MPRI – Cours 2-12

ÉCOLE POLYTECHNIQUE

F. Morain

CNRS INRIA

II. Integer factorization

2007/09/24

I. Introduction.

II. Finding small factors of integers.

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

IV. Combining congruences.

I. Introduction

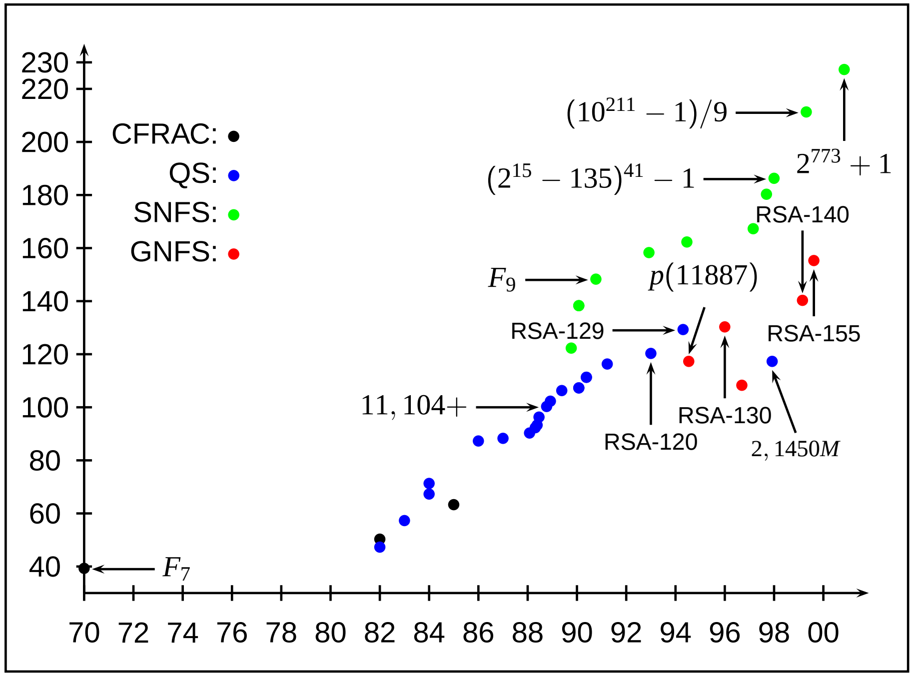
Input: an integer N;

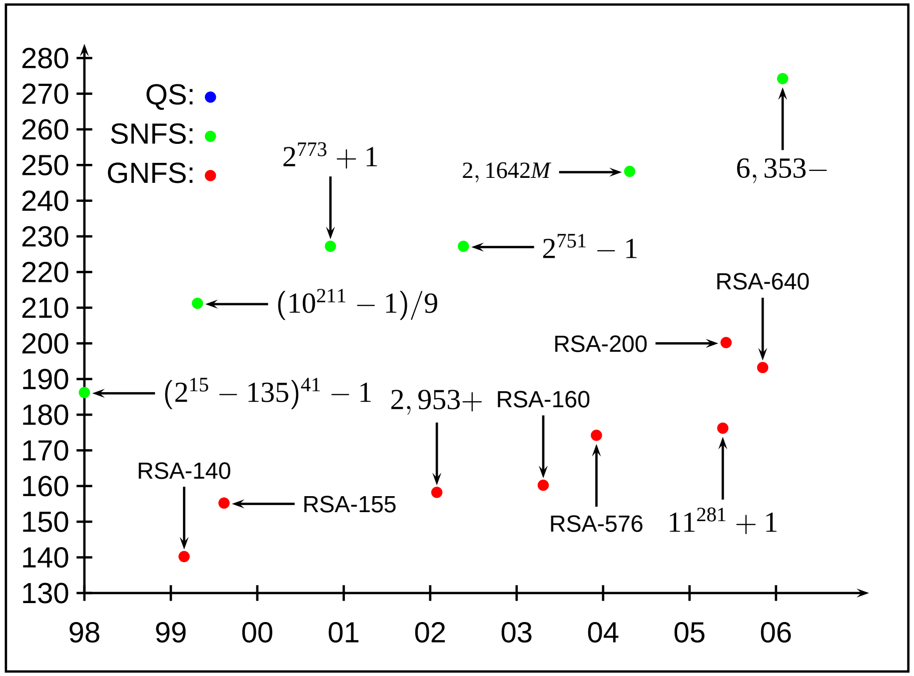
Output: $N = \prod_{i=1}^{k} p_i^{\alpha_i}$ with p_i (proven) prime.

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.

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:

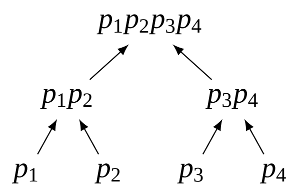
- Multiplication of two n-bit integers (resp. degree n polynomials over a ring R) can be realized in $O(\mathsf{M}_{\mathrm{int}}(n))$ (resp. $O(\mathsf{M}_{\mathrm{pol}}(n))$) bit-operations (resp. operations in R) with traditional (resp. best) value of n^2 (resp. $n \log n \log\log n$).
- Quotient 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: $\lg \mathcal{P}_B = \sum_{p\leq B} \lg p = O(B \lg B)$ and TD costs $O(B^{1+o(1)}(\lg \mathcal{X}))$ (if all x_i have the same size).

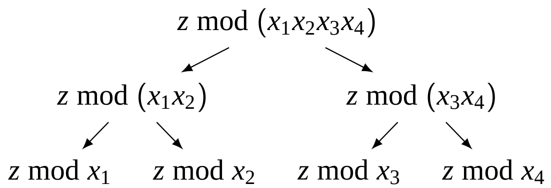
B) Product trees

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

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



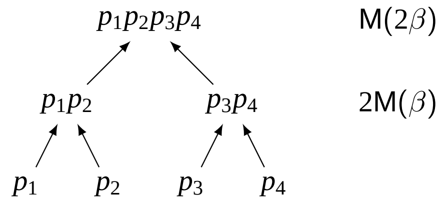
2. [Remainder tree] Compute $z \bmod x_1, \ldots, z \bmod x_n$.



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

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

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

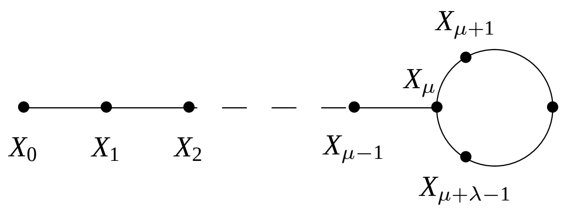
Step 2: $O(\log n \mathsf{M}_{\text{int}}(b))$.

Step 3: $O(b_k(\lg b)\mathsf{M}_{\text{int}}(b_k))$ 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.

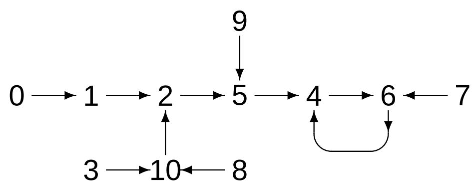
C) Pollard's ρ

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



Ex1. If $E_m = G$ is a finite group with m elements, and $a \in G$ of ordre N, $f(x) = ax$ and $x_0 = a$, (x_n) is purely periodic, i.e., $\mu = 0$, and $\lambda = N$.

Ex2. Soit $E_m = \mathbb{Z}/11\mathbb{Z}$, $f : x \mapsto x^2 + 1 \bmod 11$:



Epact

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

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

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 $\mu > 0$, $e = \lceil \frac{\mu}{\lambda} \rceil \lambda$.

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

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 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 $O(\mathsf{M}_{\mathrm{pol}}(C)\log C)$ additions and multiplications in $\mathbb{Z}/N\mathbb{Z}$.

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

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

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

Total: $O(\mathsf{M}_{\mathrm{pol}}(B^{0.5})\mathsf{M}_{\mathrm{int}}(\log N)(\log B + \log\log N))$. 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: $R = \prod_{p^\alpha \le B_1} p^\alpha = \mathrm{lcm}(1, 2, \ldots, B_1)$.

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 $h(X) = \prod_{0 \leq u < w}(X - b^u) \in \mathbb{Z}/N\mathbb{Z}[X]$
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: $O((\log w)\mathsf{M}_{\mathrm{pol}}(w)) = O(B_2^{0.5+o(1)})$.

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 (see Gaudry's part).

IV. Combining congruences

A) Basics.

B) Naive methods.

C) The quadratic sieve and extensions.

D) Linear algebra.

A) Basics

Kraitchik's idea: find x s.t. $x^2 \equiv 1 \bmod N$, $x \not\equiv \pm 1 \bmod N$.

Step 0: build a prime basis $\mathcal{B} = \{p_1, p_2, \ldots, p_k\}$.

Step 1: find a lot of relations $(R_i)_{i \in I}$: $R_i = \prod_{j=1}^{k} p_j^{a_{i,j}} \equiv 1 \bmod N$

Step 2: find $I' \subset I$ s.t.

$$\prod_{i \in I'} R_i = x^2$$

over $\mathbb{Z}$, which is equivalent to

$$\forall\, j, \sum_{i \in I'} a_{i,j} \equiv 0 \bmod 2,$$

which is a classical linear algebra problem.

Step 3: x is a squareroot of 1 and with probability $\geq 1/2$, $\gcd(x-1, N)$ is non-trivial.

B) Naive methods

A very naive one:

0. Build $\mathcal{B} = \{p_1 = 2, 3, \ldots, p_k\}$.

1. Generate k random relations

$$p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \bmod N$$

and hope to factor the residue to get:

$$p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k} \bmod N$$

from which

$$p_1^{e_1 - f_1} p_2^{e_2 - f_2} \cdots p_k^{e_k - f_k} \equiv 1 \bmod N.$$

Store the $(e_i - f_i) \bmod 2$ in the matrix M.

2. Find dependancies relations of M and deduce solutions of $x^2 \equiv 1 \bmod N$.

Hypothesis:

$$x(e) = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \bmod N$$

is a random integer in $[1..N[$.

A numerical example

Let $N = 143$, $\mathcal{B} = \{2, 3, 5\}$. We compute:

$$(R_1) : 2^3 \times 3 \times 5^4 \equiv 2^7 \bmod N,$$
$$(R_2) : 2^3 \times 3^3 \times 5^4 \equiv 2^3 \bmod N,$$
$$(R_3) : 3^3 \times 5^4 \equiv 1 \bmod N.$$

Combining (R_1) and (R_2), we get:

$$(2^{-2} \times 3^2 \times 5^4)^2 \equiv 1 \bmod N$$

or $12^2 \equiv 1 \bmod N$ and $\gcd(12 - 1, N) = 11$.

De Bruijn's function

Define

$$\psi(x, y) = \#\{z \leq x, z \text{ is } y - \text{smooth}\}.$$

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

Prop. Let

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

For all real $\alpha > 0$, $\beta > 0$, as $x \rightarrow \infty$

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

Analysis

Prop. The cost of the naive algorithm is $O(L^{2+o(1)})$.
Proof.
Proba($x(e)$ is p_k-smooth)$= \frac{\psi(N,p_k)}{N} \Rightarrow$ we need $k\frac{N}{\psi(N,p_k)}$ relations.
Using trial division, testing p_k-smoothness costs k divisions.
Linear algebra costs $O(k^r)$ with $2 \le r \le 3$ (see later).
Total cost is:

$$O\left(k^2\frac{N}{\psi(N,p_k)}\right) + O(k^r).$$

Put $k = L(N)^b$, from which $p_k \approx k \log k = O(L(N)^{b+o(1)})$. Cost is now:

$$O(L^{2b}L^{1/(2b)}) + O(L^{rb}) = O(L^{\max(2b+1/(2b),rb)}).$$

$2b + 1/(2b)$ is minimal for $b = 1/2$ and has value 2, which is larger than rb for all r. $\square$

Going further

Suppose that $x(e) \le N^\alpha$ and replace trial division with a $L(N)^{db}$ for some d. Then optimize

$$f = \max((1+d)b + \alpha/(2b), rb).$$

Prop. Let $\beta_m = \sqrt{\alpha/(2(d+1))}$ and $\beta = \sqrt{\alpha/(2(r-(d+1)))}$. Then f is minimal for

$$b = \min(\beta_m, \beta) = \begin{cases} \beta_m & \text{if } d+1 \ge r/2,\ f = 2(d+1)\beta_m \\ \beta & \text{otherwise, } f = r\beta \end{cases}$$

Application: using Bernstein or Pollard-Strassen leads to $d = 0$, and therefore the complexity is $L(N)^{r/\sqrt{2(r-1)}}$.

Dixon's variant: generate $y^2 \equiv \prod_j p_j^{a_{i,j}} \bmod N$. Same complexity.

C) Quadratic sieve

Pb. The above methods are not practical, since factoring the relations is too costly. Can we build residues of size N^α for $\alpha < 1$?

CFRAC: (Morrison and Brillhart) use the continued fraction expansion of $\sqrt{N}$, leads to residues of size $N^{1/2}$; first real-life algorithm, factored F_7 in 1970.

Schroeppel's linear sieve: relations $F(a,b) = (\lfloor\sqrt{N}\rfloor + a)(\lfloor\sqrt{N}\rfloor + b) - N$ for small a and b satisfy

$$F(a,b) \equiv (\lfloor\sqrt{N}\rfloor + a)(\lfloor\sqrt{N}\rfloor + b) \bmod N$$

and $N = \lfloor\sqrt{N}\rfloor^2 + R$, $R = O(\sqrt{N})$. All numbers have size $O(\sqrt{N})$.
Moreover, if $p \mid F(a,b)$, then $p \mid F(a+p,b)$, etc.

Pomerance's quadratic sieve:

Use $a = b$

$$\left(a + \left\lfloor\sqrt{N}\right\rfloor\right)^2 \equiv \left(a + \left\lfloor\sqrt{N}\right\rfloor\right)^2 - N \approx 2a\sqrt{N}.$$

$$p \mid F(a) \Longleftrightarrow \left(a + \left\lfloor\sqrt{N}\right\rfloor\right)^2 \equiv N \bmod p$$

implies N is a square modulo p and
$p \mid F(a) \Leftrightarrow a \equiv a_-$ or $a \equiv a_+ \bmod p$.
Prop. The cost is $O(L(N)^{r/\sqrt{4(r-1)}})$.
Proof. Precomputing all roots of $F(a) \bmod p$ costs L^b.
The cost of sieving over $|a| \le L^c$ is

$$\sum_{p \le L^b} \frac{2L^c}{p} = L^{c+o(1)}.$$

The number of L^b-smooth values of $F(a)$ in the interval is $L^{c-1/(4b)} \Rightarrow$ take $c = b + 1/(4b)$ and optimize $L^{\max(b,b+1/(4b),rb)}$ which yields $b = 1/\sqrt{4(r-1)}$. $\square$

Real life implementation

Multiplier: factor kN instead of N for small k so as to have a lot of small prime factors in $\mathcal{B} = \{p, (\frac{kN}{p}) = +1\}$.

Early abort strategy: (CFRAC) abandon factorization of residue if doesn't seem to factor.

Large prime variation: suppose we end up with

$$x(e)^2 = (\prod p)C$$

for some $p_k < C(e) < p_k^2$. Then we know that $C(e)$ is prime. We can keep the relation and hope for another

$$x(e')^2 = (\prod p)C$$

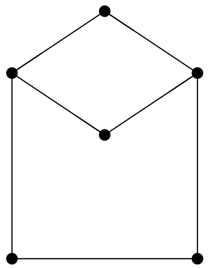
so that $(x(e)x(e')/C)^2$ is factored over $\mathcal{B}$. Works due to the birthday paradox. Use hashing to store C's.

Two large primes

$$F(a_1) = (\prod_1)p_1p_2, \quad F(a_2) = (\prod_2)p_2p_3, \quad F(a_3) = (\prod_3)p_3p_1$$

$$\Rightarrow (F(a_1)F(a_2)F(a_3))/(p_1p_2p_3)^2 \equiv (\prod_{123}) \bmod N.$$

Store (p_1, p_2) as an edge in the graph $G = (V, E)$ whose vertices are the p_i's (not in $\mathcal{B}$), including $(1, p_2)$.



Idea: inject new edges as they arrive and count the cycles that appear. Then use a classical graph algorithm to find actual cycles.

When adding (p, q):

1. $p \notin V$, $q \notin V$: a new component appears;
2. $p \in V$, $q \notin V$: no new component, nor cycle;
3. $p \in V$, $q \in V$, p and q not in the same component: two components merge, no cycle;
4. $p \in V$, $q \in V$, p and q in the same component: a cycle appears.

Prop. Let v be the number of vertices, e the number of edges, and c the number of connected components. Then the number of cycles is $c + e - v$.

Rem. a classical *union-find* is used to manage components.

Real sieving in QS

Never factor residues, but test

$$\mathcal{R}(a) = \log|F(a)| - \sum_{\substack{p^e \mid F(a) \\ p \in \mathcal{B}}} \log p^e < \log p_k$$

and replace $\log|F(a)|$ by $\log|2a\sqrt{N}|$. In practice, fits in a `char`; use integer approximations; ignore small primes.

Large primes: relax $\mathcal{R}(a) < 2\log p_k$, say.

MPQS: (Montgomery, 1985) use families of quadratic polynomials $\Rightarrow$ massive computations become possible: email (A. K. Lenstra & M. S. Manasse, 1990), INTERNET (RSA-129).

Rem. a lot more tricks exist (SIQS, etc.).

D) Linear algebra

Fundamental property: combination matrices are sparse, since $\Omega(N) \leq \log_2 N$.

N	size	$\#$coeffs $\neq 0$ per relation
RSA-100	$50,000 \times 50,000$	
RSA-110	$80,000 \times 80,000$	
RSA-120	$252,222 \times 245,810$ $(89,304 \times 89,088)$	
RSA-129	$569,466 \times 524,338$ $(188,614 \times 188,160)$	47
RSA-130	$3,504,823 \times 3,516,502$	39
RSA-140	$4,671,181 \times 4,704,451$	32
RSA-155	$6,699,191 \times 6,711,336$	62
RSA-160	$5,037,191 \times 5,037,191$	??
$6,353-$	$19,591,108 \times 19,590,832$	229

Gauss

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Computations:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ x & x & x & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ 1 & 0 & 1 & \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Finally:

$$L_1 + L_3 = 0, L_1 + L_2 + L_4 = 0$$

Rem. The companion matrix can be merged into A.

Rem. additings rows use XOR's on `unsigned long` (in C). Still in $O(k^3)$ but with a very small constant.

Structured Gaussian elimination: use a sparse encoding of M and perform elimination so as to slow the fill-in down as much as possible.

Going further: Lanczos and Wiedemann benefit from sparse encoding, and cost $O(k^{2+\varepsilon})$. Many subtleties.

Biggest open problem: how to distribute this phase in a clean an efficient way? Currently the bottleneck of this kind of algorithms.

Conclusions

A broad view of integer factorization.

More to come: NFS.

Close: discrete log algorithms as companions to integer factorization.