Math 445 Number Theory

September 10-12, 2008

If the Miller-Rabin test tells us that a number N is composite, how do we find its factors? The most straightforward approach; test divide all numbers less than $\sqrt{N}$, or better, all *primes* less than $\sqrt{N}$; eventually you will find a factor. But this requires on the order of $\sqrt{N}$ steps, which is far too large.

A different method uses the fact that if $N = ab$ and $a_1, \ldots a_n$ are chosen at random, a is more likely to divide one of the a_i (or rather (for later efficiency), one of the differences $a_i - a_j$), than N is. This can be tested for by computing gcd's, $d = (a_i - a_j, N)$; this number is $1 < d < N$ if a (or some other factor) divides $a_i - a_j$ but N does not, and finds us a proper factor, d, of N. The probability that a divides none of the differences is approximately $1 - 1/a$ for each difference, and so is approximately

$$(1-\frac{1}{a})^{\binom{n}{2}} = ((1-\frac{1}{a})^a)^{\frac{n(n-1)}{2a}} \approx ((1-\frac{1}{a})^a)^{\frac{n^2}{2a}} \approx ((1-\frac{1}{a})^a)^{\frac{n^2}{2a}} \approx (e^{-1})^{\frac{n^2}{2a}} = e^{\frac{-n^2}{2a}}$$

which is small when $n^2 \approx a \leq \sqrt{N}$, i.e., $n \approx N^{1/4}$. The problem with this method, however, is that it requires $n(n-1)/2 \approx \sqrt{N}$ calculations, and so is no better than trial division! We will rectify this by choosing the a_i *pseudo-randomly* (which will also explain the use of differences). This will lead us to the Pollard ρ method for factoring.

If $N = ab$ and $a_1, \ldots a_n$ are chosen at random, a is more likely to divide one of the differences $a_i - a_j$ than N is. This can be tested for by computing gcd's, $d = (a_i - a_j, N)$; this number is $1 < d < N$ if a (or some other factor) divides $a_i - a_j$ but N does not, and finds us a proper factor, d, of N. The problem with this method, however, is that it requires $n(n-1)/2$ gcd computations, which is too large. This can be remedied by generating the a_i *pseudo-randomly.*

The idea: choose a relatively simple to compute function, like $f(x) = x^2 + c$. Starting from some number a_1, we generate a sequence by repeatedly applying f to a_1 ;

$$a_2 = f(a_1), a_3 = f(a_2) = f^2(a_1), \ldots , a_k = f(a_{k-1}) = f^{k-1}(a_1), \ldots$$

The point is that if ever we have $a | a_i - a_j$, then since

$$a_{i+1} - a_{j+1} = (a_i^2 + c) - (a_j^2 + c) = a_i^2 - a_j^2 = (a_i - a_j)(a_i + a_j)$$

we have $a | a_{i+1} - a_{j+1}$, as well. So (by induction!) $a | a_{i+k} - a_{j+k}$ for all $k \geq 0$. So we can test for occurances of $1 < (a_i - a_j, N) < N$ by testing only a relatively few pairs; we get the effect of testing many more of them for free. In particular, we test $(a_{2i} - a_i, N)$ for each i. This is effective, since if $1 < (a_j - a_i, N) < N$ for $j > 2i$, then $1 < (a_{2j-2i} - a_{j-i}, N)$ as well. So testing $a_{2i} - a_i$ will essentially test these other pairs at the same time. Turning this into an algorithm:

Given N composite, choose a function $f(x) = x^2 + c$ and a starting point a_1; set $b_1 = f(a_1)$ and then build the sequences $a_i = f(a_{i-1})$ and $b_i = f^2(b_{i-1})$. Compute $(b_i - a_i, N)$ and

if for some i, $1 < (b_i - a_i, N) < N$, stop: we have found a factor.

if $(b_i - a_i, N) = N$ or i gets too large, reset the parameters: use a new a_1 or a new c.

We expect in the generic case for this process to find a factor by the time i gets in the range of $N^{1/4}$ (or rather, the square root of the smallest prime factor of N), but this is not guaranteed.