MPRI – Cours 2-12-2

ÉCOLE POLYTECHNIQUE F. Morain CNRS INRIA

Lecture IIb: Introduction to integer factorization

2009/11/30

I. Introduction.

II. Finding small factors of integers.

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

IV. ECM.

I. Introduction

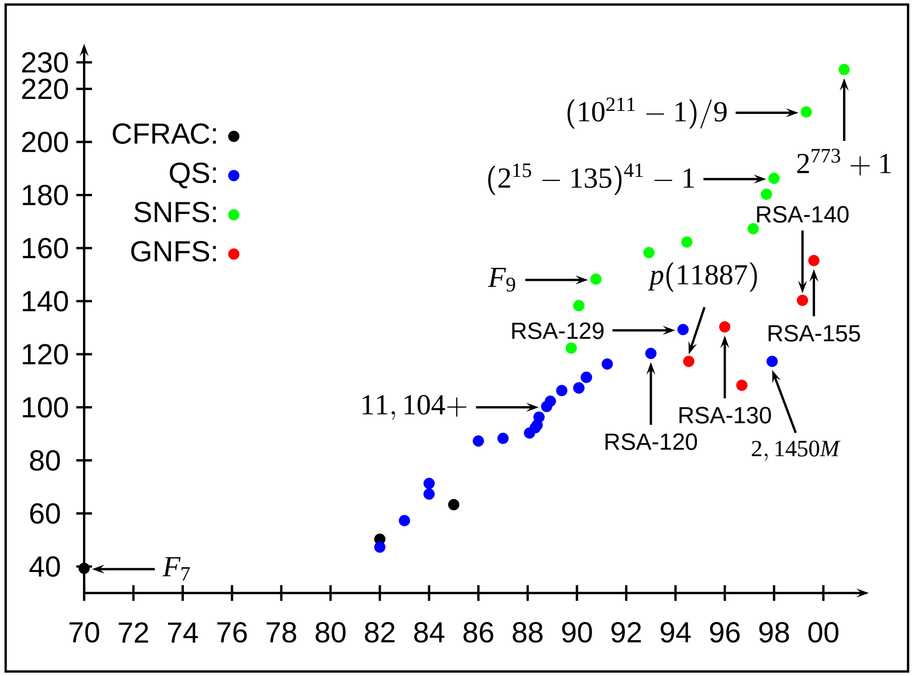
Input: an integer N;

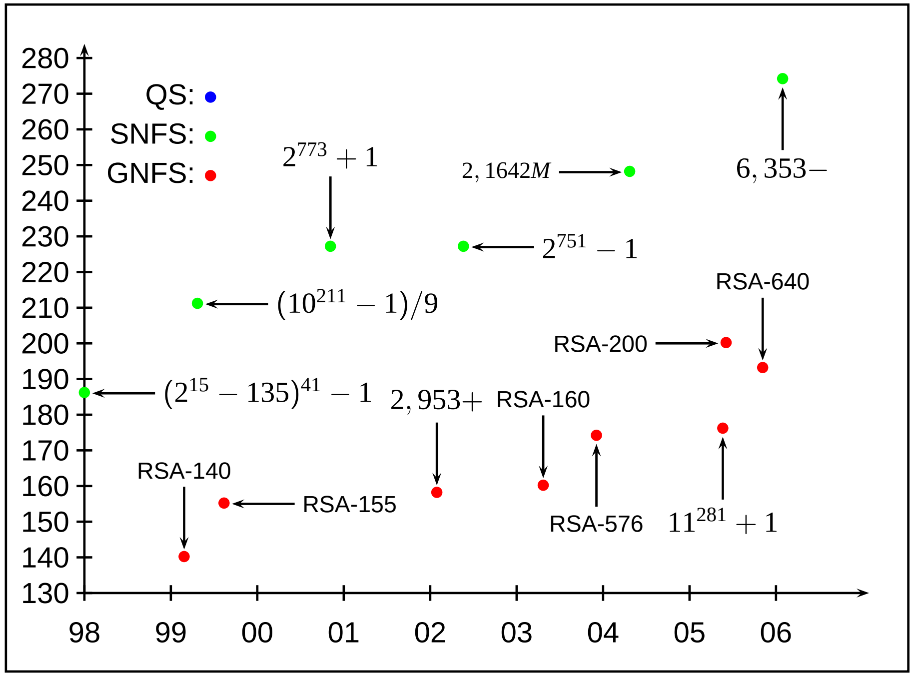
Output: $N = \prod_{i=1}^{k} p_i^{\alpha_i}$ with p_i (proven) prime.

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?





More figures for RSA-*dd*

dd	who	when	time
100	Manasse & A. K. Lenstra	1991	7 MIPS-year
110	AKL	1992	one month on $5/8$ of a MasPar 16K
120	AKL, Dodson, Denny, Manasse, Lioen, te Riele	1993	835 MIPS-year
129	Atkins, Graff, AKL, Leyland + INTERNET	1994	5000 MIPS-year
130	Dodson, Montgomery, Elkenbracht-Huizing, AKL, WWW, Fante, Leyland, Weber, Zayer	1996	500 MIPS-year
140	te Riele, Cavallar, Lioen, Montgomery, Dodson, AKL, Leyland, Murphy, Zimmermann	1999	1500 MIPS-year
155	CABAL	1999	8000 MIPS-year
200	Franke et al.	05/2005	60 years 2.2GHz Opteron

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

Size of the factors: $D_k = \lim_{N\to+\infty} \log N_k / \log N$ 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.

A crucial species: smooth numbers

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

Main interest: all relation collecting algorithms (Quadratic sieve, index calculus, etc.).

de Bruijn's function:

$$\psi(x, y) = \mathrm{Card}\{N \leq x, p \mid N \Rightarrow p \leq y\}.$$

Main theorem: (Canfield, Erdős, Pomerance) For all $\varepsilon > 0$, uniformly in $y \geq (\log x)^{1+\varepsilon}$, when $x \to \infty$

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

A useful function:

$$L(x) = \exp\left(\sqrt{\log x \log\log x}\right).$$

Prop. For all $\alpha > 0$, $\beta > 0$, when $x \to \infty$

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

II. Finding small factors of integers

Pb. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ be a finite set of primes, $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ a finite sequence of integers. For x in $\mathcal{X}$, define the $\mathcal{P}$-smooth part of x

$$F(x) = \prod_{\substack{p \in \mathcal{P} \\ p^e || x}} p^e.$$

How can we compute all $F(x)$ rapidly?

Basic case: $\mathcal{P}_B = \{2, 3, \ldots, B\}$; $\mathcal{X} = \{x_1\}$.

Rem. building $\mathcal{P}_B$ is a classical exercise (Eratosthenes sieve); $B = 2^{32}$ is not a problem (store $(p_{i+1} - p_i)/2$ as a `char`).

A) Trial division

Algorithm: divide all x's by all p's.

Claims:

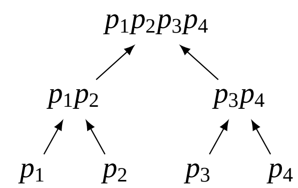
- Multiplication of two n-bit integers (resp. degree n polynomials over a ring R) can be realized in $O(\mathsf{M}_{\mathrm{int}}(n))$ (resp. $O(\mathsf{M}_{\mathrm{pol}}(n))$) bit-operations (resp. operations in R) with traditional (resp. best) value of n^2 (resp. $n \log n \log \log n$).
- Quotient and remainder of $a(X)$ of degree $n+m$ by $b(X)$ of degree n can be done using $O(\mathsf{M}_{\mathrm{pol}}(m) + \mathsf{M}_{\mathrm{pol}}(n) + n)$ operations over R. A $2n$-bit integer divided by a n-bit one takes $O(\mathsf{M}_{\mathrm{int}}(n))$.

Basic case: $\lg \mathcal{P}_B = \sum_{p \le B} \lg p = O(B \lg B)$ and TD costs $O(B^{1+o(1)}(\lg \mathcal{X}))$ (if all x_i have the same size).

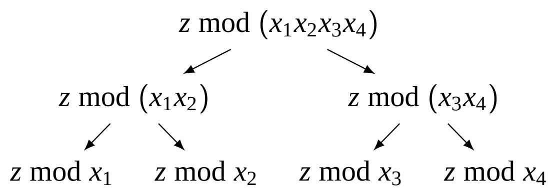
B) Product trees

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

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



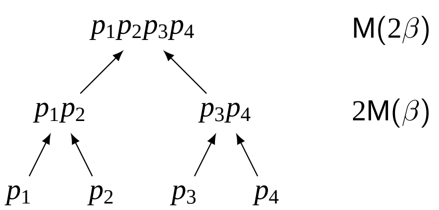
2. [Remainder tree] Compute $z \bmod x_1, \ldots, z \bmod x_n$.



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

Product trees (cont'd)

Imagine all p_i's have the same size β.



Product tree: $2\mathsf{M}(\beta) + \mathsf{M}(2\beta)$.

Naive case: $\underbrace{p_1p_2}_{\mathsf{M}(\beta)} + \underbrace{(p_1p_2)p_3}_{\mathsf{M}(2\beta,\beta)} + \underbrace{(p_1p_2p_3)p_4}_{\mathsf{M}(3\beta,\beta)} \approx 6\mathsf{M}(\beta)$.

Comparison: $4\mathsf{M}(\beta)$ vs. $\mathsf{M}(2\beta)$? Equal if $\mathsf{M}(\beta) = \beta^2$, product tree better if $\mathsf{M}(\beta) = \beta^a$, $a < 2$.

General principle: only the last step counts.

Validity and analysis

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

Analysis: given $b =$ total number of bits in $\mathcal{P}$ and $\mathcal{X}$, $O((\lg b)\mathsf{M}_{\mathrm{int}}(b) = O(b(\lg b)^{2+o(1)})$.

Step 1: $O(\log m \mathsf{M}_{\mathrm{int}}(b))$.

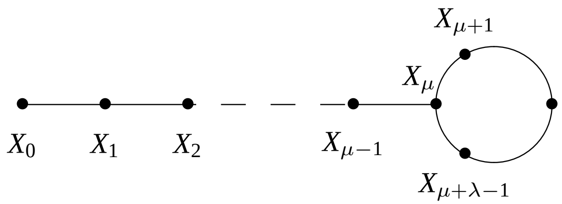
Step 2: $O(\log n \mathsf{M}_{\mathrm{int}}(b))$.

Step 3: $O(b_k(\lg b)\mathsf{M}_{\mathrm{int}}(b_k))$ since $e \in O(\lg b)$; overall cost is obtained via $\sum b_k = O(b)$.

Rem. If more information is needed, use Bernstein for $b(\lg b)^{3+o(1)}$. See Bernstein's web page.

C) Pollard's ρ (again!)

Prop. Let $f : E \to E$, $\#E = m$; $X_{n+1} = f(X_n)$ with $X_0 \in E$. The functional digraph of X is:



Fact. There exists a unique $e > 0$ (epact) s.t. $\mu \leq e < \lambda + \mu$ and $X_{2e} = X_e$. On average, $e = O(\sqrt{m})$.

Floyd's algorithm:

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


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.

Thm. (Bach, 1991) Proba finding $p \mid N$ after k iterations is at least

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

when p goes to infinity.

In practice:

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

D) Pollard Strassen

Input: $B \leq \sqrt{N}$.

Output: smallest $p \leq B$ dividing N if any.

0. Let $C = \lceil \sqrt{B} \rceil$.

1. Compute $f(X) = \prod_{1 \leq j \leq C} (X + j) \in \mathbb{Z}/N\mathbb{Z}[X]$.

2. Compute all $g_i = f(iC) \in \mathbb{Z}/N\mathbb{Z}$ for $0 \leq i < C$.

3. if $\gcd(g_i, N) = 1$ for all i then return FAILURE else set $k = \min_i \{\gcd(g_i, N) > 1\}$.

4. return $\min_d \{kc + 1 \leq d \leq kc + c, d \mid N\}$.

Validity: $p \mid N$ and $p \mid f(iC)$ for some $0 \leq i < C$ if and only if p divides some number in $\{iC + 1, \ldots, iC + C\}$.

Analysis

Step 1: product tree again, hence $O(\mathsf{M}_{\rm pol}(C)\log C)$ additions and multiplications in $\mathbb{Z}/N\mathbb{Z}$.

Step 2: multipoint evaluation $O(\mathsf{M}_{\rm pol}(C)\log C)$, same as remainder tree, since $f(a) = f(X) \bmod (X-a)$.

Step 3: C gcd's for a cost of $O(C\mathsf{M}_{\rm int}(\log N)\log\log N)$.

Step 4: $O(C\mathsf{M}_{\rm int}(\log N))$.

Total: $O(\mathsf{M}_{\rm pol}(B^{0.5})\mathsf{M}_{\rm int}(\log N)(\log B + \log\log N))$. Deterministic.

Rem. Bostant/Gaudry/Schost got rid of the $\log B$ term.

III. Pollard's $p-1$ method

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

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

Same if some R is known s.t. $p-1 \mid R$ and we compute

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

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

Let s_j denote the 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. Zimmermann (58 dd of $2^{2098}+1$, 2005).

Second phase: faster

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 $0 \le u < w$, $v_1 \le v \le v_2$. One has $\gcd(b^s - 1, N) > 1$ iff $\gcd(b^{wv} - b^u, N) > 1$.

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 is $w + (v_2 - v_1) + O(\log_2 w) = O(\sqrt{B_2})$, memory is also $O(\sqrt{B_2})$.

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

Second phase: faster

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

Analysis:

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

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

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

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

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

Continuing $p-1$ with the birthday paradox

Consider $\mathcal{B} = \langle b \bmod p\rangle$. By hypothesis, $\#\mathcal{B} = s$.
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.

Rem. This idea can be reused in many factoring algorithms.

Just the beginning of the story

- Prototype of the Φ_k factoring methods: Williams's $p+1$ method, Bach + Shallit.
- Quadratic forms.
- ECM.

IV. ECM

Rem. Over a ring, the addition law must be defined with some care, due to singular points (see Lenstra for a rigorous presentation).

$$E_N = \{\, (x, y),\ y^2 \equiv x^3 + ax + b \bmod N \,\} \cup \{\, O_N \,\},$$

s.t. $\gcd(4a^3 + 27b^2, N) = 1$. A point is an element of E_N. Note for all prime $p \mid N$, E_p is actually an elliptic curve.

Pseudo-addition: given two points P and Q, returns
- either a point R s.t. $P_p \oplus Q_p = R_p$ for all $p \mid N$;
- or a divisor d of N.

ECM: Pollard's $p-1$ on a curve

Algorithm:

- find $(E, P = (x_0, y_0))$ s.t. $y_0^2 \equiv x_0^3 + ax_0 + b \bmod N$;
- compute $[R]P$ on E until the pseudo-addition returns a factor of N.
- all variants of $p-1$ works, except the FFT 2nd phase.

An example:

$$E_N : y^2 \equiv x^3 + x + 1 \bmod 143, P = (0, 1)$$

$$Q = [2]P = (36, 124)$$

$$[2]Q = (127, 71).$$

Computing $[3]Q = [3!]P$:

$$\lambda = (124 - 71) \times (36 - 127)^{-1} \bmod 143.$$

But

$$\gcd(36 - 127, 143) = \gcd(52, 143) = 13.$$

ECM: analysis

ECM works when there exists E_N s.t. E_p has a smooth cardinality. Varying E makes $\#E$ vary $\Rightarrow$ we can use a lot of them.

Rationale: $\#E_p$ should behave as a random number $\approx p$.

Conjecture (Lenstra) Let $L(x) = \exp\left(\sqrt{\log x \log\log x}\right)$. Using $L(p)^{1/\sqrt{2}}$ curves, we can find $p \mid N$ with $O(L(p)^{\sqrt{2}})$ operations on curves.

In practice:

- very very efficient for $p \approx 10^{30--40}$, record of p with 67 decimal digits.
- Many tricks known: fast elliptic group laws (even Edwards); forcing a torsion subgroup is also possible.