SMOOTH NUMBERS AND THE QUADRATIC SIEVE

Carl Pomerance*

When faced with a large number n to factor, what do you do first? You might say "Look at the last digit," with the idea of cheaply pulling out possible factors of 2 and 5. Sure, and more generally, you can cheaply test for divisibility by all of the very small primes. So it may as well be assumed that the number n has no small prime factors, say below $\log n$. Since it is also cheap to test for probable primeness, say through the strong probable prime test,[1] and then actually prove primality[2] in the case that you become convinced n is prime, it also may as well be assumed that the number n is composite.

Trial division is a factoring method (and in the extreme, a primality test) that involves sequentially trying n for divisibility by the consecutive primes. This method was invoked above for the removal of the small prime factors of n. The only thing stopping us from continuing beyond the somewhat arbitrary cut off of $\log n$ is the enormous time that would be spent if the smallest prime factor of n is fairly large. For example, if n were a modulus being used in the RSA cryptosystem, then as current protocols dictate, n would be the product of two primes of the same order of magnitude. In this case, factoring n by trial division would take roughly $n^{1/2}$ steps. This already is an enormous calculation if n has thirty decimal digits, and for numbers only slightly longer, the calculation is not possible at this time by the human race and all of their computers.

Difference of squares

We have long known however that trial division is not the only game in town for factoring. Take the number 8051 for example. This number is composite and not divisible by any prime up to its logarithm. One can see instantly (if one looks) that it is $8100 - 49$, that is,

$$8051 = 90^2 - 7^2.$$

Thus, we can use algebra to factor 8051 as a difference of squares. It is $(90-7) \times (90+7)$, or 83×97. Every odd composite can be factored as a difference of squares (an easy exercise), so why don't we use this method instead of trial division?

Let us try again on the number $n = 1649$. Again, 1649 is composite, but not divisible by any prime up to its logarithm. What worked with 8051 was to take the first square above 8051, namely 90^2, and then notice that $90^2 - 8051 = 49$, where 49 is recognized as a square. It would seem in general that one could walk through the sequence $x^2 - n$ with $x = \lceil n^{1/2} \rceil, \lceil n^{1/2} \rceil + 1, \ldots$, looking for squares. With $n = 1649$ we have

$$\begin{aligned} 41^2 - n &= 32, \\ 42^2 - n &= 115, \\ 43^2 - n &= 200, \\ &\;\vdots \end{aligned} \tag{1}$$

* I am indebted to John Voight for the careful notes he took at my lecture, and to Lancelot Pecquet, who offered numerous improvements for an earlier version of this article.

[1] See the lecture of S. Wagon.

[2] See my lecture on primality.

with no squares in immediate sight.

Despite this failure, the above 3 equations may in fact be already used to factor n. Note that while neither 32 nor 200 is a square, their product, which is 6400, *is* a square, namely 80^2. Thus, since

$$41^2 \equiv 32 \pmod{n},$$
$$43^2 \equiv 200 \pmod{n},$$

we have $41^2 \times 43^2 \equiv 80^2 \pmod{n}$, that is,

$$(41 \times 43)^2 \equiv 80^2 \pmod{n}. \tag{2}$$

We have found a solution to $a^2 \equiv b^2 \pmod{n}$. Is this interesting? There are surely plenty of uninteresting pairs a, b with $a^2 \equiv b^2 \pmod{n}$. Namely, take any value of a and let $b = \pm a$. Have we exhausted all solutions with this enumeration? Well no, since factoring n as a difference of squares would give a pair a, b with $b \neq \pm a$. Further, any pair a, b with

$$a^2 \equiv b^2 \pmod{n}, \quad b \not\equiv \pm a \pmod{n} \tag{3}$$

must lead to a nontrivial factorization of n, via $\gcd(a-b, n)$. Indeed, (3) implies that n divides $(a-b)(a+b)$, but divides neither factor, so both $\gcd(a-b, n), \gcd(a+b, n)$ must be nontrivial factors of n. Moreover, gcd's are simple to compute via Euclid's algorithm of replacing the larger member of $\gcd(u, v)$ by its residue modulo the smaller member, until one member reaches 0. Finally, if n has at least two different odd prime factors, then at least half of the solutions to $a^2 \equiv b^2 \pmod{n}$ with ab coprime to n also have $b \not\equiv \pm a \pmod{n}$, that is, (3) is satisfied.[3]

So, I repeat the question, is our solution to (2) interesting? Here, we have $a = 114 = 41 \times 43 \bmod n$ and $b = 80$. Yes, we do have a, b satisfying (3), so yes this is an interesting pair. We now compute $\gcd(114 - 80, 1649)$. The first step of Euclid gives $\gcd(34, 17)$, and the second step gives 17. That is, 17 is a divisor of 1649. Division reveals the cofactor, it is 97, and a couple of primality tests show that our factorization is complete.[4]

If we had actually waded through the sequence in (1) looking for a square, we would have had to go all the way to $57^2 - n = 40^2$, and clearly in this example, trial division would have been superior. In general, trying to factor a number as a difference of squares is inferior to trial division. But by altering n by multiplying it by small odd numbers, one can

[3] The proof: For an odd prime power p^u, the congruence $y^2 \equiv 1 \pmod{p^u}$ has exactly 2 solutions, so since n is divisible by at least two different odd primes, the congruence $y^2 \equiv 1 \pmod{n}$ has at least 4 solutions. Label these values of y as $y_1, y_2, \ldots, y_s$, where $y_1 = 1, y_2 = -1$. A complete enumeration of the pairs of residues a, b modulo n that are coprime to n and with $a^2 \equiv b^2 \pmod{n}$ then is all pairs $a, y_i a$, where a runs over residues coprime to n and $i = 1, \ldots, s$. Two out of s of these pairs have $i = 1, 2$ and $s - 2$ out of s of these pairs have $i = 3, \ldots, s$. The latter pairs satisfy (3), and since $s \geq 4$, we are done.

[4] Actually, since we have previously checked that 1649 has no prime factors up to its logarithm, about 7.4, we already know that 17 and 97 have no prime factors below their square roots, and so are both prime.

then skew things so that in the worst case the time spent is only about $n^{1/3}$ steps, essentially a factoring method of R. S. Lehman. For example, if one tries to factor $5 \times 1649 = 8245$ by a difference of squares, the first try works, namely $8245 = 91^2 - 6^2 = 97 \times 85$. Taking the gcd of the factors with 1649 reveals the factorization we are looking for.

A crucial lemma

But these thoughts lead down a different road. We would like to pursue what looked like perhaps fortuitous good luck with the equations in (1). If one were to try and describe what we did as an algorithm that works for a general number n, the part where we pick out the first and third of the equations to get the right sides to multiply to a square looks a bit suspect. In general we have this sequence of integers $x^2 - n$ as x runs starting above $n^{1/2}$ and we wish to pick out a subsequence with product a square. Surely we should not be expected to examine all subsequences, since the number of these grows exponentially.

Let us look at a probabilistic model for this problem. We are presented with a random sequence of integers in the interval $[1, X]$ and we wish to stop the sequence as soon as some non-empty subsequence has product a square. After how many terms do we expect to stop, how do we recognize when to stop, and how do we find the subsequence? The answers to these questions involve *smooth numbers*.

A number m is *smooth* if all of its prime factors are small. Specifically, we say m is B-smooth if all of its prime factors are $\leq B$. Simple enough, but what do smooth numbers have to do with our problem? The first observation is that if a number in our sequence is *not* smooth, then it is unlikely it will be used in a subsequence with product a square. Indeed, if the number m is divisible by the large prime p, then if m is to be used in the square subsequence, then there necessarily must be at least one other term m' in the subsequence which also is divisible by p. (This other term m' may be m itself, that is, perhaps $p^2|m$.) But given that p is large, multiples of p will be few and far between, and finding this mate for m will not be easy. So, say we agree to choose some cut off B, and discard any number from the sequence that is not B-smooth. Let us look at the numbers that are not discarded, these B-smooth numbers. The following lemma is crucial.

Lemma. *If $m_1, m_2, \ldots, m_k$ are positive B-smooth integers, and if $k > \pi(B)$ (where $\pi(B)$ denotes the number of primes in the interval $[1, B]$), then some non-empty subsequence of (m_i) has product a square.*

Proof. For a B-smooth number m, look at its *exponent vector* $\mathbf{v}(m)$. This is a simple concept. If m has the prime factorization

$$m = \prod_{i=1}^{\pi(B)} p_i^{v_i},$$

where p_i is the ith prime number and each exponent v_i is a nonnegative integer, then $\mathbf{v}(m) = (v_1, v_2, \ldots, v_{\pi(B)})$. So, a subsequence $m_{i_1}, \ldots, m_{i_t}$ has product a square if and only if $\mathbf{v}(m_{i_1}) + \ldots + \mathbf{v}(m_{i_t})$ has all even entries. That is, if and only if this sum of vectors is the 0-vector mod 2. Now the vector space $\mathbf{F}_2^{\pi(B)}$, where $\mathbf{F}_2$ is the finite field with 2 elements, has dimension $\pi(B)$. And we have $k > \pi(B)$ vectors. So this sequence of vectors is linearly dependent in this vector space. However, a linear dependency when the field of scalars is $\mathbf{F}_2$ is exactly the same as a subsequence sum being the 0-vector. This completes the proof of the lemma.

One should notice that this proof suggests an answer to our algorithmic question of how to find the subsequence. The proof uses linear algebra, and this subject is rife with algorithms. Well no, there is really only one algorithm that is taught in beginning linear algebra classes, namely Gaussian reduction of a matrix, and then there are many, many applications of this one technique. Well, now we have another application. With a collection of smooth numbers, form their exponent vectors, reduce these modulo 2, and then use Gaussian reduction to find a nonempty subsequence with sum the 0-vector modulo 2.

In particular, we shall find the concept of an exponent vector very useful in the sequel. Note that knowledge of the complete exponent vector of a number m is essentially equivalent to knowledge of the complete prime factorization of m. Also note that though in the proof of the lemma we wrote out exponent vectors in traditional vector notation, doing so in general may itself be an exponentially huge problem, even if we "know" what the vector is. Luckily, exponent vectors are sparse creatures with most entries being zero, so that one can work with them modulo 2 in a more compact notation that merely indicates where the odd entries are.

A proto-algorithm

So now we have a proto-algorithm. We are given a number n which is composite, not a power,[5] and has no prime factors up to its logarithm. Our goal is to find a nontrivial factorization of n.

1. Choose a parameter B, and examine the numbers $x^2 - n$ for B-smooth values, where x runs through the integers starting at $\lceil n^{1/2} \rceil$.
2. When you have more than $\pi(B)$ numbers x with $x^2 - n$ being B-smooth, form the exponent vectors of these B-smooth numbers, and use linear algebra to find a subsequence $x_1^2 - n, \ldots, x_t^2 - n$ which has product a square, say A^2.
3. From the exponent vectors of the numbers $x_i^2 - n$, we can produce the prime factorization of A, and thus find the least nonnegative residue of A modulo n, call it a. Find too the least nonnegative residue of the product $x_1 \ldots x_t$ modulo n, call it b.
4. We have $a^2 \equiv b^2 \pmod{n}$. If $a \not\equiv \pm b \pmod{n}$, then compute $\gcd(a-b, n)$. Otherwise, return to step 1, find additional smooth values of $x^2 - n$, find a new linear dependency in step 2, and attempt once again to assemble congruent squares to factor n.

There are clearly a few gaps in this proto-algorithm. One is a specification of the number B. Another is a fuller description of how one examines a number for B-smoothness, namely how one recognizes a B-smooth number when presented with one.

Recognizing smooth numbers

[5] We insist that n not be a power in order to ensure that n is divisible by at least two different odd primes. It is easy to check if a number is a power by taking roots via Newton's method, and for close calls to integers, exponentiating that integer to see if n is a power of it.

Let us look at the second gap first, namely the recognition of B-smooth numbers. Trial division is a candidate for a good method, even though it is very slow as a worst-case factorization algorithm. The point is that B-smooth numbers are very far from the worst-case of trial division, in fact they approach the best case. There are fewer than B primes up to B, so there are very few trials to make. In fact the number of trial divisions is at most the maximum of $\log_2 n$ and $\pi(B)$.

But we can do better. Let us review the sieve of Eratosthenes. One starts with a list of the numbers from 2 to some bound X. Then one recognizes 2, the first unmarked number, as prime, and crosses off every second number starting at 4, since these are all divisible by 2 and hence not prime. The next unmarked number is 3, which is the next prime, and then every third number is crossed off after 3, and so on. If one completes this with primes up to $X^{1/2}$, then every remaining unmarked number is prime. But we are not so interested in the unmarked numbers, rather the marked ones. In fact, the more marked a number the better, since it is then divisible by many small primes. This sieve procedure can rather easily be made completely rigorous. Indeed, interpret making a "mark" as taking the number in the relevant location and replacing it with its quotient by the prime being sieved. In addition, sieve by higher powers of the primes as well, again dividing by the underlying prime. If one does this procedure with the primes up to B and their higher powers, the B-smooth numbers up to X are detected by locations that have been transformed into the number 1. The time to do this is proportional to X times the sum of the reciprocals of the primes up to B and their powers up to X. This sum, see (4), is about $\log\log B$. Thus, in time proportional to $X \log\log B$ (for B at least 16, say, to have $\log\log B > 1$) we can locate all of the B-smooth numbers up to X. That is, *per number*, we are spending only about $\log\log B$ steps on average. This count compares with about B steps per number for trial division, a very dramatic comparison indeed. Further, there is a trick for reducing the complexity of the individual steps in the sieve involving the subtraction of low-precision logarithms to simulate the division. So it is a win-win for sieving.

However, we are not so interested in locating smooth numbers in the interval from 2 to X, but rather in the polynomial sequence $x^2 - n$ as x runs. This is not a big hurdle, and essentially the same argument works. What makes the sieve of Eratosthenes work is the regular places in the sequence where we see a multiple of p. This holds as well for any polynomial sequence. One solves the quadratic congruence $x^2 - n \equiv 0 \pmod{p}$. For $p > 2$ there are either no solutions, 2 solutions, or in the case that $p|n$, just 1 solution. Of course the generic cases are 0 and 2 solutions, and each should occur roughly half the time. If there are no solutions, then there is no sieving at all. If there are 2 solutions, say a_1, a_2, where $a_i^2 - n \equiv 0 \pmod{p}$, then we find the multiples of the prime p in the sequence for each $x \equiv a_i \pmod{p}$ for $i = 1, 2$. For example, say $n \equiv 4 \pmod{7}$, so that $a_1 = 2, a_2 = 5$. Then the values of x where 7 divides $x^2 - n$ can be found in quite regular places, namely those values of x that are congruent to 2 or 5 modulo 7. After an initial value is found that is 2 (mod 7), one can find the remaining places in this residue class by adding 7's sequentially to the location number, and similarly for the 5 (mod 7) residue class. One has a similar result for higher powers of p and also for the special case of $p = 2$. As with the sieve of Eratosthenes above, the number of steps we will spend per polynomial value is proportional to just $\log \log B$. That is, it takes about as much time to tell if a value is smooth as it does just to look at the value, as long as we amortize the time over many members of the polynomial sequence. So this is how we recognize B-smooth values of $x^2 - n$, we use a *quadratic sieve*.

An optimization problem

Next, in our proto-algorithm, there should be some guidance on the choice of the parameter B. Of course, we would like to choose B *optimally*, that is, we should choose a value of B which minimizes the time spent to factor n. This optimization problem must balance two principal forces. On the one hand, if B is chosen very small, then we do not have to find very few B-smooths in the sieve, and the matrix we will be dealing with will be small. But numbers that are B-smooth with a very small value of B are very sparsely distributed among the natural numbers, and so we may have to traverse a very long sequence of x values to get even one B-smooth value of $x^2 - n$, much less the requisite number of them. On the other hand, if B is chosen large, then B-smooth numbers are fairly common, and perhaps we will not have such a hard time finding them in the polynomial sequence $x^2 - n$. But, we will need to find a great many of them for large B, and the matrix we will be dealing with will be large.

So, the optimization problem must balance these conflicting forces. To solve this problem, we should have a measure of the liklihood that a value $x^2 - n$ is B-smooth. This in fact is a very hard problem in analytic number theory, one that is essentially unsolved in the interesting ranges. However, the number n we are trying to factor may not be up on the latest research results! That is, perhaps we should be more concerned with what is *true* rather than what is *provable*, at least for the design of a practical algorithm. This is where heuristics enter the fray. Let us assume that a polynomial value is just as likely to be smooth as a random number of the same magnitude. This assumption has been roughly borne out in practice with the quadratic sieve algorithm, as well as other factorization methods such as the number field sieve.[6]

What is the order of magnitude of our polynomial values? If x runs in the interval $[n^{1/2}, n^{1/2} + n^{\epsilon}]$, where $0 < \epsilon < 1/2$, then the numbers $x^2 - n$ are all smaller than approximately $2n^{1/2+\epsilon}$. Let X be this bound, and let us ask for the chance that a random number up to X is B-smooth.

An analytic tidbit

In analytic number theory we use the notation $\psi(X, B)$ for the counting function of the B-smooth numbers in the interval $[1, X]$. That is,

$$\psi(X, B) = \#\{m : 1 \leq m \leq X,\ m \text{ is } B\text{-smooth}\}.$$

Let us try our hand at estimating this function in the special case that $B = X^{1/2}$. We can do this by an inclusion-exclusion, with a single exclusion, since no number up to X is divisible by two primes $> X^{1/2}$. Thus,

$$\psi(X, X^{1/2}) = \lfloor X \rfloor - \sum_{X^{1/2} < p \leq X} \lfloor X/p \rfloor,$$

[6] See the lecture of P. Stevenhagen.

where p runs over primes in the stated interval. This identity uses the fact that there are exactly $\lfloor X/p \rfloor$ multiples of p in the interval $[1, X]$. And the multiples of p are definitely not $x^{1/2}$-smooth, so must be excluded. By removing the floor functions in the above display, we create an error bounded by $\pi(X)$, so that the prime number theorem implies that

$$\psi(X, X^{1/2}) = X\left(1 - \sum_{X^{1/2} < p \leq X} 1/p\right) + O(X/\log X).$$

The prime number theorem also implies that [5]

$$\sum_{p \leq t} 1/p = \log\log t + C + O(1/\log t), \tag{4}$$

for a particular constant C, so that

$$\begin{aligned}
\sum_{X^{1/2} < p \leq X} 1/p &= \sum_{p \leq X} 1/p - \sum_{p \leq X^{1/2}} 1/p \\
&= \log\log X - \log\log(X^{1/2}) + O\left(1/\log(X^{1/2})\right) \\
&= \log 2 + O(1/\log X).
\end{aligned}$$

We thus have that

$$\psi(X, X^{1/2}) = (1 - \log 2)X + O(X/\log X),$$

that is, we have

$$\frac{\psi(X, X^{1/2})}{X} \sim 1 - \log 2 \quad \text{as} \quad X \to \infty.$$

For example, about 30% of all numbers have no prime factors above their squareroot. It may seem surprising that such a large proportion of numbers can be built out of so few primes.

The u^u philosophy

In fact one can easily see that the exact same argument shows that

$$\frac{\psi(X, X^{1/u})}{X} \sim 1 - \log u$$

for each fixed value of u in the interval $[1, 2]$. But what of larger values of u? Here we have the celebrated result of K. Dickman that

$$\frac{\psi(X, X^{1/u})}{X} \sim \rho(u) \tag{5}$$

[5] The prime number theorem can be used in the form $\pi(t) = t/\log t + O(t/(\log t)^2)$, and then partial summation is applied to estimate the sum in (4). Actually, (4) is a theorem of Mertens that predates the prime number theorem.

for each fixed $u \geq 1$, where $\rho(u)$ is the Dickman–de Bruijn function. This function is defined as the continuous solution to the differential difference equation $u\rho'(u) = -\rho(u-1)$ for $u > 1$, with initial condition $\rho(u) \equiv 1$ on the interval $[0,1]$. The function $\rho(u)$ is always positive, and as u grows, it decays to 0 roughly like u^{-u}. A result of E. R. Canfield, P. Erdős, and myself is that even if u is not fixed, we still have something like (5) holding. Namely, for $X \to \infty, u \to \infty$ subject to $X^{1/u} > (\log X)^{1+\epsilon}$, we have

$$\frac{\psi(X, X^{1/u})}{X} = u^{-(1+o(1))u}, \tag{6}$$

for any fixed $\epsilon > 0$.

The choice of the smoothness bound

Let $B = X^{1/u}$. The expression $X/\psi(X, X^{1/u})$ is approximately equal to the reciprocal of the probability that a random integer up to X is B-smooth (it is exactly this reciprocal if X is an integer), and so $X/\psi(X, X^{1/u})$ is about equal to the expected number of random trials of choosing numbers in $[1, X]$ to find one which is B-smooth. However, we would like to have about $\pi(B)$ numbers that are B-smooth and sieving allows us to spend about $\log\log B$ steps per candidate, so the expected number of steps to find our requisite stable of B-smooths is about $\pi(B)(\log\log B)X/\psi(X, X^{1/u})$. Our goal is to minimize this expression. It is a bit ungainly, but if we make the rough approximations $\pi(B)\log\log B \approx X^{1/u}$, $X/\psi(X, X^{1/u}) \approx u^u$, we are looking at the simpler expression

$$X^{1/u}u^u.$$

We would like to choose u so as to minimize this expression. Take logarithms: so we are to minimize

$$\frac{1}{u}\log X + u\log u.$$

The derivative is 0 when

$$u^2(\log u + 1) = \log X.$$

Taking the log of this equation, we find that $\log u \sim \frac{1}{2}\log\log X$, so that

$$u \sim (2\log X \log\log X)^{1/2}$$

and

$$B = \exp\left((2^{-1/2} + o(1))(\log X \log\log X)^{1/2}\right). \tag{7}$$

This calculation allows us not only to find the key parameter B, but also to estimate the running time. With B given by (7), we have

$$X^{1/u}u^u = \exp\left((2^{1/2} + o(1))(\log X \log\log X)^{1/2}\right), \tag{8}$$

and so this expression stands as the number of steps to find the requisite number of B-smooth values. Recall that $X = 2n^{1/2+\epsilon}$ in our proto-algorithm, so that the expression in

(8) is of the form $n^{o(1)}$. That is, we do not need to take ϵ as fixed in the expression for X; it may tend to 0 slowly. Letting then $X = n^{1/2+o(1)}$, we get

$$B = \exp\left((1/2 + o(1))(\log n \log\log n)^{1/2}\right), \ X^{1/u}u^u = \exp\left((1 + o(1))(\log n \log\log n)^{1/2}\right),$$

where the second expression is the number of steps to do the sieving to find the requisite number of B-smooth polynomial values.

A general principal, a moral, and three bullets

The above heuristic analysis is instructive in that it can serve, in an almost intact manner, for many factoring algorithms. What may change is the number X, which is the bound on the auxiliary numbers that are examined for smoothness. In the quadratic sieve algorithm, we have X just a little above $n^{1/2}$. In the number field sieve, this bound on auxiliary numbers is much smaller, it is of the form $n^{o(1)}$. Smaller numbers are much more likely to be smooth than larger numbers, and this general principal "explains" the asymptotic superiority of the number field sieve over the quadratic sieve.

Our story has a moral. Smooth numbers are not an artifact, they were forced upon us once we decided to combine auxiliary numbers to make a square. In fact for random numbers this is a theorem of mine: the bound in (8), for $X^{1/u}u^u$ for the depth of the search for random auxiliary numbers below X to form a square, is tight. So the heuristic passage to B-smooth numbers is justified—one is unlikely to be able to assemble a square from random numbers below X in fewer choices than the bound in (8), even if one does not restrict to B-smooth numbers with B the bound in (7).

There are several important points about smooth numbers that make them indispensable in many number-theoretic algorithms:

- Smooth numbers have a simple multiplicative structure.
- Smooth numbers are easy to recognize.
- Smooth numbers are surprisingly numerous.

Gaussian reduction

If Gaussian reduction is used on the final matrix, our complexity bound is ruined. In fact, our matrix will be about $B \times B$, and the bound for the number of steps to reduce such a matrix is about B^3. With B given as in (8), the time for the matrix step is then about $\exp((3/2 + o(1))(\log n \log\log n)^{1/2})$, which then would be the dominant step in the algorithm, and would indicate that perhaps a smaller value of B than in (8) would be in order.

There are several thoughts about the matrix issue. First, Gaussian reduction, though it may be the only trick in the beginning undergraduate linear algebra book, is not the only trick we have. There are in fact fast asymptotic methods, that are practical as well. I refer to the Wiedemann coordinate recurrence method, the Lanczos method, etc. These reduce the complexity to the shape $B^{2+o(1)}$, and so the total asymptotic, heuristic complexity of the quadratic sieve becomes, as first intimated above, $\exp((1 + o(1))(\log n \log\log n)^{1/2})$. Also, Gaussian reduction is not quite as expensive as its complexity estimate indicates. In practice, the matrix starts out as quite sparse, and so for awhile fill-in can be avoided.

And, the arithmetic in the matrix is binary, so a programmer may exploit this, using say a 32-bit word size in the computer, and so process 32 matrix entries at once.

The matrix poses another problem as well, and that is the space that is needed to store it and process it. This space problem is mitigated by using a sparse encoding of the matrix, namely a list of where the 1's are in the matrix. This sparse encoding might be used at the start until the matrix can be cut down to size somewhat.

In practice, people have found that it pays to slightly deoptimize B on the low side. This in essence is a concession to the matrix problem, both to the space required and the time required. While sieving can easily be distributed to many unextraordinary processors, no one knows how to do this efficiently with the matrix, and so this final step might well hog the memory and time of a large expensive computer.

Conclusion

The quadratic sieve is a deterministic factoring algorithm with conjectured complexity $\exp((1+o(1))(\log n \log\log n)^{1/2})$. It is currently the algorithm of choice for "hard" composites with 20 to 120 digits. (By "hard" I mean that the number does not have a small prime factor that could be discovered by trial division or the elliptic curve method, nor does the number succumb easily to other special methods such as the $p-1$ factoring method or the special number field sieve.) For larger numbers, the number field sieve moves to the front, but this "viability border" between the quadratic sieve and the number field sieve is not very well defined, and shifts as new computer architectures come on line and when new variations of the underlying methods are developed.

There are many variations of the quadratic sieve which speed it up considerably (but do not change the asymptotic complexity estimate of $\exp((1+o(1))(\log n \log\log n)^{1/2})$; that is, the variations only affect the "$o(1)$"). The most important of these variations is the idea of using multiple polynomials, due to J. Davis and P. Montgomery.

All of the practical factoring methods beyond trial division are heuristic, though the elliptic curve method is "almost" rigorous. The fastest, rigorous factoring algorithm is a probabilistic method of H. Lenstra and me, with expected complexity $\exp((1+o(1))(\log n \log\log n)^{1/2})$, namely the same as for the quadratic sieve (though here a "fatter" $o(1)$ makes the rigorous method inferior in practice). The fastest, rigorous deterministic factoring algorithm is due to J. Pollard and to V. Strassen, with a complexity of $n^{1/4+o(1)}$.

For further reading and references to original papers and other surveys, see

R. Crandall and C. Pomerance, *Prime numbers: a computational perspective*, Springer-Verlag, New York, 2001.

A. Granville, *Smooth numbers: computational number theory and beyond*, these proceedings.

C. Pomerance, *A tale of two sieves*, Notices Amer. Math. Soc. **43** (1996), 1473–1485.