Factoring Integers – an Introduction

Richard P. Brent
MSI & CECS, ANU

12 August 2010

Motivation – Unique Prime Factorisation

Every natural number (integer) $n > 1$ is a product of prime powers

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k},$$

and this representation is unique except for the ordering, i.e. it is unique if we assume that $p_1 < p_2 < \cdots p_k$ and the exponents α_i are positive.

$$1001 = 7 \cdot 11 \cdot 13$$

$$500 = 2^2 \cdot 5^3$$

$$2^{32} + 1 = 641 \cdot 6700417$$

Given a large integer n, how can we actually find the prime factors of n?

Historical Example

$M_n = 2^n - 1$ is a *Mersenne number*. Mersenne (1644) claimed that

$$M_{67} = 147573952589676412927$$

was prime.

Lucas (1875) showed that M_{67} was composite, but he did not find any factors (we'll see later how to show that a number is composite without factoring it, using Fermat's little theorem).

Cole (1903) showed that

$$M_{67} = 193707721 \cdot 761838257287,$$

but it took him "three years of Sundays".

Nowadays, the Magma package running on my laptop takes about one second to find Cole's factors of M_{67}. The smallest Mersenne number that is not completely factored is now M_{919}.

Algebraic Factors

Sometimes it's easy to find some (usually not all) factors of a number using algebra.

For example, we know that

$$x^3 + 1 = (x + 1)(x^2 - x + 1).$$

Put $x = 2^k$. This gives

$$2^{3k} + 1 = (2^k + 1)(2^{2k} - 2^k + 1).$$

For example, $2^{33} + 1$ has a factor $2^{11} + 1$.

Similarly, $10^{67} - 1$ has a factor $10 - 1 = 9$.

This is obvious if you write $10^{67} - 1 = \underbrace{99\cdots99}_{\text{67 digits}}$.

Aurifeuillian Factors

These are trickier than algebraic factors, and harder to spell!

For example,

$$\begin{aligned} 4x^4 + 1 &= (4x^4 + 4x^2 + 1) - 4x^2 \\ &= (2x^2 + 1)^2 - (2x)^2 \\ &= (2x^2 + 2x + 1)(2x^2 - 2x + 1) \end{aligned}$$

Put $x = 2^k$:

$$2^{4k+2} + 1 = (2^{2k+1} + 2^{k+1} + 1)(2^{2k+1} - 2^{k+1} + 1).$$

For example, $2^{118} + 1 = (2^{59} + 2^{30} + 1)(2^{59} - 2^{30} + 1)$.

Algebraic and Aurifeuillian factors

Aurifeuillian factors are usually different from algebraic factors. For example, $15^{15} + 1$ has algebraic factors $15 + 1$, $15^3 + 1$, $15^5 + 1$, and Aurifeuillian factors 19231, 142111. Combining this information, we find:

$$15^{15} + 1 = 2^4 \cdot 31 \cdot 211 \cdot 1531 \cdot 19231 \cdot 142111.$$

Unfortunately, algebraic and Aurifeuillian factors only apply in very special cases. They don't give a general factoring method.

Modular Arithmetic

Recall that

$$a = b \bmod N$$

means that N divides $(a - b)$ We sometimes write this

$$N \mid (a - b).$$

Exercise. If $p \mid N$, $a = b \bmod N$, and $b = c \bmod p$, then $a = c \bmod p$.
e.g. $17 = 32 \bmod 15$, $32 = 2 \bmod 3$, so $17 = 2 \bmod 3$.

It's not generally true unless $p \mid N$.
e.g. $17 = 32 \bmod 15$, $32 = 0 \bmod 2$, but $17 \neq 0 \bmod 2$.

Fast Algorithms

A polynomial time algorithm is an algorithm (think of a computer program if you prefer) whose running time is at most a polynomial in the length ℓ of the input data.

e.g. $N = 123456789$ is an integer whose length is 9 (in units of decimal digits).

Usually length is measured in units of binary digits (bits), but this does not change the definition of polynomial-time algorithm.

A polynomial is a function like

$$P(x) = ax^3 + \underbrace{bx^2 + cx + d}\,,$$

but we can ignore the low-order terms here.

Generally, "fast" = "polynomial time", but the degree of the polynomial and the size of the constant "a" are important in practice.

Fast Modular Exponentiation

The example “Exponentiating Mod Wise” on page 35 of John Hutchinson’s notes illustrates a “fast” algorithm for computing

$$a^b \bmod N.$$

This is feasible even for numbers with hundreds of digits, because the time is (at most) a cubic polynomial in the input size.

Greatest Common Divisor (GCD)

The *Euclidean Algorithm* is a fast algorithm for computing the greatest common divisor $\gcd(M, N)$ of two integers M and N.

Application. If $p \mid M$ and $p \mid N$, then $p \mid \gcd(M, N)$ and it could happen that $p = \gcd(M, N)$.

This can be useful when trying to factor N.

Testing Primality

Fermat's *little* Theorem "FLT" (Thm. 19, pg. 44):
If p is prime then

$$a^p = a \bmod p.$$

Sometimes the Theorem is stated as

$$a^{p-1} = 1 \bmod p,$$

but then we need the extra condition $a \neq 0 \bmod p$. Suppose we are given an integer $N > 2$. Before trying to factor N, choose some integer a, where $1 < a < N - 1$, and compute

$$b = a^N \bmod N.$$

If $b \neq a \bmod N$, then N must be composite (otherwise get a contradiction to FLT), so it makes sense to try to factor N. If $b = a \bmod N$ then probably N is prime, so we could be wasting our time trying to factor N. Better to check if N really is prime. This can be done "fast" – see "talks" on my web page.

“Divide and Conquer” Factoring Strategy

N is a large integer that we know is composite. We want to find a *nontrivial* factor f of N (nontrivial means $1 < f < N$).

Once f has been found, we can test f and $q = N/f$ to see if they are prime; if so the factorisation of $N = f \cdot q$ is complete.

Otherwise, we have at least reduced the problem to one or two smaller problems (factoring f and/or q).

This is an example of the very useful “divide and conquer” strategy – if you can’t immediately solve a problem, try to reduce it to one or more smaller problems.

Example

Try to factor $N = 1001$.

It's easy to see that 2, 3, 5 are not divisors of N, but $f = 7 \mid N$, and the quotient is $q = N/f = 143$.

f is prime, but q is not.

From $q = 12^2 - 1$ we get $q = 11 \cdot 13$. Thus $N = 7 \cdot 11 \cdot 13$.

Since it is easy to divide out powers of 2, I'll assume from now on that N is *odd*.

Trial division

The simplest way to factor N is to divide it by 2, 3, 4, ... until we find some $p \mid N$. Then p is the smallest factor of N and must be prime (otherwise N would have a smaller factor).

Drawback. The time required is proportional to p. Thus, trial division can be very slow if p is large. Since p is the smallest prime factor of N,

$$p \leq \sqrt{N}.$$

Improvements. We only need to divide by *primes* $2, 3, 5, 7, 11, \ldots$ (but we need a list of these or some way of generating them).

By the *Prime Number Theorem* this saves a factor of order $\log N$, which is not very significant.

Fermat's Method

Trial division is good for finding *small* factors. Fermat (1643) proposed a method that is good for finding factors that are close to $\sqrt{N}$ – the other extreme.

Suppose N is odd and $N = uv$, where $0 < u \le v$.
Let

$$x = \frac{v+u}{2}, \quad y = \frac{v-u}{2}.$$

Note that x and y are integers, since $v \pm u$ is even.

$$x + y = v, \quad x - y = u,$$

$$N = uv = (x-y)(x+y) = x^2 - y^2$$

Fermat tries to find integers x and y satisfying the equation $x^2 - N = y^2$, with y small. Thus, he starts with $x = \lceil \sqrt{N} \rceil$, the smallest integer whose square is $\ge N$.

Fermat by hand (or on a calculator)

Suppose $N = 9401$. We find $96^2 < N < 97^2 = 9409$, so start with $x = 97$ and increase x by 1 until (hopefully) we find a value such that $x^2 - N$ is a perfect square.

It's easy to increase x by 1 and update $x^2 - N$, since $(x+1)^2 - x^2 = 2x + 1$.

x	$2x+1$	$x^2 - N$
97	195	8
98	197	203
99	199	$400 = 20^2$

Thus $N = 99^2 - 20^2 = (99 - 20)(99 + 20) = 79 \cdot 119$.

It would be much more work to find this by trial division!

Problems with Fermat's Method

- The factors u and v of N that are found by Fermat's method might not be prime, so they have to be factored (but this should not be so hard, since they are smaller than N).
- The worst case ($u = 3$, $v = N/3$) is very slow – even slower than trial division.

Trial division (by odd divisors) takes about $u/2$ steps.

Fermat takes about

$$x - \sqrt{N} = \frac{u+v}{2} - \sqrt{N} = \frac{u}{2} + \left(\frac{N}{2u} - \sqrt{N}\right) \text{ steps,}$$

and

$$\frac{N}{2u} > \sqrt{N} \text{ if } u < \frac{\sqrt{N}}{2}.$$

Improving Fermat – the Quadratic Sieve

Fermat's method tries to find x, y such that $x^2 - y^2 = N$, but it would be enough to find x, y such that

$$x^2 - y^2 = 0 \bmod N,$$

i.e.

$$x^2 = y^2 \bmod N,$$

provided $x \neq \pm y \bmod N$.

Then $\gcd(x - y, N)$ will give a factor of N.

The quadratic sieve (QS) method tries to factor $x_i^2 \bmod N$ for several different x_i, and combine the results to get an equation

$$x^2 = y^2 \bmod N.$$

With luck (at least 50% of the time) this gives nontrivial factors of N.

Small Example of the Quadratic Sieve

Consider $N = 1649$, so $40 < \sqrt{N} < 41$.

$$\begin{aligned} 41^2 &= 1681 = 32 = 2^5 \bmod N \qquad (*) \\ 42^2 &= 1764 = 115 = 5 \cdot 23 \bmod N \\ 43^2 &= 1849 = 200 = 2^3 \cdot 5^2 \bmod N \quad (*) \end{aligned}$$

Multiply the two "relations" marked $(*)$, giving

$$(41 \cdot 43)^2 = 2^8 \cdot 5^2 = (2^4 \cdot 5)^2 \bmod N,$$

i.e.

$$x^2 = y^2 \bmod N,$$

where $x = 41 \cdot 43 = 114 \bmod N$, and $y = 2^4 \cdot 5 = 80$.
We were lucky, because $x \neq \pm y \bmod N$.

$$\gcd(x - y, N) = \gcd(114 - 80, N) = 17,$$

and it's easy to check that $17 \mid N$, in fact $N = 17 \cdot 97$. This would not have been so easy to find using Fermat's method.

How lucky were we?

Suppose $N = p \cdot q$ where p, q are distinct primes, and

$$x^2 = y^2 \bmod N.$$

Thus $$N \mid (x-y)(x+y),$$

so $$p \mid (x-y) \ \text{ or } \ p \mid (x+y)$$

and $$q \mid (x-y) \ \text{ or } \ q \mid (x+y).$$

There are 4 cases. If $p \mid (x-y)$ and $q \mid (x-y)$ then $N|(x-y)$, so $x = y \bmod N$ and we don't find a factor of N.
Similarly, if $p \mid (x+y)$ and $q \mid (x+y)$ then $N|(x+y)$, so $x = -y \bmod N$ and we don't find a factor.

However, in the other two cases we do find a factor. For example, if $p \mid (x-y)$ and $q \mid (x+y)$, we get p from $\gcd(N, x-y)$ and q from $\gcd(N, x+y)$.

If the 4 cases are equally likely, we have a 50% chance of success.

A Larger Example

Let's try $N = 1098413$. Compute $x^2 - N$ for $x > \sqrt{N}$ and try to factor $x^2 - N$ using only primes in the *factor base* $S = \{2, 3, 5, 7, 11, 13, 17, 19, 23\}$:

$$\begin{aligned}
1051^2 &= 2^2 \cdot 7 \cdot 13 \cdot 17 \bmod N \\
1063^2 &= 2^2 \cdot 7^3 \cdot 23 \bmod N \quad (*) \\
1077^2 &= 2^2 \cdot 7 \cdot 13^3 \bmod N \quad (*) \\
1119^2 &= 2^2 \cdot 7 \cdot 17^2 \cdot 19 \bmod N \\
1142^2 &= 7^2 \cdot 13 \cdot 17 \cdot 19 \bmod N \\
1237^2 &= 2^2 \cdot 13 \cdot 19^2 \cdot 23 \bmod \quad (*)
\end{aligned}$$

Multiply the relations $(*)$ to get:

$$(1063 \cdot 1077 \cdot 1237)^2 = (2^3 \cdot 7^2 \cdot 13^2 \cdot 19 \cdot 23)^2 \bmod N,$$

i.e. $$326330^2 = 391638^2 \bmod N,$$

and compute $\gcd(N, 391638 - 326330) = 563$, giving $$N = 563 \cdot 1951.$$

How to find the starred relations

If N is large we might get thousands of relations – how do we predict which ones to multiply in order to get a square?

The problem boils down to linear algebra.

Take a matrix with a column for each prime in the factor base, and a row for each relation. Enter 0 if the prime exponent is *even* and 1 if it is *odd*.

Now, we have to find a set of rows whose sum (mod 2) is all zeros.

In other words, find a *linear dependency* between the rows of the matrix, working over the field $F_2 = \{0, 1\}$ (where the operations are addition and multiplication mod 2).

Example

For example, with $N = 1098413$, the matrix we get is:

$$\begin{array}{ccccccccc} 2 & 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \end{array}$$

$$\left[\begin{array}{cccccccccl} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & (*) \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & (*) \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & (*) \end{array}\right]$$

(the numbers in grey are the primes in the factor base)

Adding the rows marked $(*)$, using arithmetic mod 2, we get

$$\left[\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

which means that these rows are linearly dependent over F_2.

Finding a linear dependency takes about the same work as solving a system of linear equations, and is feasible even if the matrix is very large.

Related Factoring Methods

- Instead of just considering $x^2 - N$ we can consider several quadratic polynomials $a_i x^2 + b_i x + c_i$ where $b_i^2 - 4a_i c_i = N$. This gives the *Multiple Polynomial Quadratic Sieve* (MPQS), which is faster than the quadratic sieve if the polynomials are chosen correctly.
- Instead of working over the integers $\bmod N$, we can work over *number fields*. This gives the *Number Field Sieve* (NFS) which is complicated but the best method known for factoring large N.
- QS, MPQS and NFS take a time which depends mainly on the size of N and is more or less independent of the size of the factors of N (unlike trial division and other methods that we'll consider later).

Example – the Ninth Fermat Number

Fermat numbers are numbers of the form $2^{2^n} + 1$.

Fermat thought they were all prime, but Euler found the factorisation:

$$F_5 = 641 \cdot 6700417.$$

F_6, F_7 and F_8 are not too hard to factor, but

$$F_9 = 2424833 \cdot c_{148},$$

where c_{148} is a composite number with 148 decimal digits.

Using the Number Field Sieve, the factors of c_{148} were found:

$$c_{148} = p_{49} \cdot p_{99},$$

where

$p_{49} = 7455602825647884208337395736200454918783366342657$

and p_{99} is a prime with 99 decimal digits – you can find it by division!

Current Record

The largest number factored so far by NFS is RSA768, which is a number with 768 bits (232 decimal digits). It turned out to be a product of two primes, each having 116 decimal digits (though not close enough to be found by Fermat's method).

```
3347807169895689878604416984821269081770479498371376856891
2431388982883793878002287614711652531743087737814467999489
```

and

```
3674604366679959042824463379962795263227915816434308764267
6032283815739666511279233373417143396810270092798736308917
```

It's not yet feasible to factor 1024-bit ($\approx$ 300 digit) numbers, but it might be in a few years' time.

Another Idea – the Pollard “p-1” Method

Suppose $N = p \cdot q$ where p is a prime (not too large); q might be prime or composite.

By Fermat's little theorem,

$$2^{p-1} = 1 \bmod p.$$

Let E be any multiple of $p - 1$. Then

$$2^E = 1 \bmod p,$$

so

$$p \mid (2^E - 1).$$

If we don't know p but can *guess* a suitable E, we can compute

$$\gcd(2^E - 1, N),$$

and (with some luck) this will give us p.

Guessing E

If all the prime power factors of $p-1$ are $\leq B$, take

$$E = \prod_{p_i^{\alpha_i} \leq B} p_i^{\alpha_i}.$$

Because E might be large (roughly e^B), we don't usually compute E explicitly; instead we compute $2^E \bmod N$ using a loop like:

$$a \leftarrow 2; \ \text{for } i = 1, 2, \ldots \text{ do } a \leftarrow a^{p_i^{\alpha_i}} \bmod N.$$

Guessing E continued

In practice we don't know the factors of $p - 1$ (because we don't know p), but we do know that the time for the computation is proportional to B, so we just take a fairly large value, say $B \approx 1000000$, depending on how much computer time we are willing to use.

If we are lucky, and all the prime power factors of $p - 1$ are $\leq B$, then we will find the factor p of N.

Otherwise, we have to increase B and try again, or try another method (e.g. MPQS).

Example

The Pollard $p-1$ method is great if we are lucky enough that $p-1$ has all “small” prime factors.

For example, Nohara found a 66-decimal digit factor p of $N = 960^{119} - 1$.

It turns out that

$p-1 = 2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 17 \cdot 23 \cdot 31 \cdot 163 \cdot 401 \cdot 617 \cdot 4271 \cdot 13681 \cdot 22877 \cdot 43397 \cdot 203459 \cdot 1396027 \cdot 6995393 \cdot 13456591 \cdot 2110402817,$

and 2110402817 is small enough (if you have a fast computer).

However, this situation is unusual. A 66-digit number is extremely unlikely to have all its prime factors so small. (The chance is roughly 1 in a million.)

Worst Case

If $p - 1 = 2q$ where q is a prime, then the Pollard "$p - 1$" method is very slow.

(p, q) is called a "Sophie Germain" pair after Marie-Sophie Germain (1776–1831). There seem to be infinitely many such pairs, e.g. $(5, 2), (7, 3), (11, 5), (23, 11)$, but no one has proved this.

The problem is similar to the problem of *twin primes*, that is pairs $(p, p + 2)$ where p and $p + 2$ are both prime.

The Elliptic Curve Method (ECM)

- The set $G = \{1, 2, \ldots, p-1\}$ forms a *group* of order $p-1$ with the operation "multiplication mod p" if p is a prime.
- The Pollard $p-1$ method works well if the group order is "smooth" – meaning that all its prime factors are small.
- In Lenstra's Elliptic Curve Method (ECM), we can choose different groups with orders close to (but not usually equal to) p, until we are lucky and find one whose order is sufficiently smooth.
- By a result of Hasse, the group orders are in the interval $(p+1-2\sqrt{p}, p+1+2\sqrt{p})$.
- ECM is the best method for finding "small" factors p of large numbers N, say factors $p < N^{1/3}$.
- The running time of ECM depends mainly on the size of p, and only weakly on the size of N.

ECM Examples

I factored the 10-th and 11-th Fermat numbers using ECM.

For example,

$$F_{10} = 2^{1024} + 1 = p_8 \cdot p_{10} \cdot p_{40} \cdot p_{252},$$

$$\begin{aligned} p_8 &= 45592577 \\ p_{10} &= 6487031809 \\ p_{40} &= 4659775785220018543264560743076778192897 \\ p_{252} &= 130439874405 \cdots 127014424577 \end{aligned}$$

p_8 and p_{10} are “easy”.
p_{40} was found by ECM, and would have been very hard to find by any other method.
p_{252} can be found by division once the other factors are known (of course, we have to check that it is prime).

ECM Examples continued

$$F_{11} = 2^{2048} + 1 = p_6 \cdot p'_6 \cdot p_{21} \cdot p_{22} \cdot p_{564},$$

$$\begin{aligned} p_6 &= 319489 \\ p'_6 &= 974849 \\ p_{21} &= 167988556341760475137 \\ p_{22} &= 3560841906445833920513 \\ p_{564} &= 1734624471 \cdots 6598834177 \end{aligned}$$

The 21-digit and 22-digit factors were found by ECM; then it is easy to find the 564-digit factor p_{564} (though proving that it is prime is not so easy).

ECM Record

The largest factor found by ECM is a 73-digit factor

$$p_{73} = 1808422353177349564546512035512530001279481259854248860454348989451026887$$

of

$$2^{1181} - 1$$

(found by Bos, Kleinjung, Lenstra and Montgomery on 7 March 2010, using a cluster of PlayStation 3 game consoles).

The largest prime factor of the group order is 10801302048203.

Summary

We've looked at several methods for factoring integers:

- Trial division (simple but slow).
- Fermat's method (also simple, but slow in most cases).
- Quadratic sieve (QS) and MPQS.
- Number field sieve (NFS) – the best general-purpose method.
- Pollard $p - 1$ (fast if you are lucky).
- Elliptic curve method (ECM) – the best method for finding "small" factors.

A good strategy for factoring is:

- Check if the number N to be factored is a prime power!
- If not, try to find factor(s) by ECM and divide them out.
- If what remains is not a prime power, try MPQS or NFS.

References

1. Richard Brent, *Factor Tables*,
`http://wwwmaths.anu.edu.au/~brent/factors.html`
(has links to other websites of interest).

2. Richard Crandall and Carl Pomerance, *Prime Numbers: A Computational Perspective*, second edition, Springer-Verlag, New York, 2005.

3. G. H. Hardy, E. M. Wright, A. Wiles et al, *An Introduction to the Theory of Numbers*, sixth edition, Oxford University Press, 2008.

4. Hans Riesel, *Prime Numbers and Computer Methods for Factorization*, second edition, Birkhäuser Boston, 1994.

5. Sam Wagstaff, *The Cunningham Project*,
`http://homes.cerias.purdue.edu/~ssw/cun/`.