The Pollard rho method.

Prof. Andrew Booker gives the following version of the Pollard rho method in his Notes for the Number Theory course of 2016 at the University of Bristol.

Outline. We note firstly that, given composite integer N, the smallest prime factor p of N satisfies $p \leq \sqrt{N}$. Then if we take a random (or in practice a *quasi-random*) sequence of integers $(m_0, m_1, \ldots, m_{k-1})$ with integer $k \simeq 10 \cdot N^{\frac{1}{4}}$ $(\geq 10 \cdot \sqrt{p})$ we find, with high probability, that the m_i are pairwise distinct modulo N. Also—and this is the vital part—we are almost certain to find indices $0 \leq i < j \leq k$ such that $m_i \equiv m_j$ modulo p. This means that $1 < \gcd(m_i - m_j, N) < N$ and so we have found a non trivial factor of N, namely $\gcd(m_i - m_j, N)$. Now, by carefully choosing the way in which we compute a pseudo-random sequence $(m_0, m_1, m_2, \ldots)$ we are able to apply this method to factorise N in time $O\Big(N^{\frac{1}{4}}\Big)$.

The pseudo-random sequence. The idea is to choose a pseudo-random sequence $(m_0, m_1, m_2, \ldots)$ to make your algorithm time and space efficient. It turns out that, for any given integer $c \neq 0, -2$, the sequence defined by taking some initial *good*[1] seed m_0, and letting $m_{i+1} = m_i^2 + c \pmod{N}$ for $i \geq 0$, is pseudo-random. Now, in the Pollard rho method we are searching for $i \neq j$ such that $1 < \gcd(m_i - m_j, N) < N$. Note that our implementation of this will be more time and space efficient if we can refine our search in some way (instead of doing a brute force search of all the indices $i < j$) and optimise the complexity of computation at each step. However, using a polynomial pseudo-random sequence as just described, we can achieve this as follows.

Note firstly that if $\gcd(m_i - m_j, N) = d$ then $d \mid m_i - m_j$, i.e. $m_i \equiv m_j \pmod{d}$. But, in this case, $m_{i+1} \equiv m_i^2 + c \equiv m_j^2 + c = m_{j+1} \pmod{d}$, and so the sequence $(m_0, m_1, m_2, \ldots)$ is eventually periodic—with period ℓ say—modulo d. Note that the period ℓ must of course divide $j - i$. (Make sure that you understand why.) Now let $r := j - i$. Then, for any $s, t \geq i$, $m_s \equiv m_t \pmod{d}$ whenever $s \equiv t \pmod{r}$.

Now let s be the least multiple of r exceeding $i - 1$. (Thus $s := kr$ for some integer $k \geq 1$ and $s \geq i$.) Also let $t := 2s$. Then $m_s \equiv m_{2s} \pmod{d}$. Consequently, amongst the numbers $m_{2s} - m_s$, we expect to find at least one such that $1 < \gcd(m_{2s} - m_s, N) < N$ with $0 \leq s \leq 10 \cdot N^{\frac{1}{4}}$. (Indeed, as explained in section 5, we expect all (m_i, m_j) such that $0 \leq i < j \leq 10 \cdot N^{\frac{1}{4}}$ to be distinct modulo N, and we expect to find one such pair, (m_i, m_j) such that $d := \gcd(m_j - m_i, N)$ satisfies $1 < d < N$. Now let $r := j - i$ and let k be least such that $k \cdot r \geq i$. Then, as noted above, we have that $m_s \equiv m_{2s} \pmod{d}$ for all $s \geq k \cdot r$. So now choose the least k' such that $i + k' \cdot r \geq kr$ and let $s' = i + k' \cdot r$. Then $m_{s'} \equiv m_{2s'} \equiv m_i \equiv m_j \pmod{d}$ so that $1 < d \leq \gcd(m_{2s'} - m_{s'}, N)$. Also, it is again almost certainly the case that $m_{s'}$ and $m_{2s'}$ are distinct modulo N, so that we do also expect that $\gcd(m_{2s'} - m_{s'}, N) < N$.)

Note. This means that we only need to consider/test pairs of the form (m_s, m_{2s}) with[2] $1 \leq s \leq 10 \cdot N^{\frac{1}{4}}$ instead of all pairs (m_i, m_j) such that $0 \leq i < j \leq 10 \cdot N^{\frac{1}{4}}$. However we can efficiently compute the pairs (m_s, m_{2s}) for successive values of s. Indeed, $(m_{s+1}, m_{2(s+1)}) \equiv (m_s^2 + c, (m_{2s}^2 + c)^2 + c) \pmod{N}$. Thus the expected running time to find a factor—hence to factor N when it is the product of two primes—is $O(N^{\frac{1}{4}})$. Also note that you can clearly

[1] You will need to determine what *good* means here—either from the literature or via testing/sampling. I suspect that a good seed m_0 is (for example) a prime such that, for only a small number of the initial iterates m_i (i.e. for $i = 0, 1, 2, \ldots$), is m_i actually smaller than N so that the iterates very soon start bouncing around modulo N in a pseudo-random manner.

[2] For $s = 0$, $(m_s, m_{2s}) = (m_0, m_0)$. I.e. m_s and m_{2s} are not distinct in this case. Hence we do not consider $s = 0$.

implement this method in a very space efficient manner (as you do not have to store in memory a long list of iterates).

Example. Consider the integer $N = 78667$. We make use of the pseudo-random sequence defined by $m_0 = 3$, $m_{i+1} = m_i^2 - 1 \pmod{N}$ to obtain a factorisation of N. The sequence is

$$(3, 8, 63, 3968, 11623, 22889, 62767, 52928, 41313, 4736, 9600, \dots)$$

so our algorithm computes, $\gcd(m_2 - m_1, N) = \gcd(63 - 8, 78667) = 1$, and continuing, that $\gcd(m_{2s} - m_{1s}, N) = 1$ for $s = 2, 3, 4$. However,

$$\gcd(m_{10} - m_5, N) = \gcd(9600 - 22889, 78667) = 97$$

and so, as $78667/97 = 811$ we have found the factorisation $78667 = 97 \cdot 811$.

Note. Working modulo $d = 97$ we have $m_0 \equiv 3$, $m_1 \equiv 8$, $\dots$, $m_4 \equiv 80$, $m_5 \equiv 94 \equiv m_{10}$, etc. Since the iteration becomes cyclic at at this point, the iterates can be represented in the ρ shaped diagram below. (Hence the origin of the name Pollard rho.)

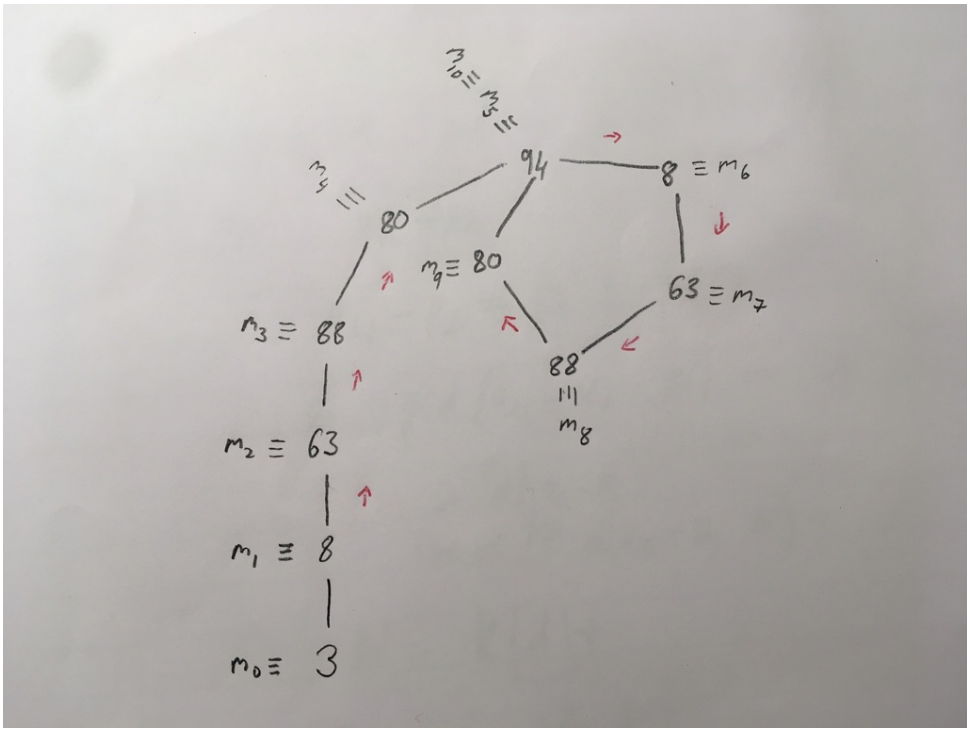

Figure 1: The iterates m_i modulo 97.