Fast Chinese remaindering in practice

Joris van der Hoeven

CNRS, École polytechnique

Vienna, November 16, 2017

http://www.TeXmacs.org

Integer matrix multiplication

$M \in \mathbb{Z}^{n \times n}$ $\quad$ $N \in \mathbb{Z}^{n \times n}$

$|M_{i,j}| < 2^p$ $\quad$ $|N_{i,j}| < 2^p$

$$P = M N$$

$$|P_{i,j}| < n\, 2^{2p}$$

Integer matrix multiplication

$M \in \mathbb{Z}^{n \times n}$ $\quad$ $N \in \mathbb{Z}^{n \times n}$

$|M_{i,j}| < 2^p$ $\quad$ $|N_{i,j}| < 2^p$

$$P = M N$$

$$|P_{i,j}| < n\, 2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\, n\, 2^{2p}$$

$$p_k < 2^{52}$$

Integer matrix multiplication

$M \in \mathbb{Z}^{n \times n}$ $\quad$ $N \in \mathbb{Z}^{n \times n}$

$|M_{i,j}| < 2^p$ $\quad$ $|N_{i,j}| < 2^p$

$$P = M N$$

$$|P_{i,j}| < n\, 2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\, n\, 2^{2p}$$

$$p_k < 2^{26}$$

Integer matrix multiplication

$M \in \mathbb{Z}^{n \times n}$ $\quad$ $N \in \mathbb{Z}^{n \times n}$

$|M_{i,j}| < 2^p$ $\quad$ $|N_{i,j}| < 2^p$

$$P = M\,N$$

$$|P_{i,j}| < n\,2^{2\,p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\,n\,2^{2\,p}$$

$$p_k < 2^{23}$$

Integer matrix multiplication

$M \in \mathbb{Z}^{n \times n}$ $N \in \mathbb{Z}^{n \times n}$

$|M_{i,j}| < 2^p$ $|N_{i,j}| < 2^p$

$$P = M N$$

$$|P_{i,j}| < n\, 2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\, n\, 2^{2p}$$

$$p_k < 2^{23}$$

Multi-modular reduction

$M_k = M \bmod p_k \in \mathbb{F}_{p_k}^{n \times n}$ $N_k = N \bmod p_k \in \mathbb{F}_{p_k}^{n \times n}$

Integer matrix multiplication

$M \in \mathbb{Z}^{n \times n}$ $\quad$ $N \in \mathbb{Z}^{n \times n}$

$|M_{i,j}| < 2^p$ $\quad$ $|N_{i,j}| < 2^p$

$$P = M\,N$$

$$|P_{i,j}| < n\,2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\,n\,2^{2p}$$

$$p_k < 2^{23}$$

Multi-modular reduction

$M_k = M \bmod p_k \in \mathbb{F}_{p_k}^{n \times n}$ $\quad$ $N_k = N \bmod p_k \in \mathbb{F}_{p_k}^{n \times n}$

Modular multiplications

$$P_k = M_k\,N_k \in \mathbb{F}_{p_k}^{n \times n}$$

Integer matrix multiplication

$$M \in \mathbb{Z}^{n \times n} \qquad N \in \mathbb{Z}^{n \times n}$$

$$|M_{i,j}| < 2^p \qquad |N_{i,j}| < 2^p$$

$$P = M N$$

$$|P_{i,j}| < n\, 2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\, n\, 2^{2p}$$

$$p_k < 2^{23}$$

Multi-modular reduction

$$M_k = M \bmod p_k \in \mathbb{F}_{p_k}^{n \times n} \qquad N_k = N \bmod p_k \in \mathbb{F}_{p_k}^{n \times n}$$

Modular multiplications

$$P_k = M_k N_k \in \mathbb{F}_{p_k}^{n \times n}$$

Multi-modular reconstruction (Chinese remaindering)

$$P_1, \ldots, P_\ell \longmapsto P$$

Integer matrix multiplication

$$M \in \mathbb{Z}^{n \times n} \qquad N \in \mathbb{Z}^{n \times n}$$

$$|M_{i,j}| < 2^p \qquad |N_{i,j}| < 2^p$$

$$P = M\,N$$

$$|P_{i,j}| < n\,2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\,n\,2^{2p}$$

$$p_k < 2^{23}$$

Multi-modular reduction

$$M_k = M \bmod p_k \in \mathbb{F}_{p_k}^{n \times n} \qquad N_k = N \bmod p_k \in \mathbb{F}_{p_k}^{n \times n}$$

Modular multiplications

$$P_k = M_k\,N_k \in \mathbb{F}_{p_k}^{n \times n}$$

Multi-modular reconstruction (Chinese remaindering)

$$P_1, \ldots, P_\ell \longmapsto P$$

Integer matrix multiplication

$$M \in \mathbb{Z}^{n\times n} \qquad N \in \mathbb{Z}^{n\times n}$$

$$|M_{i,j}| < 2^p \qquad |N_{i,j}| < 2^p$$

$$P = M\,N$$

$$|P_{i,j}| < n\, 2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\, n\, 2^{2p}$$

$$p_k < 2^{23}$$

Multi-modular reduction → cost $O(\ell^2\, n^2)$

$$M_k = M \bmod p_k \in \mathbb{F}_{p_k}^{n\times n} \qquad N_k = N \bmod p_k \in \mathbb{F}_{p_k}^{n\times n}$$

Modular multiplications

$$P_k = M_k\, N_k \in \mathbb{F}_{p_k}^{n\times n}$$

Multi-modular reconstruction (Chinese remaindering)

$$P_1, \ldots, P_\ell \longmapsto P$$

Integer matrix multiplication

$$M \in \mathbb{Z}^{n \times n} \qquad N \in \mathbb{Z}^{n \times n}$$
$$|M_{i,j}| < 2^p \qquad |N_{i,j}| < 2^p$$
$$P = M\,N$$
$$|P_{i,j}| < n\, 2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\, n\, 2^{2p}$$
$$p_k < 2^{23}$$

Multi-modular reduction → cost $O(\ell^2\, n^2)$

$$M_k = M \bmod p_k \in \mathbb{F}_{p_k}^{n \times n} \qquad N_k = N \bmod p_k \in \mathbb{F}_{p_k}^{n \times n}$$

Modular multiplications → cost $O(\ell\, n^3)$

$$P_k = M_k\, N_k \in \mathbb{F}_{p_k}^{n \times n}$$

Multi-modular reconstruction (Chinese remaindering)

$$P_1, \ldots, P_\ell \longmapsto P$$

Integer matrix multiplication

$$M \in \mathbb{Z}^{n\times n} \qquad N \in \mathbb{Z}^{n\times n}$$

$$|M_{i,j}| < 2^p \qquad |N_{i,j}| < 2^p$$

$$P = M\,N$$

$$|P_{i,j}| < n\,2^{2p}$$

Pick primes

$p_1, \ldots, p_\ell$ pairwise distinct primes

$$p_1 \cdots p_n \geqslant 2\,n\,2^{2p}$$

$$p_k < 2^{23}$$

Multi-modular reduction → cost $O(\ell^2\,n^2)$

$$M_k = M \bmod p_k \in \mathbb{F}_{p_k}^{n\times n} \qquad N_k = N \bmod p_k \in \mathbb{F}_{p_k}^{n\times n}$$

Modular multiplications → cost $O(\ell\,n^3)$

$$P_k = M_k\,N_k \in \mathbb{F}_{p_k}^{n\times n}$$

Multi-modular reconstruction (Chinese remaindering) → cost $O(\ell^2\,n^2)$

$$P_1, \ldots, P_\ell \longmapsto P$$

How to reduce the $O(\ell^2)$ cost of multi-modular reduction and Chinese remaindering for small values of ℓ?

How to reduce the $O(\ell^2)$ cost of multi-modular reduction and Chinese remaindering for small values of ℓ?

Observation 1

Many (namely $\Theta(n^2)$) reductions/reconstructions with respect to the same moduli.

→ We are allowed to perform pre-computations as a function of $p_1, \ldots, p_\ell$

How to reduce the $O(\ell^2)$ cost of multi-modular reduction and Chinese remaindering for small values of ℓ?

Observation I

Many (namely $\Theta(n^2)$) reductions/reconstructions with respect to the same moduli.

→ We are allowed to perform pre-computations as a function of $p_1, \ldots, p_\ell$

Observation II

We are free to choice the moduli $p_1, \ldots, p_\ell$ as long as they satisfy the size constraints

→ Can we pick special primes that allow for speed-ups?

Asymptotically fast algorithms

- Remainder trees → $O(\mathrm{I}(\ell) \log \ell)$
  Fiduccia 1972, Moenck–Borodin 1972–1974

  $\mathrm{I}(\ell)$: complexity of ℓ-bit integer multiplication

  $\mathrm{I}(\ell) = O(\ell \log \ell \, K^{\log^* \ell})$, with $K = \begin{cases} 6 & \text{[Harvey 2017]} \\ 4 & \text{[Harvey–vdH–Lecerf 2014, prime hypothesis]} \end{cases}$

Asymptotically fast algorithms

- Remainder trees → $O(\mathrm{I}(\ell)\log \ell)$
  Fiduccia 1972, Moenck–Borodin 1972–1974

  $\mathrm{I}(\ell)$: complexity of ℓ-bit integer multiplication

  $\mathrm{I}(\ell) = O(\ell \log \ell \, K^{\log^* \ell})$, with $K = \begin{cases} 6 & \text{[Harvey 2017]} \\ 4 & \text{[Harvey–vdH–Lecerf 2014, prime hypothesis]} \end{cases}$

- $O(\mathrm{I}(\ell)\log \ell)$ with improved constants: Bostan–Lecerf–Schost 2003, Bernstein 2004

Asymptotically fast algorithms

- Remainder trees → $O(\mathrm{I}(\ell)\log \ell)$
  Fiduccia 1972, Moenck–Borodin 1972–1974

  $\mathrm{I}(\ell)$: complexity of ℓ-bit integer multiplication

  $\mathrm{I}(\ell) = O(\ell \log \ell \, K^{\log^* \ell})$, with $K = \begin{cases} 6 & \text{[Harvey 2017]} \\ 4 & \text{[Harvey–vdH–Lecerf 2014, prime hypothesis]} \end{cases}$

- $O(\mathrm{I}(\ell)\log \ell)$ with improved constants: Bostan–Lecerf–Schost 2003, Bernstein 2004

- $O\left(\mathrm{I}(\ell)\frac{\log \ell}{\log \log \ell}\right)$ modulo pre-computations: vdH 2017

Asymptotically fast algorithms

- Remainder trees → $O(\mathrm{I}(\ell)\log \ell)$
  Fiduccia 1972, Moenck–Borodin 1972–1974

  $\mathrm{I}(\ell)$: complexity of ℓ-bit integer multiplication

  $\mathrm{I}(\ell) = O(\ell \log \ell \, K^{\log^* \ell})$, with $K = \begin{cases} 6 & \text{[Harvey 2017]} \\ 4 & \text{[Harvey–vdH–Lecerf 2014, prime hypothesis]} \end{cases}$

- $O(\mathrm{I}(\ell)\log \ell)$ with improved constants: Bostan–Lecerf–Schost 2003, Bernstein 2004

- $O\left(\mathrm{I}(\ell)\frac{\log \ell}{\log \log \ell}\right)$ modulo pre-computations: vdH 2017

Faster algorithms for small ℓ

- Implementing fast modular arithmetic: vdH–Lecerf–Quintin 2016

Asymptotically fast algorithms

- Remainder trees → $O(\mathrm{I}(\ell)\log \ell)$
  Fiduccia 1972, Moenck–Borodin 1972–1974

  $\mathrm{I}(\ell)$: complexity of ℓ-bit integer multiplication

  $\mathrm{I}(\ell) = O(\ell \log \ell \, K^{\log^* \ell})$, with $K = \begin{cases} 6 & \text{[Harvey 2017]} \\ 4 & \text{[Harvey–vdH–Lecerf 2014, prime hypothesis]} \end{cases}$

- $O(\mathrm{I}(\ell)\log \ell)$ with improved constants: Bostan–Lecerf–Schost 2003, Bernstein 2004

- $O\left(\mathrm{I}(\ell)\frac{\log \ell}{\log \log \ell}\right)$ modulo pre-computations: vdH 2017

Faster algorithms for small ℓ

- Implementing fast modular arithmetic: vdH–Lecerf–Quintin 2016

- Reduction to fast linear algebra modulo pre-computations
  Doliskani–Giorgi–Lebreton–Schost 2017

Asymptotically fast algorithms

- Remainder trees → $O(\mathrm{I}(\ell)\log\ell)$
  Fiduccia 1972, Moenck–Borodin 1972–1974

  $\mathrm{I}(\ell)$: complexity of ℓ-bit integer multiplication

  $\mathrm{I}(\ell) = O(\ell \log \ell\, K^{\log^* \ell})$, with $K = \begin{cases} 6 & \text{[Harvey 2017]} \\ 4 & \text{[Harvey–vdH–Lecerf 2014, prime hypothesis]} \end{cases}$

- $O(\mathrm{I}(\ell)\log\ell)$ with improved constants: Bostan–Lecerf–Schost 2003, Bernstein 2004

- $O\left(\mathrm{I}(\ell)\frac{\log \ell}{\log\log \ell}\right)$ modulo pre-computations: vdH 2017

Faster algorithms for small ℓ

- Implementing fast modular arithmetic: vdH–Lecerf–Quintin 2016

- Reduction to fast linear algebra modulo pre-computations
  Doliskani–Giorgi–Lebreton–Schost 2017

Special prime numbers

- Pseudo-Mersenne primes $p_k = 2^n - \varepsilon$: cryptography, different complexity model
  Bajard–Kaihara–Plantard 2009

- w: suitable bitsize for fast machine modular arithmetic: $w = 48$, $w = 23$, etc.
- s: small positive integer, typically $4 \leqslant s \leqslant 8$.

- w: suitable bitsize for fast machine modular arithmetic: $w = 48$, $w = 23$, etc.
- s: small positive integer, typically $4 \leqslant s \leqslant 8$.

Idea: design fast conversions for pseudo-Mersenne moduli $m_1, \ldots, m_\ell$

$$\begin{aligned} m_i &= 2^{sw} + \delta_i \quad (i = 1, \ldots, \ell) \\ |\delta_i| &< 2^w \end{aligned}$$

- w: suitable bitsize for fast machine modular arithmetic: $w = 48$, $w = 23$, etc.
- s: small positive integer, typically $4 \leqslant s \leqslant 8$.

Idea: design fast conversions for pseudo-Mersenne moduli $m_1, \ldots, m_\ell$

$$\begin{aligned} m_i &= 2^{sw} + \delta_i \qquad (i = 1, \ldots, \ell) \\ |\delta_i| &< 2^w \end{aligned}$$

The three numbering systems

- $0 \leqslant x < m_1 \cdots m_\ell$: usual binary representation
- $(a_1, \ldots, a_\ell) = (x \text{ rem } m_1, \ldots, x \text{ rem } m_\ell)$: multi-modular representation
- $x = b_1 + b_2\, m_1 + b_3\, m_1\, m_2 + \cdots + b_\ell\, m_1 \cdots m_{\ell-1}$: mixed radix representation

- w: suitable bitsize for fast machine modular arithmetic: $w = 48$, $w = 23$, etc.
- s: small positive integer, typically $4 \leqslant s \leqslant 8$.

Idea: design fast conversions for pseudo-Mersenne moduli $m_1, \ldots, m_\ell$

$$\begin{aligned} m_i &= 2^{sw} + \delta_i \qquad (i = 1, \ldots, \ell) \\ |\delta_i| &< 2^w \end{aligned}$$

The three numbering systems

- $0 \leqslant x < m_1 \cdots m_\ell$: usual binary representation
- $(a_1, \ldots, a_\ell) = (x \text{ rem } m_1, \ldots, x \text{ rem } m_\ell)$: multi-modular representation
- $x = b_1 + b_2\, m_1 + b_3\, m_1\, m_2 + \cdots + b_\ell\, m_1 \cdots m_{\ell-1}$: mixed radix representation

Binary representation ⇝ multi-modular representation

- Euclidean division by $m_i \longrightarrow$ multiplication of quotient by δ_i
- Division by $(s\, w)$-bit number $\longrightarrow$ multiplication by w-bit number $\longrightarrow$ speed-up $\approx 2\, s$

- $m_i = 2^{sw} + \delta_i$
- $(a_1, \ldots, a_\ell) = (x \operatorname{rem} m_1, \ldots, x \operatorname{rem} m_\ell)$
- $x = b_1 + b_2\, m_1 + b_3\, m_1\, m_2 + \cdots + b_\ell\, m_1 \cdots m_{\ell-1}$

Multi-modular representation ⇝ mixed radix representation

$$b_1 = a_1$$

For $i \geqslant 2$, compute

$$u_{j,i} = (b_j + b_{j+1}\, m_j + \cdots + b_{i-1}\, m_j \cdots m_{i-2}) \operatorname{rem} m_i$$

using Horner

$$\begin{aligned} u_{i-1,i} &= b_{i-1} \\ u_{j,i} &= (b_j + u_{j+1,i} \cdot m_j) \operatorname{rem} m_i \qquad (j = i-2, \ldots, 1) \end{aligned}$$

We have

$$x \operatorname{rem} m_i = (u_{1,i} + b_i\, m_1 \cdots m_{i-1}) \operatorname{rem} m_i = a_i.$$

The inverse v_i of $m_1 \cdots m_{i-1}$ modulo m_i can be precomputed. Now

$$b_i = v_i\, (a_i - u_{1,i}) \operatorname{rem} m_i.$$

- $m_i = 2^{sw} + \delta_i$
- $(a_1, \ldots, a_\ell) = (x \operatorname{rem} m_1, \ldots, x \operatorname{rem} m_\ell)$
- $x = b_1 + b_2\, m_1 + b_3\, m_1\, m_2 + \cdots + b_\ell\, m_1 \cdots m_{\ell-1}$

Multi-modular representation ⇝ mixed radix representation

$$b_1 = a_1$$

For $i \geqslant 2$, compute

$$u_{j,i} = (b_j + b_{j+1}\, m_j + \cdots + b_{i-1}\, m_j \cdots m_{i-2}) \operatorname{rem} m_i$$

using Horner

$$\begin{aligned} u_{i-1,i} &= b_{i-1} \\ u_{j,i} &= (b_j + u_{j+1,i} \cdot (\delta_j - \delta_i)) \operatorname{rem} m_i \qquad (j = i-2, \ldots, 1) \end{aligned}$$

We have

$$x \operatorname{rem} m_i = (u_{1,i} + b_i\, m_1 \cdots m_{i-1}) \operatorname{rem} m_i = a_i.$$

The inverse v_i of $m_1 \cdots m_{i-1}$ modulo m_i can be precomputed. Now

$$b_i = v_i\, (a_i - u_{1,i}) \operatorname{rem} m_i.$$

- $m_i = 2^{sw} + \delta_i$
- $(a_1, \ldots, a_\ell) = (x \text{ rem } m_1, \ldots, x \text{ rem } m_\ell)$
- $x = b_1 + b_2\, m_1 + b_3\, m_1\, m_2 + \cdots + b_\ell\, m_1 \cdots m_{\ell-1}$

Mixed radix representation ⤳ binary representation

Horner style evaluation: for $i = \ell, \ldots, 1$, compute

$$x_i = b_i + b_{i+1}\, m_i + \cdots + b_\ell\, m_i \cdots m_{\ell-1}$$

using the recurrence relation

$$x_i = b_i + x_{i+1}\, m_i.$$

Complexity analysis

- Binary → multi-modular representation: $\sim\ell^2 s$ hardware multiplications
- Multi-modular → binary representation: $\sim\ell\, s\, (\ell + s)$ hardware multiplications

Main idea

- For the moment, our pseudo Mersenne moduli $m_i = 2^{sw} + \delta_i$ are too large.

Main idea

- For the moment, our pseudo Mersenne moduli $m_i = 2^{sw} + \delta_i$ are too large.
- We really need pseudo Mersenne moduli m_i that can be factored

$$\begin{aligned} m_i &= m_{i,1} \cdots m_{i,s} \\ m_{i,j} &< 2^{\mu} \end{aligned}$$

Main idea

- For the moment, our pseudo Mersenne moduli $m_i = 2^{sw} + \delta_i$ are too large.
- We really need pseudo Mersenne moduli m_i that can be factored

$$
\begin{aligned}
m_i &= m_{i,1} \cdots m_{i,s} \\
m_{i,j} &< 2^{\mu}
\end{aligned}
$$

- Here μ is the bit-size for machine modular arithmetic (slightly larger than s)

Main idea

- For the moment, our pseudo Mersenne moduli $m_i = 2^{sw} + \delta_i$ are too large.
- We really need pseudo Mersenne moduli m_i that can be factored

$$\begin{aligned} m_i &= m_{i,1} \cdots m_{i,s} \\ m_{i,j} &< 2^{\mu} \end{aligned}$$

- Here μ is the bit-size for machine modular arithmetic (slightly larger than s)
- We also need to require that the $m_{i,j}$ are pairwise coprime

Main idea

- For the moment, our pseudo Mersenne moduli $m_i = 2^{s w} + \delta_i$ are too large.
- We really need pseudo Mersenne moduli m_i that can be factored

$$\begin{aligned} m_i &= m_{i,1} \cdots m_{i,s} \\ m_{i,j} &< 2^{\mu} \end{aligned}$$

- Here μ is the bit-size for machine modular arithmetic (slightly larger than s)
- We also need to require that the $m_{i,j}$ are pairwise coprime

Gentle moduli

- s-gentle moduli: $m_{i,1}, \ldots, m_{i,s}$ of the above type
- Binary ⇔ multi-modular representation: $\sim \ell\, s\, (\ell + 2\, s)$ hardware multiplications

Main idea

- For the moment, our pseudo Mersenne moduli $m_i = 2^{sw} + \delta_i$ are too large.
- We really need pseudo Mersenne moduli m_i that can be factored

$$\begin{aligned} m_i &= m_{i,1} \cdots m_{i,s} \\ m_{i,j} &< 2^{\mu} \end{aligned}$$

- Here μ is the bit-size for machine modular arithmetic (slightly larger than s)
- We also need to require that the $m_{i,j}$ are pairwise coprime

Gentle moduli

- s-gentle moduli: $m_{i,1}, \ldots, m_{i,s}$ of the above type
- Binary $\Leftrightarrow$ multi-modular representation: $\sim \ell\, s\, (\ell + 2\, s)$ hardware multiplications

Additional trick

- Taking s even and $\delta_i = -\varepsilon_i^2$, we already have $m_i = (2^{sw/2} + \varepsilon_i)(2^{sw/2} - \varepsilon_i)$
- Conversions $\mathbb{Z}/(m_i\,\mathbb{Z}) \cong \mathbb{Z}/((2^{sw/2} + \varepsilon_i)\,\mathbb{Z}) \times \mathbb{Z}/((2^{sw/2} - \varepsilon_i)\,\mathbb{Z})$ are fast
- super s-gentle moduli: $2^{sw/2} + \varepsilon_i = m_{i,1} \cdots m_{i,s/2}$ and $2^{sw/2} - \varepsilon_i = m_{i,s/2+1} \cdots m_{i,s}$

- Determine $m_{i,j}$ once and for all for the desired size parameters s, w, μ, etc.
- We implemented a sieving procedure in MATHEMAGIX, making use of PARI-GP.

ε	m_1	m_2	m_3	m_4	m_5	m_6	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
27657	28867	4365919	6343559	13248371	20526577	25042063	29, 41, 43, 547, ...
57267	416459	1278617	2041469	6879443	25754563	28268089	416459, ...
77565	7759	8077463	8261833	18751793	19509473	28741799	59, 641, ...
95253	724567	965411	3993107	4382527	19140643	23236813	43, 724567, ...
294537	190297	283729	8804561	19522819	19861189	29537129	23^2, 151, 1879, ...
311385	145991	4440391	4888427	6812881	7796203	32346631	17, 79, 131, ...
348597	114299	643619	6190673	11389121	32355397	32442427	31, 277, ...
376563	175897	1785527	2715133	7047419	30030061	30168739	17, 127, 1471, ...
462165	39841	3746641	7550339	13195943	18119681	20203643	67, 641, 907, ...
559713	353201	873023	2595031	11217163	18624077	32569529	19, 59, 14797, ...
649485	21727	1186571	14199517	15248119	31033397	31430173	19, 109, 227, ...
656997	233341	1523807	5654437	8563679	17566069	18001723	79, 89, 63533, ...
735753	115151	923207	3040187	23655187	26289379	27088541	53, 17419, ...
801687	873767	1136111	3245041	7357871	8826871	26023391	23, 383777, ...
826863	187177	943099	6839467	11439319	12923753	30502721	73, 157, 6007, ...
862143	15373	3115219	11890829	18563267	19622017	26248351	31, 83, 157, ...
877623	514649	654749	4034687	4276583	27931549	33525223	41, 98407, ...
892455	91453	2660297	3448999	12237457	21065299	25169783	29, 397, 2141, ...

Table. List of 6-gentle moduli for $w = 22$, $\mu = 25$, and $\varepsilon < 1000000$.

The gentle modulus hunt (II)

ε	m_1	m_2	m_3	m_4	m_5	m_6	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
936465	543889	4920329	12408421	15115957	24645539	28167253	19, 59, 417721, ...
2475879	867689	4051001	11023091	13219163	24046943	28290833	867689, ...
3205689	110161	12290741	16762897	22976783	25740731	25958183	59, 79, 509, ...
3932205	4244431	5180213	5474789	8058377	14140817	25402873	4244431, ...
5665359	241739	5084221	18693097	21474613	23893447	29558531	31, 41, 137, ...
5998191	30971	21307063	21919111	22953967	31415123	33407281	101, 911, 941, ...
6762459	3905819	5996041	7513223	7911173	8584189	29160587	43, 137, 90833, ...
9245919	2749717	4002833	8274689	9800633	15046937	25943587	2749717, ...
9655335	119809	9512309	20179259	21664469	22954369	30468101	17, 89, 149, ...
12356475	1842887	2720359	7216357	13607779	23538769	30069449	1842887, ...
15257781	1012619	5408467	9547273	11431841	20472121	28474807	31, 660391, ...

Table. List of 6-gentle moduli for $w = 23$, $\mu = 25$ and $\varepsilon < 16000000$.

→ Taking w closer to μ greatly reduces the number of hits

→ But hits still tend to exist in sufficient number

ε	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
889305	50551	1146547	4312709	5888899	14533283	16044143	16257529	17164793
2447427	53407	689303	3666613	4837253	7944481	21607589	25976179	32897273
2674557	109841	1843447	2624971	5653049	7030883	8334373	18557837	29313433
3964365	10501	2464403	6335801	9625841	10329269	13186219	17436197	25553771
4237383	10859	3248809	5940709	6557599	9566959	11249039	22707323	28518509
5312763	517877	616529	879169	4689089	9034687	11849077	24539909	27699229
6785367	22013	1408219	4466089	7867589	9176941	12150997	26724877	29507689
7929033	30781	730859	4756351	9404807	13807231	15433939	19766077	22596193
8168565	10667	3133103	3245621	6663029	15270019	18957559	20791819	22018021
8186205	41047	2122039	2410867	6611533	9515951	14582849	16507739	30115277

Table. List of 8-gentle moduli for $w = 22$, $\mu = 25$ and $\varepsilon < 10000000$.

→ Increasing s greatly reduces the number of hits

→ Hits continue to exist for $s \leqslant 8$

The gentle modulus hunt (IV)

ε	m_1	m_2	···	m_5	m_6	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
15123	380344780931	774267432193	···	463904018985637	591951338196847	37, 47, 239, ...
34023	9053503517	13181369695139	···	680835893479031	723236090375863	29, 35617, ...
40617	3500059133	510738813367	···	824394263006533	1039946916817703	23, 61, 347, ...
87363	745270007	55797244348441	···	224580313861483	886387548974947	71, 9209, ...
95007	40134716987	2565724842229	···	130760921456911	393701833767607	19, 67, ...
101307	72633113401	12070694419543	···	95036720090209	183377870340761	41, 401, ...
140313	13370367761	202513228811	···	397041457462499	897476961701171	379, 1187, ...
193533	35210831	15416115621749	···	727365428298107	770048329509499	59, 79, ...
519747	34123521053	685883716741	···	705516472454581	836861326275781	127, 587, ...
637863	554285276371	1345202287357	···	344203886091451	463103013579761	79, 1979, ...
775173	322131291353	379775454593	···	194236314135719	1026557288284007	322131291353, ...
913113	704777248393	1413212491811	···	217740328855369	261977228819083	37, 163, 677, ...
1400583	21426322331	42328735049	···	411780268096919	626448556280293	21426322331, ...

Table. List of 6-gentle moduli for $w = 44$, $\mu = 50$ and $\varepsilon < 200000$. Followed by some super-gentle ones.

- Doubling w and μ tends to yield a much larger number of hits
- Sieving needs to be further optimized
- Suggestion for PARI-GP: an efficient routine for B-smooth factorization