufficiently large, and so the set $\mathcal{N}_p$ is finite. □

We should finally add that numerical evidence, as well as heuristics, suggest that the size of the longest block of digits 1 in the binary expansion of $3^m$ should be $O(\log m)$, with a small implied constant.

## Bibliography

1. R. Balasubramanian, J.-M. Deshouillers, and F. Dress, *Problème de Waring pour les bicarrés*. 1, 2, C.R. Acad. Sci. Paris Sér. I Math. **303** (1986), 85–88, 161–163.
2. L. E. Dickson, *History of the theory of numbers*. II, New York, 1934.
3. F. Dress, *Théorie additive des nombres, problème de Waring et théorème de Hilbert*, Enseign. Math. **18** (1972), 175–190.
4. H. Ehlich, *Zur Pillaischen Vermutung*, Arch. Math. **16** (1965), 223–226.

W. J. Ellison, *Waring's problem*, Amer. Math. Monthly **78** (1971), 10–36.

6. D. E. Knuth, *The art of computer programming*, Addison-Wesley, Reading, MA, 1981.
7. J. M. Kubina and M. C. Wunderlich, *Extending Waring's conjecture to* 471,600,000, Math. Comp. **55** (1990), (to appear).
8. K. Mahler, *On the fractional parts of powers of real numbers*, Mathematika **4** (1957), 122–124.
9. D. Ridout, *Rational approximations to algebraic numbers*, Mathematika **4** (1957), 125–131.
10. R. M. Stemmler, *The ideal Waring theorem for exponents* 401-200,000, Math. Comp. **18** (1964), 144–146.

CENTRE DE RECHERCHE EN MATHÉMATIQUES DE BORDEAUX (URA CNRS NO. 226), UNIVERSITÉ BORDEAUX I, F-33405, TALENCE CEDEX, FRANCE. *E-mail*: dezou@frbdx11.bitnet