Smooth numbers

An understanding of "smooth numbers" is essential for the analysis of factor base methods. If y is a positive real number we say that $n \in \mathbb{N}$ is y-smooth, if all prime factors of n have size at most y. Thus 105 is 8-smooth, but not 6-smooth.

Definition $\psi(x, y)$ is the number of y-smooth integers $n \leq x$.
For example, the 3-smooth integers up to 20 are 1,2,3,4,6,8,9,12,16,18, so that $\psi(20, 3) = 10$.

We will only be interested in the situation in which $2 \leq y < x$. The ratio $\log x / \log y$ plays a crucial role in understanding $\psi(x, y)$ and, following standard practice, we define

$$u = \frac{\log x}{\log y}$$

and introduce the **Dickman function** $\rho(u)$.

Facts (which we will use without proof)
(i) There is a function $\rho(u)$ such

$$\psi(x, y) = \rho(u)x + O(x/\log y) \quad (x \geq y \geq 2).$$

The "main term" $\rho(u)x$ may be smaller than the "error term" $O(x/\log y)$ unless u is fairly small. However one can prove formulae with better error terms. The function $\rho(u)$ satisfies $\rho(u) = 1$ for $0 \leq u \leq 1$ and otherwise may be defined by the recursive relation

$$\rho(u) = \rho(k) - \int_k^u \frac{\rho(t-1)}{t} dt,$$

where $k = \lfloor u \rfloor$. In particular we have $\rho(u) = 1 - \log u$ when $1 \leq u \leq 2$.

(ii) We have $\rho(u) \leq (k!)^{-1}$, where $k = \lfloor u \rfloor$. A good approximation (we will not make this precise) is to take $\rho(u)$ "=" u^{-u} for large u.

We will verify statement (i) in the case $1 \leq u \leq 2$.

Lemma (Buchstab's formula)
For $x \geq 1$ and $0 < y \leq z$ we have

$$\psi(x,y) = \psi(x,z) - \sum_{y<p\le z} \psi(x/p,p).$$

(In such sums p always runs over primes.)

Proof For any $w \le x$, each w-smooth integer $n > 1$ can be written uniquely as $n = mp$ where $p \le w$ and m is p-smooth, by taking p as the largest prime factor of n. Since 1 is w-smooth we have

$$\psi(x,w) = 1 + \sum_{p\le w} \psi(x/p,p).$$

The lemma follows on comparing the cases $w = y$ and $w = z$.

Proposition
We have

$$\psi(x,y) = (1 - \log u)x + O(x/\log x)$$

for $1 \le u \le 2$.

Proof We have $\psi(w,z) = \lfloor w \rfloor$ when $z \ge w$. Applying Buchstab with $z = x$ and $\sqrt{x} \le y \le x$ gives

$$\begin{aligned}
\psi(x,y) &= \psi(x,x) - \sum_{y<p\le x} \psi(x/p,p) \\
&= \lfloor x \rfloor - \sum_{y<p\le x} \lfloor x/p \rfloor \ \text{(since } p > x/p \text{ for such } p\text{)} \\
&= x\left(1 - \sum_{y<p\le x} 1/p\right) + O\left(\sum_{p\le x} 1\right) \\
&= x(1 - \log\log x + \log\log y) + O(x/\log x) \\
&= x(1 - \log u) + O(x/\log x).
\end{aligned}$$

In the penultimate step we used the facts that

$$\sum_{p\le t} 1/p = \log\log t + C + O(1/\log t)$$

for a certain numerical constant C (analogous to the Euler constant) and that the number of primes up to x is $O(x/\log x)$.

Smooth square factoring

We are now ready to describe a simple factor base factoring method. It is very far from being the best such method.

Step 1 Pick a smoothness bound b and let $B = \{p \leq b : p \text{ prime}\}$.

Step 2 Pick elements $x \in \mathbb{Z}/n\mathbb{Z}$ at random, and see if the least residue of $x^2 (\text{mod } n)$ is b-smooth, using trial division. If it is, record its prime factorization. Repeat until more than $\#B$ such values have been found.

Step 3 With the equations

$$x_i^2 = \prod_{p \in B} p^{e_{p,i}} (\text{mod } n)$$

use Gaussian elimination over $\mathbb{F}_2$ to find a set of indices I such that

$$\sum_{i \in I} e_{p,i} = 0 (\text{mod } 2)$$

for each $p \in B$, whence

$$\prod_{i \in I} \prod_{p \in B} p^{e_{p,i}}$$

is a square, y^2 say. Then $x^2 = y^2 (\text{mod } n)$, where

$$x = \prod_{i \in I} x_i.$$

Check whether $\gcd(x - y, n)$ is a proper factor of n, and if not try again!

What size b should one use? We expect to need $O(\#B) = O(b/\log b)$ good values of x_i, and the number of attempts before getting a b-smooth value by chance will be $O(1/\rho(u)) = O(u^u)$, say. Thus Step 2 can be expected to take something like $O(bu^u)$ steps, where $u = \log n/\log b$. Taking $b = \exp(\sqrt{(\log n)(\log\log n)/2})$ minimizes this, very roughly. Then, after allowing for Step 3, which will need something like $O((\#B)^3)$ steps, we see that the overall running time might be about

$$\exp(c\sqrt{(\log n)(\log\log n)}) \quad (*)$$

for some constant c ($= 3/\sqrt{2}$ in this calculation).

It is important to note that this grows more slowly than any fixed power n^ε.

One obvious way to improve matters is to pick values of x for which x^2 has only a small residue modulo n, which will increase its chance of being b-smooth. All such variants end up with an expected running time of the shape (*), but with various values for the constant c.

Pollard's $p-1$ method

We aim to factor an odd composite n. Let p be the least prime dividing n. Pollard's $p-1$ method (1974) is likely to succeed if $p-1$ happens to be b-smooth for some relatively small b.

Step 1 Pick a smoothness bound b, in the hope that $p-1$ will be b-smooth.

Step 2 For each prime $q \le b$ define $r(q)$ such that $q^{r(q)} \le \sqrt{n} < q^{r(q)+1}$, and set

$$k = \prod_{q \le b} q^{r(q)}.$$

Hence if $p-1$ is b-smooth we will have $p-1|k$.

Step 3 Pick an integer a and check that it is coprime to n. If it is not coprime we find a factor of n. Compute $a^k \pmod n$ by the repeated squaring technique, and find $\gcd(a^k-1, n)$. If $p-1$ is b-smooth then $p-1|k$, whence $a^k = 1 \pmod p$, and hence $p \mid \gcd(a^k-1, n)$.

Step 4 If $\gcd(a^k-1, n)$ provides a non-trivial factor of n then stop. If the gcd is n then we can try $a^{k'}$ for various divisors k' of k. If the gcd is 1 we can try increasing b.

How long does the algorithm take? Computing $a^k \pmod n$ requires $O(\log k)$ multiplications modulo n, with

$$\log k = \sum_{q \le b} r(q) \log q \le \pi(b) \log \sqrt{n} = O(b \log n).$$

This step dominates the running time, which is therefore $O(bn^{\varepsilon})$. And what is the chance of success? From Fact (ii) in subsection 3.9 this is about u^{-u} for large u. So if we tried $b = n^{1/4}$ then, in the worst case with p around $\sqrt{n}$, the probability is $1 - \log 2 = 0.31\ldots$. Thus we get an algorithm taking time $O(n^{1/4+\varepsilon})$, which succeeds in about three-tenths of cases.

The elliptic curve method

The problem with Pollard's $p-1$ method is that it can only work when $p-1$ is smooth. In Lenstra's Elliptic Curve Method (1987) we work in the group of points on a curve modulo p, and follow an analogous procedure. This time we are likely to succeed if the group order is smooth. But the difference is that now, if we are unsuccessful, we can try other curves until we hit on one whose order is smooth.

Step 1 Pick a smoothness bound b. The hope will be that we can find an elliptic curve such that the number of points over $\mathbb{Z}/p\mathbb{Z}$ (for some prime factor p of n) is b-smooth.
Step 2 Let $k = \prod_{q \le b,} q^{r(q)}$, where q runs over primes and

$$q^{r(q)} \le n^{1/2} + 2n^{1/4} + 1 < q^{r(q)+1}.$$

If $p \le \sqrt{n}$ and N_p is the number of points on an elliptic curve mod p, then Hasse's bound implies that $N_p \le n^{1/2} + 2n^{1/4} + 1$. So we will have $N_p \mid k$ if N_p happens to be b-smooth.
Step 3 Pick an integer pair (x_0, y_0) modulo n at random, and take E as the curve $y^2 = x^3 + x + y_0^2 - x_0^3 - x_0$. Then $P = (x_0, y_0)$ lies on E. Compute kP by the repeated squaring method, working modulo n. If $kP = \mathbf{0}$ modulo p for some $p \mid n$, then this process will produce a denominator which cannot be inverted modulo p, since the point at infinity is, in effect, the point $(0/0, 1/0)$. Thus, when we remove denominators in the calculation modulo n, by finding their inverses via the Euclidean algorithm, we will encounter a gcd which is greater than 1. This will produce a divisor of n.

How to choose b

Given b, the time taken to test each curve is $O(b(\log n)^c)$ for some constant c, just as for the $p-1$ method. The probability of success is about u^{-u}, where $u = (\log p)/(\log b)$. This assumes that all numbers of points are equally likely. (But they are not.) Heuristically the running time is then expected to be $O(u^u b(\log n)^c)$ steps. This is minimized by taking b to be about $\exp(\sqrt{(\log p)(\log\log p)/2})$, giving a running time

$$O(\exp(\sqrt{(1+\varepsilon)(\log n)(\log\log n)}).$$

Of course, we cannot choose b to depend on p, since p is unknown. Thus in practice we might assume p to be of size $\sqrt{n}$. Alternatively we can try an increasing succession of values of b, so that eventually we will reach a value close to $\exp(\sqrt{(\log p)(\log\log p)/2})$.

Variants of this method were used successfully to factor the Fermat numbers F_{10} and F_{11}.

The Elliptic Curve Method is one of most widely used techniques for "everyday" factoring. It is ultimately slower than the Number Field Sieve, for example. However it has the attractive feature that the time taken is less when n has a smaller prime factor, just as with the Pollard ρ-method. This makes it particularly good for "naturally occurring" factorization problems, such as the Fermat numbers mentioned above.