# Integer Factoring heuristics

- The general algorithms to factor $n$ are very slow!
  Currently, only integers in $\approx 700$ bits (i.e. $\approx 200$ digits) could be factored in practice.
  ↳ that too using specialized hardware.

- The best <u>provable</u> complexity known is:
  Expected time $\exp(O(\sqrt{\lg n \cdot \lg\lg n}))$.

- <u>Heuristic</u> complexity is $\exp(O(\lg^{1/3} n \cdot \lg^{2/3}\lg n))$.

- We'll use the notation

$$L_x(\alpha, c) := \exp(c \cdot \log^{\alpha} x \cdot \log^{1-\alpha} \log x).$$

natural log

[Pomerance '89]: The general number field sieve (GNFS) has conjectured time complexity $L_n(1/3, 2)$.

- Why this strange function $L_x(\alpha, c)$?

Smooth numbers. Defn: Number $m$ is called $y$-smooth if its prime factors are $\leq y$. Their density is

$$\Psi(x,y) := \#\{1 \leq m \leq x \mid m \text{ is } y\text{-smooth}\}.$$

- Asymptotic estimates for $\psi(x,y)$ determine the complexity of advanced integer factoring algorithms.

<u>Theorem</u> (Dickman–de Bruijn '51): $\psi(x,y) \geq x/u^u$,

where $u := \log_y x = (\log x)/\log y$.

<u>Pf. idea</u>: • Consider the regime $(\log y) < u < (y/\log y) =: t$.

• Consider primes $2 = p_1 < p_2 < \cdots < p_t$ that are $\leq y$.

• Any "good" $m$ looks like $\prod_{i\in[t]} p_i^{\alpha_i}$.

$$\Rightarrow \psi(x,y) \geq \#\left\{\bar{\alpha} \;\middle|\; \sum_{i=1}^{t} \alpha_i \leq u\right\} \geq \binom{u+t}{t} \geq \left(\frac{t}{u}\right)^u \geq \frac{y^u}{(u\cdot\log y)^u}$$

$$= x/(\log x)^u. \quad \square$$

— This bound is nontrivial only if $u^u << x$.

$\Rightarrow$ $y$ shouldn't be too small!

<u>Lemma</u>: A useful, tolerable $y$ is $L_x(\alpha, c)$, for constants $\alpha$ & $c$. Then, $u^u \approx L_x\left(1-\alpha, \frac{1-\alpha}{c}\right)$.

<u>Proof</u>:

- $\log y = \log L_x(\alpha, c) = c \cdot \log^{\alpha} x \cdot \log^{1-\alpha} \log x$

$\Rightarrow$ $u = (\log x)/\log y = \frac{1}{c} \cdot (\log^{1-\alpha} x) \cdot (\log^{\alpha-1} \log x)$

$\Rightarrow$ $u \cdot \log u \approx \frac{1}{c} \cdot (\log^{1-\alpha} x) \cdot (\log^{\alpha-1} \log x) \cdot (1-\alpha) \cdot (\log \log x)$

$= (1-\alpha)/c \cdot (\log^{1-\alpha} x) \cdot (\log \log x)^{\alpha}$

$\Rightarrow$ $u^u \approx L_x\left(1-\alpha, \frac{1-\alpha}{c}\right)$. $\square$

**Theorem:** For $y = L_x(\alpha, c)$, the probability of choosing a y-smooth $m \leq x$ is $\dfrac{\psi(x,y)}{x} \approx L_x\left(1-\alpha, \dfrac{\alpha-1}{c}\right)$.

- In integer factoring algos, the time spent depends on the bound $y$ & the above probability.

- Complexity of $L_n(\alpha, c)$, with $\alpha < 1$, is termed subexponential, in contrast to the exponential $L_n(1,c) = \exp(c \cdot \log n) = n^c$.

Eg., Eratosthenes Sieve algo. takes $L_n(1, \frac{1}{2})$ time.

## Special-case factoring

- They work better than brute-force on special $n$.

## Pollard's rho method (1975)

- Idea is to exploit the presence of a moderately "small" $p \mid n$. (Brute-force is $\tilde{O}(p)$.)

**Input:** odd $n > 1$ & a pseudorandom function $f(x)$ (say, $f := x^2 + 1 \bmod n$)

**Output:** Factor $n$ in $\tilde{O}(\sqrt{p} \cdot \lg n)$-time.

1) Randomly pick $x \in [n]$. Let $y := x$ & $d := 1$.

2) While $d = 1$

- $x \leftarrow f(x)$ ; $y \leftarrow f(f(y))$ ;
- $d \leftarrow \gcd(x-y, n)$ ;

3) If $d \neq n$ then OUTPUT $d$, else FAIL.

Assumption: $\begin{cases} p \text{ is the smallest factor of } n, \\ \{f^{\circ i}(x) \mid i \geq 0\} \text{ is a random sequence.} \end{cases}$

Lemma: Whp (in Step 2) $p \mid (x-y)$ in $O(\sqrt{p})$ iterations.

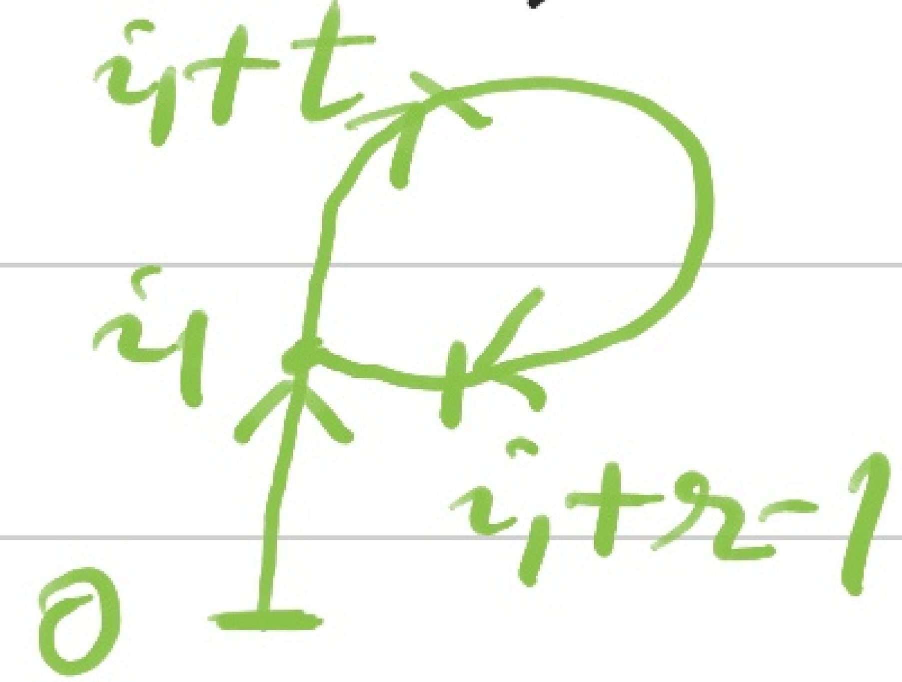
Pf: • Prob. of $\{f^{\circ i}(x) \bmod p \mid 0 \leq i < j\}$ being distinct

$$\leq \frac{p}{p} \cdot \frac{p-1}{p} \cdots \frac{p-j}{p} = \left(1-\frac{1}{p}\right) \cdots \left(1-\frac{j}{p}\right) \approx e^{-\frac{1}{p}-\frac{2}{p}-\cdots-j/p} \approx e^{-j^2/p}$$

$\Rightarrow$ For suitable $j = O(\sqrt{p})$, the probability of a repetition (mod $p$) is good. [Birthday Paradox!]

$\Rightarrow$ $\exists\ 0 \le i_1 < i_2 \le O(\sqrt{p})$, $f^{\circ i_1}(x) \equiv f^{\circ i_2}(x) \bmod p$.

$\Rightarrow$ Let $r := i_2 - i_1$ be the <u>period</u>.




• $(i_1+t)$-th iteration of Step-2 gives us:

$$f^{\circ(i_1+t)}(x) \quad \& \quad f^{\circ 2(i_1+t)}(x).$$

$\Rightarrow$ <u>Collision mod $p$</u> happens if $(i_1+t) \equiv 2(i_1+t) \bmod r$

$\Leftarrow$ $t = (r - i_1)$ is the first collision.

$\Rightarrow$ $p \mid (x-y)$ at $r \le O(\sqrt{p})$-th iteration of Step-2. $\square$

▷ Whp $p \mid d$ (Step-2) in $\tilde{O}(\sqrt{p}\cdot \lg n)$ -time.

<u>Success</u>: Brent & Pollard (1980) factored Fermat number $F_8 := 2^{2^8}+1$ into primes of 16 & 62-digits. (in 2 hours on a UNIVAC)

## <u>Pollard's $p-1$ Method</u> (1974)

– It exploits the <u>smoothness</u> of $(p-1)$, for prime $p \mid n$.

<u>Input</u>: odd $n>1$ not a perfect-power.

<u>Output</u>: Factors $n$.

1) For $r = 2, 3, 4, \ldots, R$:
- Randomly pick $a \in (\mathbb{Z}/n)^*$
- $d \leftarrow (a^k - 1, n)$, for $k := (r!)^{\lceil \lg n \rceil}$.
- If $d \notin \{1, n\}$, then OUTPUT $d$.

<u>Assumption</u>: $\exists$ primes $p \neq q \mid n$ s.t. $(p-1)$ is $R$-smooth but $(q-1)$ is <u>not</u>.

<u>Lemma</u>: Whp $n$ is factored in $\tilde{O}(R \cdot \lg^2 n)$-time.

Pf: • For $r = R$: $(p-1) \mid k$ while $(q-1) \nmid k$.

$\Rightarrow \forall a \in (\mathbb{Z}/n)^*$: $a^k \equiv 1 \mod \langle p \rangle$;

While, for $< \frac{1}{2}$ of the $a$'s : $a^k \equiv 1 \mod \langle q \rangle$.

$\Rightarrow$ $d$ factors $n$.

- Time to compute $a^k - 1 \pmod{n}$ is $\tilde{O}(\lg n \cdot r \lg n)$
- We could reach $R$ by using binary-search

$$\Rightarrow \text{overall time} = \lg R \cdot \tilde{O}(R \cdot \lg^2 n)$$
$$= \tilde{O}(R \cdot \lg^2 n). \qquad \square$$

<u>Success</u>: In GIMPS (Great Internet Mersenne Prime Search), this is used to eliminate composites.

## <u>Fermat Method</u>

- Tries to write $n = a^2 - b^2$. Iterate over $\sqrt{n + b^2}$.
  Works well if a factor of $n$ is <u>very close</u> to $\sqrt{n}$.

▷ $n = c \cdot d = \left(\frac{c+d}{2}\right)^2 - \left(\frac{c-d}{2}\right)^2$

$\Rightarrow$ ▷ $c \approx \sqrt{n} \Leftrightarrow d \approx \sqrt{n} \Leftrightarrow c-d$ is "small".

1) For $x = 1, 2, 3, \ldots$
   - If $n+x^2$ is <u>square</u>, then compute $y \leftarrow \sqrt{n+x^2}$ & OUTPUT $y-x$.

▷ It's time complexity is $\tilde{O}(m \cdot \lg n)$, where $m := c-d$.

- (Lehman '74) made it general purpose with time complexity $\tilde{O}(n^{1/3})$.

Success: The idea of finding two squares equal mod $n$, gives more advanced algorithms.

↳ Kraitchik's family of algos.

- Consider $Q(X) := X^2 - n$
- Find $x_1, \ldots, x_k \in \mathbb{Z}$ s.t. $Q(x_1) \cdots Q(x_k) = v^2$ is a square in $\mathbb{Z}$.

$\Rightarrow (x_1 \cdots x_k)^2 \equiv v^2 \mod n$

$\Rightarrow$ hopefully, $\gcd(x_1 \cdots x_k - v, n)$ factors $n$.

## Lehmer & Powers (1931)

– Compute the continued fraction $\sqrt{n} =: \underline{a_0} + \frac{1}{\underline{a_1} +} \frac{1}{\underline{a_2} +} \frac{1}{\underline{a_3} +} \cdots$

▷ Convergents $\left\{ a_0, \frac{x_1}{y_1} := a_0 + \frac{1}{a_1}, \frac{x_2}{y_2} := a_0 + \frac{1}{a_1 + \frac{1}{a_2}}, \ldots \right\}$ give improving approximations to $\sqrt{n}$ & satisfy:

$$Q_i := x_i^2 - n y_i^2 \quad , \quad |Q_i| < 2\sqrt{n} .$$

– Since $Q_i$'s are "small" one hopes to find $i_1 < i_2 < \cdots < i_k$ s.t. $Q_{i_1} \cdots Q_{i_k} = \underline{v^2}$ is an integral square.

$$\Rightarrow (x_{i_1} \cdots x_{i_k})^2 \equiv v^2 \pmod{n} .$$

# Morrison & Brillhart's implementation (1970)

- Idea: Use $Q_i$'s that are $B$-smooth.

1) Fix a bound $B$. Let $\{p_1, \ldots, p_B\}$ be the first $B$ prime numbers. (factor-base)
2) Compute set $S := \{ Q_i \mid Q_i =: (-1)^{\alpha_{i0}} p_1^{\alpha_{i1}} \cdots p_B^{\alpha_{iB}} \}$ s.t. $|S| = B+2$.
3) Consider the $B+2$ exponent-vectors $\{ (\alpha_{i0}, \ldots, \alpha_{iB}) \mid Q_i \in S \}$ $=: \overline{\alpha_i}$.
   Compute $T \subseteq S$ s.t. $\{ \overline{\alpha_i} \mid Q_i \in T \}$ sum $0 \bmod 2$.
4) $\Rightarrow$ $\prod \{ Q_i \in T \}$ is an integral square $v^2$.
5) OUTPUT $\gcd\left( \prod_{Q_i \in T} x_i - v, n \right)$.

- **Assumption**: $\{Q_i = x_i^2 - ny_i^2 \mid i \geq 1\}$ is random.

**Theorem**: The algorithm takes time $L_n\left(\frac{1}{2}, \sqrt{2}+o(1)\right)$.

**Pf**: • $\Pr[Q_i \text{ is } p_B\text{-smooth}] \approx \psi(\sqrt{n}, p_B)/\sqrt{n}$

$\Rightarrow$ expected # of $i$'s after which we get $(B+2)$, $p_B$-smooth $Q_i$'s $\approx B \cdot \sqrt{n}/\psi(\sqrt{n}, p_B)$

• Complexity is dominated by the smoothness test, which takes time $\approx B^2 \cdot \sqrt{n}/\psi(\sqrt{n}, p_B)$

• Set $B := L_{\sqrt{n}}(\alpha, c)$ to get $L_{\sqrt{n}}(\alpha,c)^2 \cdot L_{\sqrt{n}}\left(1-\alpha, \frac{1-\alpha}{c}\right)$.

$$= \exp\left(2c(\log\sqrt{n})^{\alpha} \cdot (\log\log\sqrt{n})^{1-\alpha} + \frac{1-\alpha}{c} \cdot (\log\sqrt{n})^{1-\alpha} \cdot (\log\log\sqrt{n})^{\alpha}\right)$$

which is minimized at $\alpha = c = 1/2$, to get:

$$= \exp\left(2 \cdot (\log\sqrt{n})^{1/2} \cdot (\log\log\sqrt{n})^{1/2}\right)$$

$$\approx \exp\left(\sqrt{2 \cdot \log n \cdot \log\log n}\right) = L_n\left(\tfrac{1}{2}, \sqrt{2}\right).$$

at $B := L_{\sqrt{n}}\left(\frac{1}{2}, \frac{1}{2}\right) = L_n\left(\frac{1}{2}, \frac{1}{2\sqrt{2}}\right).$ $\square$

$\ll L_n(1, o(1))$ !

<u>Success:</u> $F_7 = 2^{2^7}+1$ was factored into primes of 17 & 22 digits. Other $\approx$ 70-digit numbers.

* used C. frac of $\sqrt{257 \cdot F_7}$.

## <u>Quadratic Sieve</u>

– Pomerance (1981) suggested: (1) <u>sieving</u> idea to reduce the smoothness-test complexity.

(2) Use $Q(x) := x^2 - n$, keeping $x \approx \sqrt{n}$.

↳ Improves to $L_n(\frac{1}{2}, 1)$ !

<u>Modified Algo:</u> 1) Instead compute $Q(x) \leftarrow x^2 - n$
for $x \in \{\lfloor\sqrt{n}\rfloor + 1, \ldots, \lfloor\sqrt{n}\rfloor + N\}$,
for $N := B \cdot \sqrt{n} / \psi(\sqrt{n}, p_B)$.

2) Check smoothness as: For $i \in [B]$,
- Look at $2N/p_i$ places (only!) in the sequence that are $\equiv 0 \bmod p_i$.
- Modify the list by dividing them by $p_i^*$ (highest-power).

3) Places left with "1" were <u>$p_B$-smooth</u>.

$$\Rightarrow \text{Time now} \approx \sum_{i\in[B]} \frac{2N}{p_i} \approx N \cdot \log\log B$$

$$\approx B \cdot \log\log B \cdot \sqrt{n} / \psi(\sqrt{n}, p_B)$$

$$\approx L_{\sqrt{n}}(\alpha, c) \cdot L_{\sqrt{n}}\left(1-\alpha, \frac{1-\alpha}{c}\right), \text{ for } B := L_{\sqrt{n}}(\alpha, c).$$

$\hookrightarrow$ Minimized at $\alpha = \frac{1}{2}$ & $c = 1/\sqrt{2}$, to get:

$$\approx L_{\sqrt{n}}\left(\tfrac{1}{2}, \tfrac{1}{\sqrt{2}}\right)^2 \approx L_n\left(\tfrac{1}{2}, 1\right) \ll L_n\left(\tfrac{1}{2}, \sqrt{2}\right)$$

$$\text{for } B := L_n\left(\tfrac{1}{2}, \tfrac{1}{2}\right). \qquad \square$$

$\Rightarrow$ Allows for a 2-fold increase in the length of $n$.

Success: Lenstra & Manasse (1994) factored a 129-digit RSA-challenge using distributed computing over the internet!

## Number Field Sieve (NFS)

- Pollard (1988) suggested using algebraic number fields to factor numbers like $x^3+k$, where $k$ is small & $x$ is large.

  Lenstra, Lenstra & Manasse (1990) improved it & factored $F_9 = 2^{2^9}+1$ into 3 primes (of 7, 49 & 99 digits).

**Idea:** • Quadratic sieve can be seen as using the norm $\underline{N}$: $\mathbb{Q}(\sqrt{n}) \longrightarrow \mathbb{Q}$

$$x+y\sqrt{n} \mapsto x^2 - ny^2 .$$

• $\prod_i N(x_i - y_i\sqrt{n}) = N\left(\prod_i (x_i - y_i\sqrt{n})\right) = v^2$

make it a square in $\mathbb{Z}[\sqrt{n}]$.

• Its high-order generalization is to go to a <u>number-field</u> $K := \mathbb{Q}(\alpha) = \mathbb{Q}[X]/\langle f(x)\rangle$, where $f$ is an irreducible of $\deg = (d+1)$.

• Then, use <u>smooth numbers</u> in $\mathbb{Z}[\alpha]$.

integer ring of number field