TURNING EULER'S FACTORING METHOD INTO A FACTORING ALGORITHM

JAMES MCKEE

ABSTRACT

An algorithm is presented which, given a positive integer n, will either factor n or prove it to be prime. The algorithm takes $O(n^{1/3+\varepsilon})$ steps.

1. *Introduction*

Suppose that n is a positive integer. This paper describes an algorithm which will factor n, or prove it to be prime, in $O(n^{1/3+\varepsilon})$ steps. Here, and in similar expressions, ε is an arbitrarily small positive real number, with the implied constant in the O depending on ε. Of course, an algorithm as slow as this is of no practical interest compared to the best, probabilistic, factoring methods. On the other hand, algorithms which are *guaranteed* to factor n in time better than $O(n^{1/2+\varepsilon})$ are rather rare creatures, and so are perhaps of interest as mathematical curiosities, even if they have no other merit.

The first $O(n^{1/3+\varepsilon})$ factoring algorithm was devised by R. Sherman Lehman [5]. H. W. Lenstra Jr presented another algorithm [6], which, although not intended primarily as a factoring algorithm, again showed that n could be factored on $O(n^{1/3+\varepsilon})$ steps. Using FFT techniques, J. M. Pollard [7] and V. Strassen [9] showed that an $O(n^{1/4+\varepsilon})$ algorithm is possible, although completely impractical for reasonable sizes of n. This remains the best qualitative result. If one knew the generalised Riemann hypothesis, then Shanks' class group methods would factor n in time $O(n^{1/5+\varepsilon})$ [8].

In this paper, we present another $O(n^{1/3+\varepsilon})$ algorithm, which develops a factoring method of Euler. Euler observed that if one can express n in the form x^2+dy^2 in two essentially distinct ways, with the same d, then one can factor n. Indeed, if $n = x_1^2+dy_1^2 = x_2^2+dy_2^2$, then $(x_1y_2)^2 \equiv (x_2y_1)^2 \pmod n$, so that the greatest common divisor of $x_1y_2-x_2y_1$ and n will be a non-trivial factor of n unless $x_1y_2 \equiv \pm x_2y_1 \pmod n$. Many factoring methods have been based on this observation. Euler's method applies only to special numbers; here we extend it to cover all cases, in a practical way. The algorithm is soon better than trial division in practice, although, of course, it is much slower than the best probabilistic methods. (In a crude implementation, the algorithm was found (for worst-case numbers) to run about as fast as naive trial division for 8-digit numbers, about twice as fast for 10-digit numbers, about ten times as fast for 14-digit numbers and about twenty times as fast for 16-digit numbers.)

In the next section we describe the algorithm, giving an example of its use, with a brief discussion of practical considerations. Then the running time is established. Finally, the validity of the algorithm is proved.

Received 5 December 1994; revised 9 March 1995.

1991 *Mathematics Subject Classification* 11A51, 11E25.

Bull. London Math. Soc. 28 (1996) 351–355

2. *The algorithm*

Here a description of the algorithm is given in a manner which is intended to illustrate its $O(n^{1/3+\varepsilon})$ behaviour. In particular, the choice of d in Step 1 of the algorithm is rather larger than one should use in practice, but such considerations are not important, since in practice one should instead use a probabilistic method.

We suppose now that n is an odd integer greater than 1. The algorithm either will find a *non-trivial* factor of n (that is, a factor other than 1 or n), or, by failing to do so, will prove n to be prime.

Step 1. Check that n is not the square or higher power of a prime, else we have a non-trivial factor of n and can stop. Let x_0 be the greatest integer below $\sqrt{(n-n^{2/3})}$, and set $d = n - x_0^2$, so that $d \approx n^{2/3}$. Factorise d (by trial division, say), and compute quadratic non-residues modulo each odd prime dividing d. If the greatest common divisor of d and n is a non-trivial factor of n, then stop. Compute all square roots of n modulo d. Let $\eta_1, \ldots, \eta_w$ be these square roots.

Step 2. Test for factors of n below $(4d/3)^{1/4}$, by trial division. If a non-trivial factor of n is found, then stop.

Step 3. For $1 \leqslant a \leqslant \sqrt{(4d/3)}$, search for solutions to the equation

$$an = x^2 + dy^2 \tag{1}$$

with x, y positive integers, and $y^2 \neq a$. To this end, first check if a is a square mod d, else there can be no solutions to (1). If a is a square mod d, compute one of its square roots, α, say. To solve (1), if possible, search through x between 1 and $\sqrt{(an)}$ with $x \equiv \alpha\eta_i \pmod{d}$, for $i = 1, \ldots, w$. If a solution $an = x_1^2 + dy_1^2$ is found, with $y_1^2 \neq a$, then proceed to Step 4. If no such solution is found for a in the given range, then n is prime.

Step 4. We have $n = x_0^2 + d$ and $an = x_1^2 + dy_1^2$. Compute the greatest common divisor of n and $x_0 y_1 - x_1$. This will be a non-trivial factor of n.

Example. Take $n = 1082\,154235\,955237$. Then

$$n = 32\,896084^2 + 1893\,420181,$$

so we may take $x_0 = 32\,896084$ and $d = 1893420181$ in Step 1 of the algorithm. The algorithm will work for any x_0 between 1 and $\sqrt{n}$, so there is a wide choice for d. In the above description it was suggested that one should seek $d \approx n^{2/3}$, but this was merely to simplify the evaluation of the running time in the next section. Given d, Step 3 of the algorithm requires the computation of $O(\sqrt{d})$ square roots mod d. For each a which is a square mod d, Step 3 considers $O(w\sqrt{(an)}/d)$ values of x in (1). If d falls below $n^{2/3}$, then there are more values of x to check, but fewer square roots to compute. The computation of a square root mod d is more expensive than checking if given values of x and a lead to a solution to (1), so one should in practice take d considerably smaller than $n^{2/3}$. Here we have a value of d which is prime, and this helps the book-keeping with regard to computing square roots mod d.

To complete Step 1, we note that 2 is a non-residue mod d, which will help with the computation of future square roots. The square roots of $n \bmod d$ are just $\pm x_0$.

For Step 2 we use trial division up to $(4d/3)^{1/4} \approx 224.15$. No factor of n is found. At this point we might check that n is not a square, cube, …, sixth power—if it were a higher power of a prime, then the trial division would have spotted it. Alternatively, we may leave this check until the end, since it is so rarely necessary.

Step 3: we search through $1 \leqslant a \leqslant \sqrt{(4d/3)} \approx 50245.0$, and try to solve (1). Actually, we may start the search with a as large as 12562, rather than 1, since any solution to (1) with $a < \sqrt{(4d/3)}/4$ would yield a solution with $\sqrt{(4d/3)}/4 \leqslant a \leqslant \sqrt{(4d/3)}$ by multiplying x and y by a suitable power of 2. Here we find

$$43036n = 2591\,866961^2 + d \cdot 145101^2.$$

(For $a = 43036$, the search was over $1 \leqslant x \leqslant 6824\,338041$, with $x \equiv \pm 698\,446780 \pmod d$, so involved checking just 7 values of x.) This yields the factorisation

$$n = 12\,345701 \times 87\,654337$$

in Step 4.

3. *Running time*

We wish only to show that the algorithm runs in time $O(n^{1/3+\varepsilon})$ steps, and so take crude estimates when they are good enough. As a preliminary remark, note that w, the number of square roots of $n \pmod d$ is $O(n^\varepsilon)$. Indeed, if d has r distinct prime factors, then $w = O(2^r)$. Now $r = O(\log d/\log\log d) = O(\log n/\log\log n)$ [3], so $w = O(2^{\log n/\log\log n}) = O(n^{\log 2/\log\log n}) = O(n^\varepsilon)$.

Step 1. Checking that n is not a square or higher power of a prime can be done in polynomial time (that is, $O((\log n)^r)$ steps, for some constant r). Computing x_0 and d is also a polynomial time task. Using trial division to factor d takes time $O(n^{1/3+\varepsilon})$. Computing non-residues of odd primes dividing d takes time $O(d^{1/4+\varepsilon}) = O(n^{1/6+\varepsilon})$ [1]. Computing the square roots $\eta_1, \ldots, \eta_w$ of $n \bmod d$ takes time $O(wn^\varepsilon)$ once the non-residues are known [4], and $w = O(n^\varepsilon)$. Hence Step 1 takes $O(n^{1/3+\varepsilon})$ steps.

Step 2. $O(d^{1/4}) \cdot O(n^\varepsilon) = O(n^{1/6+\varepsilon})$ steps.

Step 3. There are $O(d^{1/2}) = O(n^{1/3})$ values of a to consider. Checking if a is a square $\bmod d$ can be done in polynomial time, given that we know the prime factorisation of d. Computing α (for given a) can also be done in polynomial time. For each a there are $O(w\sqrt{(an)}/d) = O(w) = O(n^\varepsilon)$ values of x to consider, each of which can be tested in polynomial time. Hence Step 3 takes $O(n^{1/3+\varepsilon})$ steps.

Step 4. Polynomial time.

Thus the total time is $O(n^{1/3+\varepsilon})$ steps, as claimed.

4. *Validity of the algorithm*

We need a lemma, the proof of which is standard (see, for example, [2]).

LEMMA. *Let $d > 1$ be an integer and n an odd, positive integer prime to d. Then the number of distinct square roots of $-d \bmod n$ is exactly half the number of proper representations of n by reduced binary quadratic forms $ax^2+2bxy+cy^2$ with $b^2-ac = -d$.*

Here *proper* means that in the representation $n = ax^2+2bxy+cy^2$, the greatest common divisor of x and y must be 1, and *reduced* means that $|2b| \leqslant a \leqslant c$, with $b \geqslant 0$ if either $a = |2b|$ or $a = c$. In particular, if $ax^2+2bxy+cy^2$ is reduced, with $b^2-ac = -d$, then $4d = 4ac-(2b)^2 \geqslant 3a^2$, so that $a \leqslant \sqrt{(4d/3)}$.

Using this lemma we shall show that if n is composite and has no factor below $(4d/3)^{1/4}$, then Step 3 of the algorithm will find a solution to (1) with $y^2 \neq a$, and that any such solution will lead to a non-trivial factor of n in Step 4. We suppose, then, that n is composite and that we have reached Step 3 of the algorithm. Thus

$$n = x_0^2 + d, \tag{2}$$

and n has no prime factors below $(4d/3)^{1/4}$.

From (2), $-d$ is a square mod n. Since n is odd, and not a prime power, and n and d have greatest common divisor 1, $-d$ must have at least four square roots mod n. By the lemma, there are at least eight proper representations of n by reduced forms of discriminant $-4d$. Four of these are given by $n = (\pm x_0)^2 + d\cdot(\pm 1)^2$, but there must be others. Let

$$n = ax_2^2 + 2bx_2y_2 + cy_2^2 \tag{3}$$

be such a representation, so that if $a = 1$ (which implies $b = 0$ and $c = d$), then $y_2 \neq \pm 1$. Completing the square gives

$$an = x_1^2 + dy_1^2, \tag{4}$$

where $x_1 = ax_2 + by_2$, $y_1 = y_2$. Adjusting signs if necessary, we may suppose $x_1, y_1 \geqslant 0$.

Thus we are certain to find a solution to (1), and provided $y_1^2 \neq a$ we shall proceed to Step 4. Suppose, if possible, that $y_1^2 = a$. Then (3) implies that y_1 divides n. If $y_1 > 1$, then we would have a non-trivial factor of n below $\sqrt{a}$, hence below $(4d/3)^{1/4}$, and this possibility was eliminated in Step 2. If $y_1 = 1$, then we would have $a = 1$, and (3) would not be essentially distinct from (2), which we insisted it must be. Thus y_1^2 cannot equal a, and we proceed to Step 4.

We have $(x_0y_1)^2 \equiv x_1^2 \pmod{n}$. The greatest common divisor of n and $x_0y_1 - x_1$ will be a non-trivial factor of n unless $x_0y_1 \equiv \pm x_1 \pmod{n}$.

From (4) and (2), we have

$$0 < x_0 < \sqrt{n}, \quad 0 < y_1 < \sqrt{(an/d)}, \quad 0 \leqslant x_1 < \sqrt{(an)}. \tag{5}$$

(If $y_1 = 0$, then $x_2 = \pm 1$, else (3) is not a proper representation of n, but then $n = a < d < n$, which is nonsense.) From (5),

$$0 < x_0y_1 + x_1 < n(\sqrt{(a/d)} + \sqrt{a}/\sqrt{n}) < 2n\sqrt{a}/\sqrt{d} < n,$$

provided $d > 21$. This means that we never find $x_0y_1 \equiv -x_1 \pmod{n}$. There remains the possibility that $x_0y_1 \equiv x_1 \pmod{n}$, which, from the bounds in (5), implies $x_0y_1 = x_1$. Then (4) gives

$$an = y_1^2(x_0^2 + d) = y_1^2 n,$$

so $y_1^2 = a$. We have eliminated this possibility already, and conclude that the factor of n found in Step 4 must be a non-trivial one.

For an example of the necessity of Step 2, consider $n = 789 = 26^2 + 113$, with $d = 113$, $x_0 = 26$. The other forms $ax^2 + 2bxy + cy^2$ properly representing n with discriminant $-4d$ are given by $a = 9$, $b = \pm 2$, $c = 13$, with $x = \pm 8$, $y = \pm 3$ (the sign of y depending on the signs of b and x), so that $a = y_2^2$ in (3). Completing the square gives $9n = 78^2 + 113\cdot 3^2$, which is just our original representation of n multiplied through by 9. Here, of course, $\sqrt{9} = 3$ divides n, and $3 < (4d/3)^{1/4} \approx 3.50$, so the factor 3 is discovered in Step 2 of the algorithm.

References

1. D. A. Burgess, 'On character sums and primitive roots', *Proc. London Math. Soc.* 12 (1962) 179–192.
2. H. Davenport, *The higher arithmetic* (6th edn, Cambridge University Press, 1992).
3. G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers* (5th edn, Oxford University Press, 1979).
4. N. Koblitz, *A course in number theory and cryptography*, Graduate Texts in Math. 114 (Springer, New York, 1987).
5. R. Sherman Lehman, 'Factoring large integers', *Math. Comp.* 28 (1974) 637–646.
6. H. W. Lenstra Jr, 'Divisors in residue classes', *Math. Comp.* 42 (1984) 331–340.
7. J. M. Pollard, 'Theorems on factorization and primality testing', *Proc. Cambridge Philos. Soc.* 76 (1974) 521–528.
8. R. J. Schoof, 'Quadratic fields and factorization', *Computational methods in number theory, Part II*, Math. Centre Tracts 155 (ed. H. W. Lenstra Jr and R. Tijdeman, Mathematisch Centrum, Amsterdam, 1982).
9. V. Strassen, 'Einige Resultate über Berechnungskomplexität', *Jahresber. Deutsch. Math.-Verein.* 78 (1976/77) 1–8.

Department of Pure Mathematics and Mathematical Statistics
16 Mill Lane
Cambridge CB2 1SB