INFORMATION AND COMPUTATION 78, 246–253 (1988)

Some Remarks on Computing the Square Parts of Integers

SUSAN LANDAU*

Mathematics Department, Wesleyan University,
Middletown, Connecticut 06457

Let n be a positive integer, and suppose $n = \prod p_i^{a_i}$ is its prime factorization. Let $\theta(n) = \prod p_i^{a_i - 1}$, so that $n/\theta(n)$ is the largest squarefree factor of n. We show that θ is deterministic polynomial time Turing reducible to φ, the Euler function. We also show that θ is reducible to λ, the Carmichael function. We survey other recent work on computing the square part of an integer and give upper and lower bounds on the complexity of solving the problem. © 1988 Academic Press, Inc.

1. INTRODUCTION

Integer factorization has intrigued mathematicians for centuries and computer scientists for over a decade. The problem seems quite hard, and that may be because it requires more than polynomial time. At this point no one knows. A useful approach is to study subproblems, in the hope that that will give an insight into the main question.

We say that a number is squarefree if it is not divisible by any square of a natural number except 1. We suggest that testing squarefreeness, and computing the square part of an integer is an appropriate question. In particular, in the analogous situation for polynomials, a polynomial time solution for determining the square part of a polynomial was known long before a polynomial time algorithm for factorization. We begin our study of computing square parts of integers by surveying some related work.

Suppose n is a positive integer with prime factorization $\prod p_i^{a_i}$. Let $\varphi(n)$ be the usual Euler phi-function, with

$$\varphi(n) = \prod p_i^{a_i - 1}(p_i - 1).$$

Also let

$$\lambda(n) = \operatorname{lcm}(p_1^{a_1 - 1}(p_1 - 1), ..., p_k^{a_k - 1}(p_k - 1))$$

* This work was performed while the author was visiting Yale University, and was partially supported by a Wesleyan University Project Grant.

0890-5401/88 $3.00

be the Carmichael function, and let

$$\lambda'(n) = \mathrm{lcm}(p_1 - 1, ..., p_k - 1).$$

Computing $\varphi(n)$ is deterministic polynomial time reducible to factorization. Miller showed that under the extended Riemann hypothesis (ERH), factoring is deterministic polynomial time reducible to φ (Miller, 1976). Long, and others, later showed that without any ERH assumption, factoring is random polynomial time reducible to φ (Long, 1981). The same is true for λ and λ'. (The ERH reduction is due to Miller, the random polynomial time reduction to Long. See (Miller, 1976, pp. 315–316) for a discussion of the ERH.)

Another number theoretic function of interest is $\sigma(n)$, which is the sum of the divisors of n; thus

$$\sigma(n) = \prod \frac{p_i^{a_i+1} - 1}{p_i - 1}.$$

Bach, Miller, and Shallit (1986) showed that factorization is random polynomial time reducible to computing $\sigma(n)$; clearly $\sigma(n)$ is deterministic polynomial time reducible to factoring.

If one considers the question of determining the squarefree part of a polynomial, the algorithm is very simple. Let $f(x)$ be the polynomial in question, and let $f'(x)$ be its derivative. If $g^k(x) \mid f(x)$, but $g^{k+1}(x) \nmid f(x)$, then $g^{k-1} \mid \gcd(f'(x), f(x))$, while $g^k(x) \nmid \gcd(f'(x), f(x))$. Thus $f(x)/\gcd(f'(x), f(x))$ is the largest squarefree factor of $f(x)$. By analogy we define

$$\theta(n) = \prod p_i^{a_i - 1},$$

so that $n/\theta(n)$ is the largest squarefree factor of n. Another function of interest is the decision function:

$$\theta'(n) = \begin{cases} 1 & \text{if } n \text{ is squarefree} \\ 0 & \text{otherwise.} \end{cases}$$

Clearly $\theta(n) = 1$ iff $\theta'(n) = 1$. Restricting our attention to deterministic polynomial time Turing reductions, let $f \leqslant_t g$ mean that we can compute f in deterministic polynomial time given an oracle for g. Woll (1987) was the first to observe any reduction regarding squarefreeness; she showed:

THEOREM 1 (Woll). $\theta' \leqslant_t \varphi$.

We extend this result to show that:

THEOREM 2. $\theta \leqslant_t \varphi$.

THEOREM 3. $\theta \leqslant_t \lambda$.

Despite the fact that factorization is ERH and random polynomial time reducible to λ', the same reduction which shows that $\theta \leqslant_t \varphi$ and $\theta \leqslant_t \lambda$ does not show that $\theta \leqslant_t \lambda'$. We discuss this in greater detail later.

2. MAIN RESULT

Let n be a positive integer, and suppose $\prod p_i^{a_i}$ is its prime factorization. Then $\varphi(n) = \prod p_i^{a_i - 1}(p_i - 1)$. This means that the

$$\gcd(n, \varphi(n)) = \prod p_i^{a_i - 1} \prod p_i^{e_i},$$

where e_i is 1 or 0 according to whether or not $p_i \mid q - 1$ for some prime q dividing n. It is the second part which makes things difficult; the first part of the gcd is exactly $\theta(n)$. Suppose we let

$$r = n/\gcd(n, \varphi(n)) = \prod p_i^{1 - e_i}.$$

Notice that r is never 1, since if $p_{\max}$ is the largest prime divisor of n, there will be no other prime divisor p_i of n with $p_{\max} \mid p_i - 1$. It is this idea which we exploit to create a reduction from θ to φ. To show that $\theta \leqslant_t \varphi$, we will compute a special factorization of n, one in which the primes which appear to power 1 are grouped together, those which appear to power 2 are grouped together, etc. Let $v[i]$ be the product of primes which appear to power i in n; then n can be written uniquely as

$$\prod v[i]^i,$$

with the $v[i]$ pairwise relatively prime and squarefree. For example, $6732 = 187 \times 6^2 = (11 \times 17) \times (2 \times 3)^2$. We present an algorithm (see Fig. 1), and prove correctness.

CLAIM. *Squarepart works correctly.*

Proof. We prove this by induction. For $n = 1, 2$, it is clear.

Suppose true for all $k < n$. Now if $\gcd(\varphi(n), n) = 1$, then n is squarefree, and the algorithm halts on line 4 with $v[1] = n$ and $v[i] = 1$ for $i = 2, \ldots, \lfloor \log n \rfloor$. Certainly it works correctly. So suppose that $k = \gcd(\varphi(n), n) \neq 1$. Then

$$k = \prod p_i^{a_i - 1} \prod p_i^{e_i},$$

```
**Squarepart**($n$)
**input**: $n$: an integer;
**output**: vector $v$ of length $\lfloor \log n \rfloor$ such that $v[i]$ is squarefree for each $i$, the
$v[i]$ are pairwise relatively prime, and $n = \prod (v[i])^i$;
 (1) BEGIN
 (2) FOR $i = 1$ TO $\lfloor \log n \rfloor$ DO $v[i] \leftarrow 1$;
 (3) $k \leftarrow \gcd(\varphi(n), n)$;
 (4) IF $k = 1$ THEN $v[1] \leftarrow n$, RETURN
 (5)         ELSE BEGIN
 (6)             $v \leftarrow$ squarepart($k$);
 (7)             $r \leftarrow n/k$;
 (8)             FOR $i = 1$ TO $\lfloor \log n \rfloor$ DO:
 (9)                 BEGIN
(10)                     $temp \leftarrow \gcd(r, v[i])$;
(11)                     $v[i] \leftarrow v[i]/temp$;
(12)                     $v[i+1] \leftarrow v[i+1] \cdot temp$;
(13)                     $r \leftarrow r/temp$;
(14)                 END
(15)         $v[1] \leftarrow v[1] \cdot r$;
(16)         END;
(17) END.
```

FIGURE 1.

where e_i is 1 or 0 according to whether or not $p_i | q - 1$ for some prime q dividing n. By induction we can assume that the vector v returned on line 6 is correct. Now

$$r = \prod p_i^{1 - e_i}.$$

Note that if $p_{\max}$ is the largest prime dividing n, then there is no prime p_i such that $p_{\max} | p_i - 1$. Thus $r \neq 1$.

By our induction assumption, $k = \prod (v[i])^i$, with the $v[i]$ squarefree and pairwise relatively prime. Also $n = k \cdot r$. Thus if p is a prime which divides both k and r, then p should appear to one higher power in the vector decomposition for n than it does in the one for k. This is accomplished by the calculations in lines (11)-(13). Any prime factor of n which appears in r but not in k is accounted for in line (15). Such a factor will have exponent 1 in n. ∎

CLAIM. *Assuming the call on φ takes constant time, there is a constant c such that $c \log^3 n$ is an upperbound on the running time of* Squarepart(n).

Proof. Let c be a constant such that $k = \gcd(\varphi(n), n)$, $r = n/k$, $temp \leqslant \gcd(r, k)$, $k/temp$ and $temp \cdot r$ can all be computed in less than $(c/5) \log^2(n)$ steps. Let $T(n)$ be the running time for Squarepart(n).

Certainly the claim is true for $n = 2$. Assume it holds for all $k < n$. Lines (1)–(5) take less than $(c/5) \log^2 n$ steps. Line (6) takes $T(k)$ steps, with $k < n/2$. Line (7) takes no more $(c/5) \log^2 n$ steps. Next consider lines (8)–(14): the loop is iterated $t = \lfloor \log n \rfloor$ times. Each of lines (10)–(13) take no more than $(c/5) \log^2 n$ steps. Actually we can do better. Let us consider line (10) in detail. Iterating line (10) will take no more than

$$(c/5)(\log^2 v[1] + \log^2 v[2] + \cdots + \log^2 v[t])$$

steps, which is less than

$$(c/5)(\log v[1] + \log v[2] + \cdots + \log v[t])^2$$

steps. Since $k = \prod v[i]^i$, the above expression is no bigger than $(c/5)(\log^2 k)$, and thus line (10) takes no more than $(c/5) \log^2 k$ steps. Lines (11)–(13) are similar. Therefore lines (8)–(14) take no more than $(4c/5) \log^2 n$ steps. Then

$$\begin{aligned} T(n) &\leqslant (c/5) \log^2 n + T(k) + (4c/5) \log^2 n \\ &\leqslant c \log^2 n + c \log^3(n/2) \\ &= c(\log^2 n + (\log n - 1)^3) \\ &\leqslant c \log^3 n. \quad \blacksquare \end{aligned}$$

Let $Sq(n)$ be the function which returns the vector $v[i]$ described above.

LEMMA 4. *Sq and θ are deterministic polynomial time Turing reducible to one another.*

Proof. $\theta(n) = \prod (v[i])^{i-1}$, thus $Sq \leqslant_t \theta$. The reverse is equally easy to show. Let $\theta^k(n) = \theta(\theta(\cdots(\theta(n)\cdots))$ applied k times. Then $v[1] = n/\theta(n)$, $v[2] = \theta(n)/\theta^2(n), \ldots,$ $v[\log n] = \theta^{\log n - 1}(n)/\theta^{\log n}(n)$. Since $\log n$ calls of $\theta(n)$ are allowed, the reduction is polynomial time. $\blacksquare$

This completes the proof of Theorem 2. But we should note that the algorithm Squarepart uses very little of the power of φ to compute the decomposition.

THEOREM 5. *Let $f(n)$ be any function from Z to Z such that if n is not squarefree, then $\gcd(f(n), n) \neq 1, n$. Then $\theta \leqslant_t f$. Assuming the call on f takes constant time, there is a constant c such that $c \log^4 n$ is an upperbound on the running time of* Squarepart(n).

Proof. Substitute $f(n)$ for $\varphi(n)$ in line 4 of the algorithm. We use two facts to show $\theta \leqslant_t \varphi$: (1) that $\gcd(\varphi(n), n)$ is neither 1 nor n whenever n is divisible by a nontrivial square, and (2) that r is squarefree. In general, we may have to call Squarepart(r) as well as Squarepark(k). Thus an upper bound on the running time is $O(\log^4 n)$ steps. ▮

COROLLARY 6. $\theta \leqslant_t \lambda$. *The time bound is the same as the one established for* $\varphi(n)$.

Proof. If $p^2 | n$, then $p | \lambda(n)$, and thus $\gcd(\lambda(n), n) \neq 1$. Since $\lambda(n) = \operatorname{lcm}(p_1^{a_1-1} p_1 - 1, ..., p_k^{a_k-1} p_k - 1) < n$, we know that $\gcd(\lambda(n), n) \neq n$. The time bound is the same since the only change is that $\lambda(n)$ is substituted for $\varphi(n)$ in line 3 of the program. ▮

We also have the following two lemmas.

LEMMA 7. $\lambda' \leqslant_t \lambda$.

Proof. This comes from the previous corollary and the fact that $\lambda'(n) = \lambda(n/\theta(n))$. ▮

We let $f \leqslant g \oplus h$ mean that f is deterministic polynomial time reducible to an oracle for both g and h.

LEMMA 8. $\lambda \leqslant_t \theta \oplus \lambda'$.

Proof. We know that $\lambda(n) = \operatorname{lcm}(p_1^{a_1-1}(p_1 - 1), ..., p_k^{a_k-1}(p_k - 1)) = \operatorname{lcm}(\prod p_i^{a_i-1}, p_1 - 1, ..., p_k - 1)$. But the first term is just $\theta(n)$, and $\operatorname{lcm}(a, b, c) = \operatorname{lcm}(a, \operatorname{lcm}(b, c))$. Thus we have $\lambda(n) = \operatorname{lcm}(\theta(n), \operatorname{lcm}(p_i - 1, ..., p_k - 1)) = \operatorname{lcm}(\theta(n), \lambda'(n))$. ▮

It is easy to see that the reduction of algorithm Squarepart does not show $\theta(n) \leqslant \lambda'(n)$, since there can be a prime p with $p^2 | n$ and yet $\gcd(\lambda'(n), n) = 1$; that is, λ' is insensitive to the exponents of the primes appearing in n. The fact is that $\lambda'(n)$ is a unusual function; under ERH factorization is deterministic polynomial time reducible to it, yet we have not been able to reduce $\theta(n)$ to it. Its power in factorization comes from the fact that it is a multiple of $p - 1$, for a prime factor p of n (Bach and Shallit, 1985). Both $\varphi(n)$ and $\lambda(n)$ share this characteristic, which is why factorization is (ERH and random) polynomial time reducible to the two functions. But $\varphi(n)$ and $\lambda(n)$ also have the characteristic that if $p^2 | n$ for some prime p, then the gcd of the function with n is nontrivial. The function $\lambda'(n)$ does not. For this reason we believe that λ' is a weaker function than φ or λ, and we suspect that θ is not reducible to λ'. We pose the following:

Open Question 1. What is the relationship among φ, λ, and λ'? Is $\varphi \leqslant_t \lambda'$? Is $\lambda \leqslant_t \lambda'$?

Open Question 2. What is the relationship between θ and λ'? We conjecture that λ' "does not help" to compute θ, by which we mean: if $\theta \leqslant_t \lambda'$, then θ is in polynomial time. (Note that conjecture implies that if ERH holds then θ is in P.)

3. Upper and Lower Bounds for $\theta(n)$

How hard is it to compute θ? Theorems 2, 3, and 4 give an upper bound for computing θ. This is the first deterministic reduction we know for θ other than the obvious one of integer factorization. Yet if we consider $\theta(pq)$, for p and q primes, the function θ gives us virtually no information. It would seem that θ should be a weaker function than factorization, and that the upper bound proved in this paper is too generous.

There is a connection between computing the basis for a ring of algebraic integers and computing the greatest square divisor of an integer. Several researchers have noted that given an irreducible quadratic polynomial f over Q, and an integral basis for the ring of integers defined by f, it is easy to find the squarefree part of the discriminant of f. This is simply because if $Q(\sqrt{d})$ is a quadratic extension of Q with d squarefree, then a basis for the ring of integers of that extension is:

$$1, \frac{1+\sqrt{d}}{2} \qquad \text{if} \quad d \equiv 1 \pmod{4}$$

$$1, \sqrt{d} \qquad \text{if} \quad d \equiv 2 \text{ or } 3 \pmod{4}.$$

Thus suppose we are given a polynomial $x^2 - d$ over Z, and a basis $\langle \alpha, \beta \rangle$ for the ring of integers of $K = Q(\sqrt{d})$. If $d \equiv 1 \pmod{4}$, then $\langle 1, (1+\sqrt{d'})/2 \rangle$ is also a basis for the ring of integers, where $d = d's^2$, with d' squarefree. Given $\langle \alpha, \beta \rangle$, in polynomial time we can compute d'. A similar technique will work for $d \equiv 2, 3$, or $0 \pmod{4}$. By iterating this procedure and performing the appropriate gcd computations, there is a deterministic polynomial time Turing reduction of $Sq(d)$ to determining the basis for the ring of integers of a quadratic extension of $Q(\sqrt{d})$.

Recently the reduction has been shown to be more general (Lenstra, 1987). More precisely:

Theorem 9 (Lenstra). *The problem of determining a basis for the ring of integers of an algebraic extension of Q is deterministic polynomial time Turing equivalent to the problem of computing Sq.*

This result is based on the work of Ford, Zassenhaus, and others.

Finally we have a lower bound on the complexity of computing $\theta(n)$. Let $x_1, \dots, x_n$ be a set of vertices of Z^k which span R^k. Thus $n > k$. We can view the $\{x_i\}$ as vertices of a polyhedra. One question of interest is whether there is a smaller "similar" (similar in the high school geometry sense of sides being in the same ratio) polyhedra still on the integer lattice. The cases of two, three, and four dimensions were solved by Cremona and Landau (1987). Surprisingly, both two and four dimensions have polynomial time algorithms, while the problem in three dimensions provides a lower bound for computing $\theta(n)$.

THEOREM 10 (Cremona and Landau). *Let $x_1, \dots, x_m$ be a set of vertices of Z^3 which span R3. Let $\bar{x}_i = x_i - x_1$, and let $x_i \cdot x_j$ mean the usual dot product of vectors. Then the polygon defined by $x_1, \dots, x_m$ can be shrunk iff* $\gcd_Z(\bar{x}_2 \cdot \bar{x}_2, \bar{x}_2 \cdot \bar{x}_3, \dots, \bar{x}_2 \cdot \bar{x}_m, \bar{x}_3 \cdot \bar{x}_3, \bar{x}_3 \cdot \bar{x}_4, \dots, \bar{x}_m \cdot \bar{x}_m)$ *is divisible by a nontrivial square in Z. The maximal shrinkage d can be computed in the time it takes to compute the square part of* $n = \gcd_Z(\bar{x}_2 \cdot \bar{x}_2, \bar{x}_2 \cdot \bar{x}_3, \dots, \bar{x}_2 \cdot \bar{x}_m, \bar{x}_3 \cdot \bar{x}_3, \bar{x}_3 \cdot \bar{x}_4, \dots, \bar{x}_m \cdot \bar{x}_m)$.

Let k be odd and greater than 3. Let $x_1, \dots, x_m$ be a set of vertices in Z_k which span R^k. If $n = \gcd_Z(\{x_i \cdot x_j \mid 1 \leqslant i \leqslant j \leqslant m\})$, then the figure spanned by the $\{x_i\}$ can be shrunk only if n is divisible by a square (Cremona and Landau, 1987). It is not known if this necessary condition is sufficient.

RECEIVED October 19, 1987; ACCEPTED December 8, 1987

REFERENCES

BABAI, L. (1987), personal communication.

BACH, E., MILLER, G., AND SHALLIT, J. (1986), Sums of divisors, perfect numbers and factoring, *SIAM J. Comput.* **15**, 1143–1154.

BACH, E., AND SHALLIT, J. (1985), Factoring with cyclotomic polynomials, *in* "Proceedings, 26th IEEE Symposium on Foundations of Computer Science, 1985," pp. 443–450.

CREMONA, J., AND LANDAU, S. (1987), "Shrinking Lattice Polyhedra," Tech. Report 87-05, Wesleyan University.

LENSTRA, H. (1987), Lectures at Bonn Workshop on Foundations of Computing, Bonn.

LONG, D. (1981), "Random Equivalence of Factorization and Computation of Orders," Tech. Report No. 284, Princeton University.

MILLER, G. (1976), Riemann's hypothesis and tests for primality, *J. Comput. System Sci.* **13**, 300–317.

WOLL, H. (1987), Reductions among number theoretic problems, *Inform. and Comput.* **72**, 167–179.