MPRI – Cours 2-12-2

logoX F. Morain logoCNRS logoINRIA

Lecture IV: Integer factorization

2013/10/14

I. Introduction.

II. Smoothness testing.

III. Pollard's RHO method.

IV. Pollard's $p-1$ method.

I. Introduction

Input: an integer N;

Output: $N = \prod_{i=1}^{k} p_i^{\alpha_i}$ with p_i (proven) prime.

Major impact: estimate the security of RSA cryptosystems.

Also: primitive for a lot of number theory problems.

How do we test and compare algorithms?

- Cunningham project,
- RSA Security (partitions, RSA keys) – though abandoned?
- Decimals of π.

What is the factorization of a random number?

$N = N_1 N_2 \cdots N_r$ with N_i prime, $N_i \geq N_{i+1}$.

Prop. $r \leq \log_2 N$; $\overline{r} = \log\log N$.

Size of the factors: $D_k = \lim_{N\rightarrow+\infty} \log N_k / \log N$ exists and

k	D_k
1	0.62433
2	0.20958
3	0.08832

"On average"

$$N_1 \approx N^{0.62}, \quad N_2 \approx N^{0.21}, \quad N_3 \approx N^{0.09}.$$

$\Rightarrow$ an integer has one "large" factor, a medium size one and a bunch of small ones.

II. Smoothness testing

Def. a B-smooth number has all its prime factors $\leq B$.

B-smooth numbers are the heart of all efficient factorization or discrete logarithm algorithms.

De Bruijn's function: $\psi(x, y) = \#\{z \leq x, z \text{ is } y-\text{smooth}\}$.

Thm. (Candfield, Erdős, Pomerance) $\forall\, \varepsilon > 0$, uniformly in $y \geq (\log x)^{1+\varepsilon}$, as $x \rightarrow \infty$

$$\psi(x, y) = \frac{x}{u^{u(1+o(1))}}$$

with $u = \log x / \log y$.

B-smooth numbers (cont'd)

Prop. Let $L(x) = \exp\left(\sqrt{\log x \log\log x}\right)$. For all real $\alpha > 0$, $\beta > 0$, as $x \rightarrow \infty$

$$\psi(x^\alpha, L(x)^\beta) = \frac{x^\alpha}{L(x)^{\frac{\alpha}{2\beta}+o(1)}}.$$

Ordinary interpretation:

a number $\leq x^\alpha$ is $L(x)^\beta$-smooth with probability

$$\frac{\psi(x^\alpha, L(x)^\beta)}{x^\alpha} = L(x)^{-\frac{\alpha}{2\beta}+o(1)}.$$

A) Trial division

Algorithm: divide $x \leq X$ by all $p \leq B$, say $\{p_1, p_2, \ldots, p_m\}$, $m = \pi(B) = O(B)$.

Cost: m divisions or $\sum_{p \leq B} T(x, p) = O(m \lg X \lg B)$.

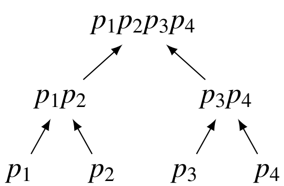
Implementation: use any method to compute and store all primes $\leq 2^{32}$ (one char per $(p_{i+1} - p_i)/2$; see Brent).

Useful generalization: given $x_1, x_2, \ldots, x_n \leq X$, can we find the B-smooth part of the x_i's more rapidly than repeating the above in $O(nm \lg B \lg X)$?

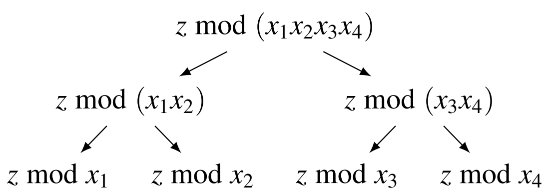
B) Product trees

Algorithm: Franke/Kleinjung/FM/Wirth improved by Bernstein

1. [Product tree] Compute $z = p_1 \cdots p_m$.



2. [Remainder tree] Compute $z \bmod x_1$, …, $z \bmod x_n$.



3. [explode valuation] For each $k \in \{1, \ldots, n\}$, compute $y_k = z^{2^e} \bmod x_k$ with e s.t. $2^{2^e} \geq x_k$; print $\gcd(x_k, y_k)$.

Validity and analysis

Validity: let $y_k = z^{2^e} \bmod x_k$. Suppose $p \mid x_k$. Then $\nu_p(x_k) \leq 2^e$, since $2^\nu \leq p^\nu \leq 2^{2^e}$. Therefore $\nu_p(y_k) \geq 2^e \geq \nu$ and the gcd will contain the right valuation.

Division: If A has $r+s$ digits and B has s digits, then plain division requires $D(r+s, s) = O(rs)$ word operations.
In case $r \gg s$, break into r/s divisions of complexity $M(s)$.

Step 1: $M((m/2) \lg B)$.

Step 2: $D(m \lg B, n \lg X) \approx (m/n)M(n) \lg B \lg X$.

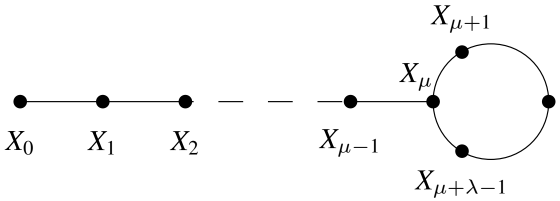
Ex. $B = 2^{32}$, $m \approx 1.9 \cdot 10^8$, $X = 2^{64}$.

Rem. If space is an issue, do this by blocks.

Rem. For more information see Bernstein's web page.

III. Pollard's RHO method

Prop. Let $f : E \rightarrow E$, $\#E = m$; $X_{n+1} = f(X_n)$ with $X_0 \in E$.



Thm. (Flajolet, Odlyzko, 1990) When $m \rightarrow \infty$

$$\overline{\lambda} \sim \overline{\mu} \sim \sqrt{\frac{\pi m}{8}} \approx 0.627\sqrt{m}.$$

Epact

Prop. There exists a unique $e > 0$ (epact) s.t. $\mu \leq e < \lambda + \mu$ and $X_{2e} = X_e$. It is the smallest non-zero multiple of λ that is $\geq \mu$: if $\mu = 0$, $e = \lambda$ and if $\mu > 0$, $e = \lceil \frac{\mu}{\lambda} \rceil \lambda$.

Floyd's algorithm:

```
X <- X0; Y <- X0; e <- 0;
repeat
    X <- f(X); Y <- f(f(Y)); e <- e+1;
until X = Y;
```

Thm. $\overline{e} \sim \sqrt{\frac{\pi^5 m}{288}} \approx 1.03\sqrt{m}$.

Application to the factorization of N

Idea: suppose $p \mid N$ and we have a random $f \bmod N$ s.t. $f \bmod p$ is "random".

---

function `f(x, N)` *return* $(x^2 + 1)$ `mod N;` *end*.
function `rho(N)`
`1. [initialization] x:=1; y:=1; g:=1;`
`2. [loop]`
    while `(g = 1)` *do*
        `x:=f(x, N); y:=f(f(y, N), N);`
        `g:=gcd(x-y, N);`
    endwhile;
`3.` *return* `g;`

Conjecture. RHO finds $p \mid N$ using $O(\sqrt{p})$ iterations.

Practice

- **Choosing f:**
  - some choices are bad, as $x \mapsto x^2$ et $x \mapsto x^2 - 2$.
  - Tables exist for given f's.
- **Trick:** compute $\gcd(\prod_i (x_{2i} - x_i), N)$, using backtrack whenever needed.
- **Improvements:** reducing the number of evaluations of f, the number of comparisons (see Brent, Montgomery).

Theoretical results (1/4)

Thm. (Bach, 1991) Proba RHO with $f(x) = x^2 + 1$ finding $p \mid N$ after k iterations is at least

$$\frac{\binom{k}{2}}{p} + O(p^{-3/2})$$

when p goes to infinity.

Sketch of the proof: define

$$f_0(X, Y) = X, f_{i+1}(X, Y) = f_i^2 + Y,$$

$$f_1(X, Y) = X^2 + Y, f_2(X, Y) = (X^2 + Y^2) + Y, \ldots$$

Divisor of N found by inspecting $\gcd(f_{2i+1}(x, y) - f_i(x, y), N)$ for $i = 0, 1, 2, \ldots$.

Bach (2/4)

Prop. (a) $\deg_X(f_i) = 2^i$;
(b) $f_i \equiv X^{2^i} \bmod Y$;
(c) $f_i \equiv Y^{2^{i-1}} + \cdots + Y \bmod X$;
(d) $f_{i+j}(X, Y) = f_i(f_j(X, Y), Y)$.
(e) f_i is absolutely irreducible (Eisentein's criterion).
(f) $f_j - f_i = f_{j-1}^2 - f_{i-1}^2 = (f_{j-1} + f_{i-1})(f_{j-2} + f_{i-2}) \cdots (f_{j-i} + X)(f_{j-i} - X)$.
(g) For all $k \geq 1$

$$f_{\ell+n} \pm f_\ell \mid f_{\ell+kn} \pm f_\ell.$$

Bach (3/4)

For $i < j$, $\rho_{i,j}$ is the unique poly in $\mathbb{Z}[X, Y]$ s.t.

(a) $\rho_{i,j}$ is a monic (in Y) irreducible divisor of $f_j - f_i$.

(b) Let $\omega_{i,j}$ denote a primitive $(2^j - 2^i)$ root of unity. Then $\rho_{i,j}(\omega_{i,j}, 0) = 0$.

Proof:

$$f_j - f_i \equiv X^{2^i}(X^{2^j - 2^i} - 1) \bmod Y,$$

and

$$X^{2^j - 2^i} - 1 = \prod_{\mu \mid 2^j - 2^i} \Phi_\mu(X).$$

$$\rho_{0,j} \left| \frac{f_j - f_0}{\prod_{d \mid j, d \neq j} \rho_{0,d}} \right.,$$

and for $i \geq 1$:

$$\rho_{i,j} \left| \frac{f_{j-1} + f_{i-1}}{\prod_{d \mid j-i, d \neq j-i} \rho_{i,i+d}} \right..$$

Conj. we have in fact equalities instead of divisibility.

Bach (4/4)

Thm1. $f_k - f_\ell$ factor over $\mathbb{Z}[X, Y]$. Moreover, they are squarefree. Proof uses projectivization of f.

Weil's thm: if $f \in \mathbb{Z}[X, Y]$ is absolutely irreducible of degree d, then N_p the number of projective zeroes of f is s.t.

$$|N_p - (p + 1)| \leq 2\binom{d-1}{2}\sqrt{p}.$$

Thm2. Fix $k \geq 1$. Choose x and y at random s.t. $0 \leq x, y < p$. Then, proba for some $i, j < k$, $i \neq j$, $f_i(x, y) = f_j(x, y) \bmod p$ is at least $\binom{k}{2}/p + O(1/p^{3/2})$ as p tends to infinity.

Proof: same as $i < j < k$ and $\rho_{i,j}(x, y) \equiv 0 \bmod p$. Use inclusion-exclusion, Weil's inequality and Bézout's theorems. □

Same result for RHO to find $p \mid N$.

IV. Pollard's $p-1$ method

- Invented by Pollard in 1974.
- Williams: $p+1$.
- Bach and Shallit: Φ_k factoring methods.
- Shanks, Schnorr, Lenstra, etc.: quadratic forms.
- Lenstra (1985): ECM.

Rem. Almost all the ideas invented for the classical $p-1$ can be transposed to the other methods.

First phase

Idea: assume $p \mid N$ and a is prime to p. Then

$$(p \mid a^{p-1} - 1 \text{ and } p \mid N) \Rightarrow p \mid \gcd(a^{p-1} - 1, N).$$

Generalization: if R is known s.t. $p-1 \mid R$,

$$\gcd((a^R \bmod N) - 1, N)$$

will yield a factor.

How do we find R**?** Only reasonable hope is that $p-1 \mid B!$ for some (small) B. In other words, $p-1$ is B-smooth.

Algorithm: $R = \prod_{p^\alpha \le B_1} p^\alpha = \mathrm{lcm}(2, \ldots, B_1)$.

Rem. (usual trick) we compute $\gcd(\prod_k((a^{r_k} - 1) \bmod N), N)$.

Second phase: the classical one

Let $b = a^R \bmod N$ and $\gcd(b, N) = 1$.
Hyp. $p-1 = Qs$ with $Q \mid R$ and s prime, $B_1 < s \le B_2$.

Test: is $\gcd(b^s - 1, N) > 1$ for some s.

$s_j = j$-th prime. In practice all $s_{j+1} - s_j$ are small (Cramer's conjecture implies $s_{j+1} - s_j \le (\log B_2)^2$).

- Precompute $c_\delta \equiv b^\delta \bmod N$ for all possible δ (small);
- Compute next value with one multiplication
  $b^{s_{j+1}} = b^{s_j} c_{s_{j+1}-s_j} \bmod N$.

Cost: $O((\log B_2)^2) + O(\log s_1) + (\pi(B_2) - \pi(B_1))$ multiplications $+(\pi(B_2) - \pi(B_1))$ gcd's. When $B_2 \gg B_1$, $\pi(B_2)$ dominates.

Rem. We need a table of all primes $< B_2$; memory is $O(B_2)$.

Record. Nohara (66dd of $960^{119} - 1$, 2006; see `http://www.loria.fr/~zimmerma/records/Pminus1.html`).

Second phase: BSGS

Select $w \approx \sqrt{B_2}$, $v_1 = \lceil B_1/w \rceil$, $v_2 = \lceil B_2/w \rceil$.

Write our prime s as $s = vw - u$, with $0 \le u < w$, $v_1 \le v \le v_2$.

Lem. $\gcd(b^s - 1, N) > 1$ iff $\gcd(b^{wv} - b^u, N) > 1$.

Algorithm:

1. Precompute $b^u \bmod N$ for all $0 \le u < w$.
2. Precompute all$(b^w)^v$ for all $v_1 \le v \le v_2$.
3. For all u and all v evaluate $\gcd(b^{vw} - b^u, N)$.

Number of multiplications: $w + (v_2 - v_1) + O(\log_2 w) = O(\sqrt{B_2})$

Memory: $O(\sqrt{B_2})$.

Number of gcd**:** $\pi(B_2) - \pi(B_1)$.

Second phase: using fast polynomial arithmetic

Algorithm:
1. Compute $h(X) = \prod_{0\leq u<w}(X - b^u) \in \mathbb{Z}/N\mathbb{Z}[X]$
2. Evaluate all $h((b^w)^v)$ for all $v_1 \leq v \leq v_2$.
3. Evaluate all $\gcd(h(b^{wv}), N)$.

Analysis:

Step 1: $O((\log w)\mathsf{M}_{\mathrm{pol}}(w))$ operations (using a product tree).

Step 2: $O((\log w)\mathsf{M}_{\mathrm{int}}(\log N))$ for b^w; $v_2 - v_1$ for $(b^w)^v$; multi-point evaluation on w points takes $O((\log w)\mathsf{M}_{\mathrm{pol}}(w))$.

Rem. Evaluating $h(X)$ along a geometric progression of length w takes $O(w \log w)$ operations (see Montgomery-Silverman).

Total cost: $O((\log w)\mathsf{M}_{\mathrm{pol}}(w)) = O\left(B_2^{0.5+o(1)}\right)$.

Trick: use $\gcd(u, w) = 1$ and $w = 2 \times 3 \times 5 \ldots$.

Second phase: using the birthday paradox

Consider $\mathcal{B} = \langle b \bmod p\rangle$; $s := \#\mathcal{B}$.

If we draw $\approx \sqrt{s}$ elements at random in $\mathcal{B}$, then we have a collision (birthday paradox).

Algorithm: build (b_i) with $b_0 = b$, and

$$b_{i+1} = \begin{cases} b_i^2 \bmod N & \text{with proba } 1/2 \\ b_i^2\, b \bmod N & \text{with proba } 1/2. \end{cases}$$

We gather $r \approx \sqrt{s}$ values and compute

$$\prod_{i=1}^{r}\prod_{j\neq i}(b_i - b_j) = \mathrm{Disc}(P(X)) = \prod_i P'(b_i)$$

where

$$P(X) = \prod_{i=1}^{r}(X - b_i).$$

$\Rightarrow$ use fast polynomial operations again.