SUMS OF DIVISORS, PERFECT NUMBERS, AND FACTORING

Eric Bach (*)
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Gary Miller
Department of Computer Science
University of Southern California
Los Angeles, CA 90089

Jeffrey Shallit (*)
Department of Computer Science
University of Chicago
Chicago, IL 60637

ABSTRACT.

Let N be a positive integer, and let $\sigma(N)$ denote the sum of the divisors of N (e. g. $\sigma(6) = 1 + 2 + 3 + 6 = 12$). We show computing $\sigma(N)$ is equivalent to factoring N in the following sense: there is a random polynomial time algorithm that, given $\sigma(N)$, produces the prime factorization of N, and $\sigma(N)$ can be computed in polynomial time given the factorization of N.

We show that the same result holds for $\sigma_k(N)$, the sum of the k −th powers of divisors of N.

We give three new examples of problems that are in Gill's complexity class BPP: perfect numbers, multiply perfect numbers, and amicable pairs. These are the first ''natural'' sets in BPP that are not obviously in RP.

(*) Research sponsored in part by NSF grant MCS 82-04506.

Key words:
factoring, sum of divisors, perfect numbers, random reduction, multiply perfect numbers, amicable pairs

I. Introduction.

Integer factoring is a well-known difficult problem whose precise computational complexity is still unknown. Several investigators have found algorithms that are much better than the classical method of trial division (see [Guy1], [Pol], [Dix], [Len]).

We are interested in the relationship of factoring to other functions in number theory. It is trivial to show that classical functions like $\phi(N)$ (the number of positive integers less than N and relatively prime to N) can be computed in polynomial time if one can factor N; hence computing $\phi(N)$ is "easier" than factoring. One would also like to find functions "harder" than factoring. The first result in this area was given in Gary Miller's thesis [Mill]. Miller showed that if the Extended Riemann Hypothesis (ERH) is true, then given $\phi(N)$ one can produce the factorization of N in polynomial time. Thus computing $\phi(N)$ is "equivalent" to factoring. He also demonstrated a similar equivalence between factoring and two other number-theoretic functions, $\lambda(N)$ and $\lambda'(N)$ (defined below). Long pointed out that if one is willing to use randomization, the ERH assumption in the above results can be eliminated, and further showed that the calculation of orders in the multiplicative group of integers (mod N) is randomly equivalent to factoring [Long]. (Section II below gives a slightly more general version of these results.) Using the results of Miller and Long, a method for composite-modulus discrete logarithm problems implies a method for factoring [Bach].

In this paper, we demonstrate an equivalence between factoring and computing the function $\sigma(N)$, the sum of the divisors of N. More formally, we prove the following

Theorem 1.
Given the factorization of N, $\sigma(N)$ can be computed in polynomial time.

Theorem 2.
Given $\sigma(N)$, we can produce the factorization of N in random polynomial time.

Theorem 1 is easy to prove; for if

$$N = {p_1}^{e_1} \cdots {p_r}^{e_r}$$

then

$$\sigma(N) = \frac{{p_1}^{e_1+1} - 1}{p_1 - 1} \cdots \frac{{p_r}^{e_r+1} - 1}{p_r - 1}$$

$$= (p_1^{e_1} + \cdots + 1) \cdots (p_r^{e_r} + \cdots + 1). \tag{0}$$

(e. g. [HW, Thm. 275]). Thus $\sigma(N)$ can be computed in polynomial time. □

In sections III and IV below, we will prove Theorem 2.

Section V discusses extensions to $\sigma_k(N)$, the sum of the k-th powers of the divisors of N. Section VI discusses some interesting corollaries, including three examples of natural problems in Gill's complexity class BPP that are not obviously in RP.

(We assume the reader is familiar with probabilistic complexity classes, as discussed in [Gill]. Recall that BPP is the class of languages recognized in polynomial time by a probabilistic Turing machine, with *two*-sided error probability bounded by a constant away from 1/2. RP is the class of languages recognized in polynomial time by a probabilistic Turing machine with *one*-sided error.)

A few words about notation: we use N to denote a number to be factored, and p and q represent prime divisors of N. The factorization of N is given by

$$N = {p_1}^{e_1} \cdots {p_k}^{e_k}$$

We use $p^e \parallel N$ to mean $p^e \mid N$ but $p^{e+1} \nmid N$, i. e. p^e is the highest power of p dividing N. By $v_p(N)$ we mean the exponent of the highest power of p dividing N; e. g. in the example of the previous sentence, $v_p(N) = e$.

If R is a ring, we use R^* to denote the group of invertible elements. For example, $\mathbb{Z}_N$ is the ring of integers (mod N), and $\mathbb{Z}_N^*$ is the group of elements relatively prime to N. By GF(q^k) we mean the Galois field with q^k elements. $N_{E/F}(a)$ is the relative norm of the element a.

By *factoring* an integer N, we mean producing the complete factorization. By *splitting* we mean finding a nontrivial divisor.

$\lambda(N)$ denotes Carmichael's lambda function. $\lambda(N)$ is the exponent of the group $\mathbb{Z}_N^*$, i. e. the least positive e for which $a^e \equiv 1 \pmod N$ for all $a \in \mathbb{Z}_N^*$. It is easy to show that

$$\lambda(N) = \mathrm{lcm}_i \{ p_i^{e_i - 1}(p_i - 1) \}.$$

$\lambda'(N)$ is defined similarly:

$$\lambda'(N) = \mathrm{lcm}_i \{ p_i - 1 \}.$$

II. Splitting N given a multiple of $p - 1$.

Most of the equivalences between functions discussed in Section I are proved as follows: let N be composite with prime divisors p and q. By doing computations in $\mathbb{Z}_N$, and using the Chinese remainder theorem, we get the effect of doing computations in $\mathbb{Z}_p$ and $\mathbb{Z}_q$. Given a randomly chosen $a \in \mathbb{Z}_N$, we construct a number $x_a \in \mathbb{Z}_N$ such that $x_a \equiv 0 \pmod p$, but $x_a \not\equiv 0 \pmod q$ with high probability. Thus $\gcd(x_a, N)$ gives a nontrivial divisor of N. (This is one of the few general ideas for factoring integers.)

The first half ($x_a \equiv 0 \pmod p$) is usually proved by exploiting some algebraic structure; the second half ($x_a \not\equiv 0 \pmod q$) by showing that the set of $a \in \mathbb{Z}_N^*$ for which $x_a \equiv 0 \pmod q$ is a *proper* subgroup of the group $\mathbb{Z}_N^*$.

As an example, we now show how to split N given a multiple of $p-1$. This theorem and its proof can essentially be found in [Mill] and [Long]. However, we include it here for two reasons: for completeness and to motivate the main ideas.

Theorem 3.

There is an algorithm $\mathrm{S}(N, M, a)$ with the following properties:

Let N be odd and divisible by at least two distinct primes. Let $p|N$. Then given M such that $p-1 \mid M$, the algorithm $\mathrm{S}(N, M, a)$ splits N for at least 50% of the choices for $a \in \mathbb{Z}_N$, and terminates in time bounded by a polynomial in $\log M$ and $\log N$.

Proof.

The body of algorithm S is given below.

Algorithm $\mathrm{S}(N, M, a)$:

S1. [*Check for nontrivial GCD*]. If $\gcd(a, N) = r$ and $r \neq 1$, then return r and stop.

S2. [*Set exponent*]. Set $Q \leftarrow MN$.

S3. [*Compute power using modular exponentiation algorithm*]. Let $b \equiv a^Q \pmod N$.

S4. [*Test*]. If $b \not\equiv \pm 1$ then return $\gcd(b-1, N)$ and stop. Else if Q is even and $b \equiv 1$, set $Q \leftarrow Q/2$ and return to step S3.

S5. [*Q odd or $b \equiv -1$*]. Failure. Return nothing. Stop.

Lemma A.

For at least 50% of all choices of a, $1 \le a < N$, algorithm $\mathrm{S}(N, M, a)$ terminates after having produced a nontrivial divisor of N.

Proof of Lemma A.

If $a \in \mathbb{Z}_N - \mathbb{Z}_N^*$ then step S1 of the algorithm will always discover a nontrivial factor of N. Hence we may assume $a \in \mathbb{Z}_N^*$.

Let $p \mid N$. By assumption $p-1 \mid M$, so $p-1 \mid MN$. We examine two cases:

Case I: There exists at least one other prime $q \mid N$ such that $q-1 \nmid MN$. Then $a^{MN} \equiv a^{(p-1)kN} \equiv 1 \pmod p$ but

$$\{a \in \mathbb{Z}_N^* | a^{MN} \equiv 1 \pmod{q}\}$$

is a proper subgroup of $\mathbb{Z}_N^*$ and so $a^{MN} \not\equiv 1 \pmod{q}$ at for at least 50% of all choices of a. For these choices of a, step S4 produces a nontrivial divisor of N.

Case II: $q - 1 \mid MN$ for all primes $q \mid N$. Then $\lambda(N) \mid MN$, so

$$G_0 = \{a \in \mathbb{Z}_N^* | a^{MN} \equiv 1 \pmod{N}\} = \mathbb{Z}_N^*$$

Now consider the following chain of subgroups:

$$G_1 = \{a \in \mathbb{Z}_N^* | a^{MN/2} \equiv 1 \pmod{N}\}$$

$$G_2 = \{a \in \mathbb{Z}_N^* | a^{MN/4} \equiv 1 \pmod{N}\}$$

$$\vdots$$

$$G_k = \{a \in \mathbb{Z}_N^* | a^{MN/2^k} \equiv 1 \pmod{N}\}$$

where $k = \nu_2(MN)$, the largest exponent of 2 dividing MN. Clearly $G_k \neq \mathbb{Z}_N^*$ since $\lambda(N)$ is even, but $MN/2^k$ is odd. Hence there exists a subscript j for which $G_{j-1} = \mathbb{Z}_N^*$ but $G_j \neq \mathbb{Z}_N^*$. We claim that $H_j \neq \mathbb{Z}_N^*$ also, where H_j is a subgroup of $\mathbb{Z}_N^*$ given by

$$H_j = \{a \in \mathbb{Z}_N^* | a^{MN/2^j} \equiv \pm 1 \pmod{N}\}.$$

We will produce an $x \in \mathbb{Z}_N^*$ not in H_j. Let $q^e \parallel N$ such that

$$b^{MN/2^j} \equiv -1 \pmod{q^e}$$

for some b; such a q must exist, for otherwise G_j would equal $\mathbb{Z}_N^*$. Let x be given by

$$x \equiv b \pmod{q^e}$$

$$x \equiv 1 \pmod{N/q^e};$$

then

$$x^{MN/2^j} \equiv -1 \pmod{q^e}$$

$$x^{MN/2^j} \equiv 1 \pmod{N/q^e}$$

and so $x \notin H_j$. Thus H_j is a *proper* subgroup of $\mathbb{Z}_N^*$.

Now we claim that for each $x \notin H_j$, step S4 of the algorithm will produce a nontrivial divisor of N. This is because $x^{MN/2^{j-1}} \equiv 1 \pmod{N}$ but $x^{MN/2^j} \not\equiv \pm 1 \pmod{N}$ implies that $x^{MN/2^j} \equiv 1 \pmod{p^f}$ for some $p \mid N$ and $x^{MN/2^j} \equiv -1 \pmod{q^e}$ for some $q \mid N$. The conclusion is that at least 50% of all $a \in \mathbb{Z}_N^*$ will lead to a splitting of N in step S4.

This completes the proof of Lemma A. □

(We remark parenthetically that algorithm S works even if step S2 is replaced by

S2'. Set $Q \leftarrow M$.

However, with S2', the second corollary below might not be true, and this corollary is needed later in the paper.)

To complete the proof of Theorem 3, it suffices to verify that all the steps of algorithm S can be done in polynomial time. This is left to the reader. □

We immediately get the following corollary:

Corollary.
The functions $\phi(N)$, $\lambda(N)$, and $\lambda'(N)$ are randomly equivalent to factoring.

The following corollary will be needed in section IV.

Corollary.
Suppose we replace step S1 of Algorithm S with

S1$'$. If $\gcd(a, N) = r$ and $r \neq 1$, then return nothing and stop.

Then if d is a factor of N produced by the new algorithm, there is some prime $q \mid N$ such that $q \nmid d$.

Proof.
Just check the proof of Theorem 3. □

III. Splitting N using $\sigma(N)$: the square-free case.

In this section we assume that

$$N = p_1 p_2 \cdots p_k$$

is the product of one or more distinct primes. This case is somewhat easier than the case where N is divisible by a square, so we give our proofs in detail.

The following procedure will state that N is prime, or with high probability produce a non-trivial divisor of N.

(By iteration, if necessary, we eventually produce the complete factorization of N.)

Algorithm A. [*Given $\sigma(N)$ with N squarefree, try to split N.*]

A0. If $\sigma(N) = N + 1$, say "prime" and stop.

A1. If N is even, output the factor 2 and stop.

Repeat until N splits:

A2. Run a single iteration of algorithm S described in section II above, using an a chosen at random, and $M = \sigma(N)$. If a non-trivial divisor of N is produced, output that divisor and stop.

A3. Choose a random monic quadratic polynomial from $\mathbb{Z}_N[X]$, say, $f(X) = X^2 + bX + c$.

A4. Choose a random linear polynomial from $\mathbb{Z}_N[X]$, say, $r(X) = tX + u$ such that t and u are not both 0.

A5. [*Ensure that $r(X) \not\equiv 0 \pmod q$ for all primes $q \mid N$*]. If $\gcd(t, N)$ splits N, output that divisor and stop.

A6. Compute $dX + e = r(X)^{\sigma(N)} \pmod{(f(X), N)}$.

A7. If $\gcd(d, N)$ splits N, output that divisor and stop.

A8. [*Failure.*] No divisor of N has been produced on this iteration.

In our analysis of Algorithm A, we will find the following group-theoretic lemma useful:

Lemma B.

Let G be a finite cyclic group, $|G| = n$. Let σ be the homomorphism defined by $\sigma(g) = g^r$. Then $\sigma(G)$ is also a finite cyclic group. We have $|\sigma(G)| = \dfrac{n}{\gcd(n, r)}$ and if $g' \in \sigma(G)$ then $|\sigma^{-1}(g')| = \gcd(n, r)$. Hence $\sigma(G)$ is the trivial group iff $n \mid r$.

Proof.

See, for example, [Alb, Thm. 23, p. 19]. □

Here are the ideas behind Algorithm A:

Steps A0 and A1 are self-explanatory.

In step A2, if for any p_i dividing N we have $p_i - 1 \mid \sigma(N)$ then algorithm S will split N in polynomial time.

Hence let us assume that for all p_i we have $p_i - 1 \nmid \sigma(N)$. Pick a p_i and call it p.

Suppose $f(X)$ is a monic quadratic polynomial chosen at random from $\mathbb{Z}_N[X]$. Then a simple argument shows that with probability

$$\frac{1}{2} \cdot \frac{p-1}{p} \tag{1}$$

$f(X)$ is irreducible (mod p); so assume it is. (In practice, of course, we choose many different f and perform the algorithm on all of them. With high probability, the algorithm succeeds somewhere.)

Similarly, for a prime q, with probability

$$\frac{1}{2} \cdot \frac{q-1}{q} \tag{2}$$

$f(X)$ splits as the product of distinct linear factors (mod q), say $f(X) = (X - \beta)\,(X - \gamma)$ (mod q), so assume it does for some $p_j \neq p_i$ (call it q).

Lemma C.

With probability at least 1/2, $\gcd(d, N)$ splits N.

Proof.

We show that we always have $d \equiv 0 \pmod p$ but $d \not\equiv 0 \pmod q$ with probability $\geq 1/2$. From this we conclude that $\gcd(d, N)$ splits N with probability $\geq 1/2$.

To see that $d \equiv 0 \pmod p$ it is enough to see that

$$r(X)^{p+1} \pmod{f(X)} \in \mathbb{Z}_p.$$

Now $f(X)$ is irreducible (mod p); hence $\mathbb{Z}_p[X]/(f(X)) \cong \mathrm{GF}(p^2)$.

Now the p−th power automorphism gives the conjugate of the element $r(X)$ in $GF(p^2)$, so $r(X)^{p+1} = N_{GF(p^2)/GF(p)}(r(X))$ lies in the base field $GF(p)$ (see [Mar]). Thus $d \equiv 0 \pmod p$.

Now let us show that $d \not\equiv 0$ with probability $\geq 1/2$. By the Chinese Remainder Theorem, we have the isomorphism

$$\mathbb{Z}_q[X]/(f(X)) \cong \mathbb{Z}_q[X]/(X - \beta) \oplus \mathbb{Z}_q[X]/(X - \gamma)$$

Indeed, we can make this isomorphism explicit. There exist fixed $w_1 X + w_2$ and $v_1 X + v_2 \in \mathbb{Z}_q[X]$ such that every linear $r(X) \in \mathbb{Z}_q[X]$ can be written uniquely as

$$r(X) \equiv c_1(w_1 X + w_2) + c_2(v_1 X + v_2) \pmod q \tag{3}$$

Here the c_1 and c_2 are in $\mathbb{Z}_q$ and depend on $r(X)$. If c_1 and c_2 are both congruent to 0 (mod q), then step A5 of the algorithm above splits N, so we may assume that c_1 and c_2 are not both 0 (mod q).

Now

$$r(X)^{\sigma(N)} \equiv c_1{}^{\sigma(N)}(w_1 X + w_2) + c_2{}^{\sigma(N)}(v_1 X + v_2) \pmod q$$

so that

$$d \equiv c_1{}^{\sigma(N)} w_1 + c_2{}^{\sigma(N)} v_1 \pmod q$$

It is easy to see that $w_1, v_1 \not\equiv 0 \pmod q$, so if $d \equiv 0$ we must have

$$-c_1{}^{\sigma(N)} w_1 v_1{}^{-1} \equiv c_2{}^{\sigma(N)} \pmod q \tag{4}$$

We count the number of pairs (c_1, c_2) for which this can happen and show that for each $c_1 \pmod q$ at most 1/2 the values c_2 satisfy (4). If $c_1 \equiv 0$, then for (4) to hold we must have $c_2 \equiv 0$. If $c_2 \not\equiv 0 \pmod q$ then we may apply Lemma B to see that for any fixed value of c_1, the number of c_2 satisfying equation (4) is $\gcd(q-1, \sigma(N))$. But since $q - 1 \nmid \sigma(N)$, this is $\le \frac{q-1}{2}$. Hence the total number of nonzero pairs for which (4) can hold is $\le (q-1)^2/2$. Dividing this by $q^2 - 1$ (total pairs (c_1, c_2) with c_1, c_2 not both 0), we get $d \equiv 0 \pmod q$ with probability

$$\le \frac{1}{2} \, \frac{q-1}{q+1}$$

Hence with probability $\ge \frac{1}{2}$, we have $d \not\equiv 0 \pmod q$.

This completes the proof of Lemma C. □

Theorem 4.

Suppose N is odd, squarefree, and not prime. If $\sigma(N)$ is given, then with probability at least 1/15, a single iteration of steps A2 through A7 splits N.

Proof.

We multiply the probabilities given in equations (1) and (2) (using the worst case $p = 5$, $q = 3$) by the likelihood that step A7 splits N to get the worst case probability 1/15. □

A brief remark is in order. Algorithm A will work even if we have a non-zero *multiple* of $\sigma(N)$ instead of $\sigma(N)$ itself. The only difference is that in step A0 we must use a random polynomial-time prime test on N; for example, the probabilistic test given in [SS].

IV. Factoring N using $\sigma(N)$: the general case.

This section serves two purposes: we generalize the algorithm in section III to the case when N is not necessarily squarefree, and we show how to obtain the *complete* factorization of N, using only the single quantity $\sigma(N)$. Roughly speaking, this has the following complexitytheoretic import: the function ''prime factorization'' is many-one polynomial-time reducible to the function σ, not just Turing-reducible as one would first suppose.

For now, assume that we merely want to split $N = p_1^{e_1} \cdots p_k^{e_k}$. The algorithm below does this, using a guess α for one of the e_i's. Since $e_i \le \log_2 N$, we can try all possible α's without spoiling the polynomial time bound.

Algorithm B. [*Try to split N given $\sigma(N)$ and α*]:

B0. If N is a prime power, output $N = p^k$ and stop.

B1. If N is even, output a relatively prime factorization $N = 2^k \cdot M$ and stop.

Repeat until N splits:

B2. Try to split N using the algorithm S from section II, using $M = \sigma(N)$. If a nontrivial factor is obtained, output that factor and stop.

B3. Choose a random monic polynomial $f(X) \in \mathbb{Z}_N[X]$ of degree $\alpha + 1$.

B4. Choose a random polynomial $r(X) \in \mathbb{Z}_N[X]$ of degree $\le \alpha$.

B5. Compute
$h(X) = d_\alpha X^\alpha + \cdots + d_1 X + e = r(X)^{\sigma(N)} \pmod{f(X)}$.

B6. For each i, $1 \le i \le \alpha$, let $g_i = \gcd(d_i, N)$.

B7. If for some i, $1 < g_i < N$, output g_i and stop.

We hope that $f(X)$ is irreducible (mod p), but has at least two distinct irreducible factors (mod q). If this is the case, we call $f(X)$ **suitable**, and write

$$f(X) = \prod_{i=1}^{s} f_i(X)^{d_i}$$

with each $f_i(X)$ irreducible, $\deg(f_i) = k_i$, and $s \ge 2$. There is then a surjective ring homomorphism

$$\phi: \mathbb{Z}_q[X]/(f(X)) \to \mathrm{GF}(q^{k_1}) \oplus \cdots \oplus \mathrm{GF}(q^{k_s}) \quad (5)$$

(by the Chinese remainder theorem). We let K denote the kernel of ϕ, and let

$$\phi_i: \mathbb{Z}_q[X]/(f(X)) \to \mathrm{GF}(q^{k_i})$$

denote the ith projection map. The interesting fact about these projections is

Lemma E.

Let $h(X) \in \mathbb{Z}_q[X]/(f)$ have degree $\le \alpha$. If for some $i < j$, $\phi_i(h) \ne \phi_j(h)$, then one of h's non-constant coefficients is relatively prime to q.

Proof.

Assume that all of h's positive-degree coefficients vanish mod q. Then h is an element of GF(q), which is unchanged by every ϕ_i. The result follows by contraposition. □

We now need two probability estimates:

Lemma F.

A monic polynomial $f(X) \in \mathbb{Z}_N[X]$ of degree $\alpha + 1$ is suitable with probability at least

$$\frac{1}{\alpha+1}\,(1 - \frac{1}{\sqrt{p}})\,(1 - \frac{1}{q} - \frac{1}{\alpha+1})$$

Proof.

First, $f(X)$ is irreducible (mod p) with probability at least $(1 - 1/\sqrt{p})/(\alpha + 1)$. Second, f is irreducible mod q with probability at most $1/(\alpha + 1)$, and has a repeated factor mod q with probability exactly $1/q$ (see [Berl, p. 80] and [Carl]). □

Lemma G.

If $f(X)$ is suitable, then $r(X) \not\in K$ with probability at least $1 - 1/q^2$.

Proof.

By the rank-nullity theorem, $\dim_{GF(q)} K + \sum k_i = \alpha + 1$. Since there are at least two positive k_i's, the result follows. □

The main result on our algorithm is

Theorem 5.

If N is not prime, then for some $\alpha \le \log_2 N$, a single iteration of steps B0 through B7 splits N with probability at least

$$\frac{1}{32(\alpha+1)}\;.$$

Proof.

If N is a prime power or even, we get a nontrivial factorization. Therefore we can assume that N is odd, with two distinct prime factors p and q. If $q - 1 \mid \sigma(N)$, then by theorem 2, step

B2 will split N, so we can assume further that $q - 1 \nmid \sigma(N)$.

Now let $p^{\alpha} \parallel N$; $\alpha \le \log_2 N$ as claimed. Assume for now that $f(X)$ is suitable and that $r(X) \notin K$; we will estimate the probability that for some i, $g_i \equiv 0 \pmod p$ and $g_i \not\equiv 0 \pmod q$.

First, since f is suitable, $g_i \equiv 0 \pmod p$ for all i, since $\sigma(N)$ is a multiple of $p^{\alpha} + \cdots + p + 1$, the annihilator of $\mathrm{GF}(p^{\alpha+1})^* / \mathrm{GF}(p)^*$.

Now consider the situation (mod q), and let $c_i = \phi_i(r^{\sigma(N)})$. By the hypothesis that $r \notin K$, some $c_i \ne 0$; if some other $c_j = 0$, then by lemma E we must split N at step B7. Therefore we may as well assume that all the c_i's are nonzero, or, what is the same thing, $r(X)$ is a unit mod $f(X)$. Since we have assumed that $q - 1 \nmid \sigma(N)$, the map $r(X) \to \phi(r(X)^{\sigma(N)})$ does not annihilate $\mathbb{Z}_q[X]/(f(X))^*$. The image of this homomorphism is then a direct product of nontrivial cyclic groups, say $C_1 \times C_2 \times \cdots \times C_s$. The probability that a random element $(c_1, \ldots, c_s)$ will have all components equal is at most $1/\mathrm{lcm}\#(C_i) \le 1/2$; by Lemma E, then, the probability that some $g_i \not\equiv 0 \pmod q$ is at least 1/2.

Theorem 5 now follows by combining the last two paragraphs, Lemmas F and G, and the estimates $p, q \ge 3$, $\alpha \ge 1$. $\square$

We now turn to the problem of complete factorization. Our first observation is that $\sigma(N)$ can be replaced by any multiple of $\sigma(N)$ with no change in the statement of Theorem 5. Since $\sigma(N_1 N_2) = \sigma(N_1)\,\sigma(N_2)$ for relatively prime N_1 and N_2, we can use $\sigma(N)$ to recursively factor the pieces produced by Algorithm B, *provided* they are relatively prime. Therefore we need to transform the output of Algorithm B into a list of pairwise coprime factors.

Our solution to this problem hinges on the following concept. We say that a factorization $N = N_1\ N_2 \cdots N_r$ *segregates* p if $\nu_p(N_i) = \nu_p(N)$ for some i. A factorization segregates every prime if and only if the elements are pairwise relatively prime, and in this case $\sigma(N_i) \mid \sigma(N)$. The procedure below produces such a factorization, provided that some prime is segregated to begin with.

Algorithm R. [*Factor refinement procedure*]

[*At all times we have* $N = N_1^{e_1} \cdots N_r^{e_r}$, *possibly needing further processing.*]

R0. Let the initial factorization be $N = N_1\ N_2$.

While factors remain with $\gcd(N_i, N_j) > 1$:

R1. Set $D = \gcd(N_i, N_j)$.

R2. Replace $N_i^{e_i}$, $N_j^{e_j}$ in the list by $D^{e_i + e_j}$, $(N_i/D)^{e_i}$, $(N_j/D)^{e_j}$.

R3. If necessary, remove units from the list and combine powers of equal numbers.

The properties of this procedure are given by

Lemma D.

Algorithm R terminates in at most $\log_2 N$ iterations, with all the N_i's relatively prime. If the initial factorization is nontrivial and segregates some $p \mid N$, then on termination there are at least two factors.

Proof.

Left to the reader. □

It remains to show that using a multiple of $\sigma(N)$, we can split N and segregate some prime. This follows from the proof of Theorem 5 (recall that $q \nmid d_i$, so q is segregated) and the corollary to Theorem 3.

Thus we have completed the proof of Theorem 2, which we restate here:

Theorem 2.

Given $\sigma(N)$, we can produce the *complete* factorization of N in random polynomial time.

Corollary.

Computing the function $r_4(N)$, the number of ways to write N as the sum of four integer squares, is (randomly) equivalent to factoring.

Proof.

Suppose

$$N = 2^{e_1} {p_2}^{e_2} \cdots {p_k}^{e_k}$$

Then a classical theorem of Lagrange (see [HW, Theorem 386]) says

$$r_4(N) = \begin{cases} 8\sigma(N) & \text{if } e_1 = 0, 1 \\ 24 \dfrac{\sigma(N)}{2^{e_1+1} - 1} & e_1 \geq 2 \end{cases}$$

Since computing $\sigma(N)$ is randomly equivalent to factoring, the result follows. □

Similar results can be proved for functions like $r_8(N)$.

V. Generalization to $\sigma_k(N)$.

A natural generalization of $\sigma(N)$ is the sum of the k-th powers of divisors of N, i. e.

$$\sigma_k(N) = \sum_{d \mid N} d^k$$

$$= (p_1^{ke_1} + \cdots + p_1^k + 1) \cdots (p_r^{ke_r} + \cdots + p_r^k + 1)$$

where $N = p_1^{e_1} \cdots p_r^{e_r}$.

We also have a corresponding generalization regarding its computational complexity.

Theorem 6.
For any fixed integer $k \neq 0$, computing $\sigma_k(N)$ is (randomly) equivalent to factoring.

Proof.
If k is negative then

$$\sigma_k(N) = \frac{\sigma_{-k}(N)}{N}$$

so it suffices to consider positive k.

The essential idea is that the map $x \to x^{\sigma_k(N)}$ takes $\mathrm{GF}(p^{k(\alpha+1)})$ into $\mathrm{GF}(p^k)$, when $p^\alpha \parallel N$.

Algorithm C. [*Try to split N given* $\sigma_k(N)$]:

C0. If N is even or a prime power, output a factor and stop.
Set $\alpha \leftarrow 1$, and repeat until N splits:

C1. Try to split N using algorithm S with $M = \sigma_k(N)$.

C2. [*Construct* $\mathrm{GF}(p^k)$.] Pick a random monic polynomial $h(Y) \in \mathbb{Z}_N[Y]$ of degree k; let R denote $\mathbb{Z}_N[Y]/(h(Y))$.

C3. Pick a random monic $f(X) \in R[X]$ of degree $\alpha + 1$.

C4. Pick a random $r(X) \in R[X]$ of degree $\leq \alpha$.

C5. Compute $h(X) = d_\alpha(Y) X^\alpha + \cdots + d_1(Y)X + e(Y)$ $= r(X)^{\sigma(N)} \pmod{f(X)}$.

C6. For each i, $1 \leq i \leq \alpha$, and each coefficient t of $d_i(Y)$, see if $\gcd(t, N)$ splits N.

C7. [*Failure*]. If $\alpha + 1 < B$, where B is a bound on the exponents in the prime factorization of N, set $\alpha \leftarrow \alpha + 1$; else set $\alpha \leftarrow 1$. (We may take $B = \log_3(N)$.)

There is only one new observation to make here: we want $h(Y)$ to be irreducible modulo two distinct divisors of N, and this happens with probability about $1/k^2$. Since $k \le \log_2 \sigma_k(N)$, we only expect to wait a polynomial-bounded time until this happens. In all other respects, Algorithm C behaves just like algorithm B. The details are left to the reader. □

VI. Some classes of numbers that can be factored quickly.

The reduction of factoring to computing $\sigma(N)$ discussed in the previous sections allows us to quickly factor those numbers N for which $\sigma(N)$ is easily computable.

Consider the equation $\sigma(N) = 2N$. Numbers satisfying this equation are known as **perfect numbers**. The Pythagoreans attributed special properties to such numbers and this led to their intense study in antiquity, culminating in Euclid's proof that numbers of the form $2^{n-1}(2^n - 1)$ are perfect when the second factor is prime. In the 18th century, Euler proved that all *even* perfect numbers must be of this form. No one knows if there are any odd perfect numbers, but if there are, they must satisfy many stringent conditions (see, e.g., [teR]). We now add one more: they are all easy to factor!

More precisely, we show that the set {*perfect numbers*}, defined to be

$$\{x \in (0, 1)^*: x \text{ (interpreted in binary) is perfect } \},$$

is recognizable in (two-sided) random polynomial time, i. e. is a member of the complexity class BPP.

Theorem 7.
{*perfect numbers*} $\in$ BPP.

Proof.

Given N, *assume* that $\sigma(N) = 2N$. Run the algorithm of sections III-IV with the appropriate polynomial time bound; the result is a (purportedly complete) factorization of N. Now check to see if N is indeed perfect by using equation (0).

We end up accepting N if N is perfect, or if we accidentally produced an incorrect factorization (i. e. one where our probabilistic prime test said all the factors were prime, but some really weren't). But such an accident happens only ε of the time, and we can fix ε ahead of time.

We end up rejecting N if N is not perfect, or if we accidentally produced an incorrect factorization as above, or if the algorithm of sections III-IV failed to produce any factorization at all in our (pre-fixed) time bound. Again, this happens only ε of the time. □

Theorem 7 gives the first "natural" set in BPP which is not known to be in RP. Of course, it is possible to construct examples like

$$L = \{\ x\#y : x \text{ is prime and } y \text{ is composite } \}.$$

$L \in$ BPP, but it is somewhat "artificial", since it may be written as the product of two languages, one of which is known to be in RP, and one which is known to be in co-RP.

Nevertheless, Theorem 7 is very likely less interesting than it appears at first glance; if there are no odd perfect numbers (as is widely believed), then the clever Lucas-Lehmer test (see [Knu]) combined with the Euclid-Euler result for even perfect numbers gives a *deterministic* polynomial time algorithm to recognize {*perfect numbers*}.

However, there are well-studied generalizations of perfect numbers for which no deterministic tests are known. For example, numbers such that $\sigma(N) = 3N$ are sometimes called **sous-doubles**; examples are 120 and 672. It is easy to see that an argument like that in Theorem 7 shows that {*sous-doubles*} $\in$ BPP.

A larger class is the set of **multiply perfect numbers**; i. e., those numbers N for which $N \mid \sigma(N)$. To show that {*multiply perfect numbers*} $\in$ BPP, we need the following lemma:

Lemma J.

$$\sigma(N) < 5N \ln \ln N \qquad \text{for } N \geq 3.$$

Proof.

A well-known theorem (e. g. [HW, Thm. 329]) states that

$$\frac{\sigma(N)\phi(N)}{N^2} \leq 1$$

A result of Rosser and Schoenfeld [RS] is

$$\frac{N}{\phi(N)} < e^C \ln \ln N + \frac{3}{\ln \ln N}$$

for $N \geq 3$. Here C is Euler's constant, approximately $.5772$.

Combining these two inequalities, we get

$$\frac{\sigma(N)}{N} < (e^C + 3) \ln \ln N$$

for $N > e^e$. From this, the result easily follows. □

Lemma J shows that we can determine if N is multiply perfect with fewer than $5 \ln \ln N$ invocations of Algorithm B. This can be done in random polynomial time, so we have proved

Theorem 8.

{*multiply perfect numbers*} $\in$ BPP.

Carmichael [Carm] found the multiply perfect numbers less than 10^9:

1, 6, 28, 120, 496, 672, 8128, 30240, 32760, 523776,

2178540, 23569920, 33550336, 45532800, 142990848, 459818240.

(We have corrected several mistakes in Carmichael's original list.) It is not known whether or not there are infinitely many multiply perfect numbers. However, there are some density results that give upper bounds; for example, Hornfeck and Wirsing have shown [HoW] that if $m(x)$ denotes the number of multiply perfect numbers $\leq x$, then

$$m(x) = O(e^{\frac{c \ln x \ln \ln \ln x}{\ln \ln x}}) .$$

To give still another example, consider the pairs (M, N) such that

$$\sigma(M) = \sigma(N) = M + N$$

Such numbers are known as **amicable pairs**; the smallest pair is (220, 284). Jacob gave Esau 220 goats and 220 sheep [God], and some scholars have interpreted this as showing that the ancient Hebrews knew about $\sigma(N)$. There is an enormous literature concerning amicable pairs (see [LM]). An argument similar to those above gives

Theorem 9.
{amicable pairs} $\in$ BPP.

It is not known whether or not there are infinitely many amicable pairs (M, N), but Erdos conjectures that the number of such pairs with $M < N < x$ is at least $cx^{1-\epsilon}$ [Guy2].

Using our methods, it is possible to show that many other types of numbers (for example, the ''betrothed numbers'' of Isaacs [Guy2, p. 33]) can be recognized in two-sided random polynomial time.[1]

In Theorems 7-9 above, we have given three sets in BPP. The two-sidedness of these sets is due to the dependence on primality testing; if we had a deterministic polynomial-time prime test, we would be able to show that {*perfect numbers*}, {*multiply perfect numbers*}, and {*amicable numbers*} are in RP. No such prime test is currently known, although there is one due to Adleman, Pomerance, and Rumely [APR] which runs in time $O((\log N)^{c \log \log \log N})$.

VII. Epilogue.

[1]Betrothed pairs (M, N) satisfy $\sigma(M) = \sigma(N) = M + N + 1$;
we note reluctantly that a pair cannot
be both betrothed and amicable.

In section II, we showed how to split N given a multiple of $p - 1$. The results on $\sigma(N)$ can be phrased similarly; if we know a multiple of $p + 1$ (or $p^2 + p + 1$, etc.) we can split N. This leads to the question: for which polynomials $f(p)$ do there exist fast algorithms for splitting N? We will address this question in a future paper [BS].

The complexity of several number-theoretic functions is still open. One example is computing discrete logarithms (mod p).

Not every difficult number-theory function is equivalent to factoring; some are apparently *harder*. For example, remarks of Shanks indicate that factorization is reducible to finding the class number of an imaginary quadratic field [Shan] but no reduction in the other direction is known, nor is it even clear that this problem is in NP.

VIII. Acknowledgements.

A preliminary version of this paper was presented at the 16th ACM Symposium on the Theory of Computing in Washington, D. C. [BMS].

Much of the research for this paper was done while the first and third authors were graduate students at the University of California at Berkeley; they would like to express their deep appreciation to Manuel Blum, who created an environment eminently suitable to conducting research.

We are pleased to acknowledge the use of the computer algebra program VAXIMA, which allowed us to confront our early ideas with the harsh reality of specific examples.

Thanks also go to the referees, especially to one who produced a particularly thorough list of improvements.

REFERENCES

[Alb] A. A. Albert, *Fundamental Concepts of Higher Algebra*, University of Chicago Press, 1956.

[APR] L. M. Adleman, C. Pomerance, and R. S. Rumely, 'On distinguishing prime numbers from composite numbers', *Ann. Math.* **117** (1983) 173-206.

[Bach] E. Bach, 'Discrete logarithms and factoring', UC Berkeley Computer Science Division Report UCB/CSD/84/186 (1984).

[BMS] E. Bach, G. Miller, and J. Shallit, 'Sums of divisors, perfect numbers, and factoring', *Proc. 16th Annual ACM Symp. Theor. Comput.* (1984) 183-190.

[BS] E. Bach and J. Shallit, 'Factoring with cyclotomic polynomials', to appear, *26th IEEE Symp. Found. Comp. Sci.*, 1985.

[Berl] E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, 1968.

[Carl] L. Carlitz, 'The arithmetic of polynomials in a Galois field', *Amer. Journ. Math.,* **54** (1932) 39-50.

[Dix] J. D. Dixon, 'Asymptotically fast factorization of integers', *Math. Comp.* **36** (1981) 255-260.

[Gill] J. Gill, 'Computational complexity of probabilistic Turing machines', *Siam J. Comput.* **6** (1977) 675-695.

[God] *Genesis*, xxxii, 14.

[Guy1] R. K. Guy, 'How to factor a number', *Proc. Fifth Manitoba Conf. on Numerical Math.*, Winnipeg, 1976, 49-89.

[Guy2] R. K. Guy, *Unsolved Problems in Number Theory*, Springer-Verlag, New York, 1981.

[HW] G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*, Oxford, 1971.

[HoW] B. Hornfeck and E. Wirsing, 'Uber die Haufigkeit vollkommener Zahlen', *Math. Annalen* **133** (1957) 431-438.

[Knu] D. E. Knuth, *The Art of Computer Programming*, V. II (Seminumerical Algorithms), 2nd edition, Addison-Wesley, Reading, Mass. (1981) 391-394.

[LM] E. J. Lee and J. S. Madachy, 'The history and discovery of amicable numbers', *Journ. Rec. Math.* **5** (1972) 77-93, 153-173, 231-249.

[Len] H. W. Lenstra, Jr., 'Elliptic Curve Factorization', typewritten ms., February 14, 1985.

[Long] D. L. Long, 'Random equivalence of factorization and computation of orders', to appear, *Theoretical Computer Science*.

[Mar] D. A. Marcus, *Number Fields*, Springer-Verlag, New York, 1977.

[Mill] G. Miller, 'Riemann's hypothesis and tests for primality', *J. Comp. System Sci.* **13** (1976) 300-317.

[Pol] J. M. Pollard, 'Theorems on factorization and primality testing', *Proc. Cambridge Phil. Soc.***76** (1974) 521-528.

[RS] J. B. Rosser and L. Schoenfeld, ‘Approximate formulas for some functions of prime numbers’, *Ill. Journ. Math.* **6** (1962) 64-94.

[Shan] D. Shanks, ‘Class number, a theory of factorization, and genera’, *Proceedings of Symposia in Pure Mathematics*, V. 20 (1969 Number Theory Institute), American Mathematical Society (1971) 415-440.

[SS] R. Solovay and V. Strassen, ‘A fast Monte-Carlo test for primality’, *Siam J. Computing* **6** (1977) 84-5.

[teR] H. J. te Riele, ‘Perfect numbers and aliquot sequences’, in Computational Methods in Number Theory, Amsterdam Math. Centre Tracts **154** (1982) 141-157.