Double-plus-one lifting and Applications to lattices

Arne Storjohann (Joint work with Colton Pauderis)
David R. Cheriton School of Computer Science
University of Waterloo

Outline

1. Variants of lifting for linear system.

$$A^{-1}b = v_0 + v_1 p + v_2 p^2 + v_3 p^3 + v_4 p^4 + v_5 p^5 + \cdots$$
$$A^{-1} = C_0 + C_1 p + C_2 p^2 + C_3 p^3 + C_4 p^4 + C_5 p^5 + \cdots$$

2. The high-order reside and its applications.

$$A^{-1} = (A^{-1} \text{ rem } p^k) + A^{-1} R p^k$$

3. Double-plus-one lifting. [ISSAC 2012, Pauderis and S.]

$$A^{-1} = (\cdots((B_0(I + R_0 p) + M_0 p^2)(I + R_1 p^3 + M_1 p^7) + \cdots)$$

4. Report on the implementation.

Part I: Lifting for linear systems

Variants of lifting include

1. Linear
   [1979, Moenck & Carter; 1982, Dixon]

2. Quadratic
   [Hensel/Newton iteration]

3. High-order
   [ISSAC 2002, S.]

4. Relaxed
   [ISSAC 2012, Barthomieu & Lebreton]

Recall: Linear systems

Every nonsingular rational (integer) matrix has an inverse

$$\begin{bmatrix} 67 & -81 & -77 & -2 & 69 & 10 \\ 29 & -9 & -18 & 27 & -74 & 94 \\ 44 & -50 & 87 & -93 & -4 & 12 \\ 92 & -22 & 33 & -76 & 27 & -2 \\ -31 & 45 & -98 & -72 & 8 & 50 \\ 99 & -16 & -38 & 57 & -32 & 25 \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{613719389}{436045910232} & \frac{20118095}{72674318372} & -\frac{851335927}{218022955116} & \frac{3893593471}{436045910232} & -\frac{433417321}{436045910232} & \frac{297871357}{72674318372} \\ -\frac{886053851}{145348636744} & \frac{620810971}{72674318372} & -\frac{1522814569}{72674318372} & \frac{3362614441}{145348636744} & -\frac{453009351}{145348636744} & -\frac{838573559}{72674318372} \\ -\frac{99479911}{109011477558} & \frac{218826900}{18168579593} & -\frac{674660030}{54505738779} & \frac{1856662385}{109011477558} & -\frac{1148992613}{109011477558} & -\frac{300458509}{18168579593} \\ \frac{447817619}{218022955116} & \frac{340592137}{36337159186} & -\frac{1594931579}{109011477558} & \frac{2114306231}{218022955116} & -\frac{2037856205}{218022955116} & -\frac{347815739}{36337159186} \\ \frac{770731325}{109011477558} & \frac{356811254}{18168579593} & -\frac{1721772194}{54505738779} & \frac{3697142975}{109011477558} & -\frac{1450539425}{109011477558} & -\frac{584700471}{18168579593} \\ \frac{2028363569}{436045910232} & \frac{1921892393}{72674318372} & -\frac{5197032317}{218022955116} & \frac{11614232501}{436045910232} & -\frac{4273458011}{436045910232} & -\frac{2043699293}{72674318372} \end{bmatrix}$$

Note: can express elements of $\mathbb{Q}$ as truncated p-adic expansions

$$-\frac{613719389}{436045910232} \leftrightarrow 70 + 58 \cdot 97 + 37 \cdot 97^2 + 40 \cdot 97^3 + 65 \cdot 97^4 + \cdots + 20 \cdot 97^{19}$$

<u>Lifting</u>

Given an $A \in \mathbb{Z}^{n \times n}$ and vector $b \in \mathbb{Z}^{n \times 1}$, lifting can be used to compute

1. the system solution $A^{-1}b$.
2. A^{-1} or interesting representations thereof.
3. a high-order residue R such that $A^{-1} = (A^{-1} \text{ rem } p^k) + A^{-1}Rp^k$.

General idea of lifting

Given

- an $n \times n$ integer matrix $A \in \mathbb{Z}^{n\times n}$
- a vector or matrix $b \in \mathbb{Z}^{n\times m}$, and
- a modulus p that is relatively prime to $\det A$

Note: usually p about same bitlength as entries in A

Lifting computes

- the p-adic expansion $A^{-1}b = v_0 + v_1 p + v_2 p^2 + v_3 p^3 + v_4 p^4 + \cdots$

$$\overbrace{\begin{bmatrix} -81 & -98 & -76 & -4 & 29 \\ -38 & -77 & -72 & 27 & 44 \\ -18 & 57 & -2 & 8 & 92 \\ 87 & 27 & -32 & 69 & -31 \\ 33 & -93 & -74 & 99 & 67 \end{bmatrix}}^{A}{}^{-1} \overbrace{\begin{bmatrix} -16 \\ -9 \\ -50 \\ -22 \\ 45 \end{bmatrix}}^{b} = \begin{bmatrix} -\frac{2784689}{4562102} \\ -\frac{2126771}{11405255} \\ -\frac{7886193}{22810510} \\ -\frac{5022303}{11405255} \\ -\frac{19469967}{22810510} \end{bmatrix} = \overbrace{\begin{bmatrix} 96 \\ 75 \\ 6 \\ 46 \\ 28 \end{bmatrix}}^{v_0} 97^0 + \overbrace{\begin{bmatrix} 20 \\ 50 \\ 91 \\ 24 \\ 7 \end{bmatrix}}^{v_1} 97 + \overbrace{\begin{bmatrix} 20 \\ 8 \\ 51 \\ 69 \\ 77 \end{bmatrix}}^{v_2} 97^2 + \cdots$$

Linear lifting to compute $A^{-1}b = v_0 + v_1 p + v_2 p^2 + v_3 p^3 + v_4 p^4 + \cdots$

[1979, Moenck & Carter; 1982, Dixon]

Precompute the first coefficient C_0 of $A^{-1} = C_0 + C_1 p + C_2 p^2 + \cdots$:

- cost is one matrix multiplication at precision p

C_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Now compute $v_0, v_1, v_2, \ldots$ in succession:

- each iteration requires two matrix$\times$vector products at precision p

v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
1	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
2	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
3	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
4	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
5	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Quadratic lifting to compute $A^{-1} = C_0 + C_1 p + C_2 p^2 + C_3 p^3 + \cdots$

[Hensel/Newton iteration]

Precompute C_0:

- cost is one matrix multiplication modulo at precision p

C_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Now double the precision at each step:

- iteration i requires a matrix multiplication at precision p^{2^i}

C_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
1	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
2	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
3	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
4	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
5	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

High-order lifting to compute $A^{-1}b = v_0 + v_1 p + v_2 p^2 + v_3 p^3 + \cdots$

Precompute key coefficients of $A^{-1} = C_0 + C_1 p + C_2 p^2 + \cdots$:

- cost is a logarithmic number of matrix multiplication at precision p

$\hat{C}_i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	●	○	●	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Now double the number of known coefficients at each step:

- iteration i requires an $n \times n$ by $n \times 2^i$ matrix multiplication at p

v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
1	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
2	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○
3	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○
4	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	○
5	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Part II: The high-order reside and its applications

$$A^{-1} = (A^{-1} \text{ rem } p^k) + A^{-1}R \cdot p^k$$

A scalar example: $A = 567$, $p = 10$, $k = 5, 10, 20, 10^{10^7}$:

$$\begin{aligned}
567^{-1} &= -1287 && + 567^{-1}(199) \cdot 10^5 \\
&= 1569664903 && + 567^{-1}(-89) \cdot 10^{10} \\
&= 29982363331569664903 && + 567^{-1}(-17) \cdot 10^{20} \\
&= (567^{-1} \text{ rem } 10^{10^7}) && + 567^{-1}(-79) \cdot 10^{10^7}
\end{aligned}$$

$$A^{-1} = (A^{-1} \text{ rem } p^k) + A^{-1}R \cdot p^k$$

A scalar example: $A = 567$, $p = 10$, $k = 5, 10, 20, 10^{10^7}$:

$$\begin{aligned} 567^{-1} &= -1287 &&+ 567^{-1}(199) \cdot 10^5 \\ &= 1569664903 &&+ 567^{-1}(-89) \cdot 10^{10} \\ &= 2998236331569664903 &&+ 567^{-1}(-17) \cdot 10^{20} \\ &= (567^{-1} \text{ rem } 10^{10^7}) &&+ 567^{-1}(-79) \cdot 10^{10^7} \end{aligned}$$

If we multiply last equation by 567 we obtain

$$1 = (1 + 79 \cdot 10^{10^7}) + (-79) \cdot 10^{10^7}$$

High-order residue computation

High-order lifting can compute an integer matrix R such that

$$A^{-1} = \overbrace{C_0 + C_1 p + C_2 p^2 + \cdots + C_{k-1} p^{k-1}}^{\text{most of these not computed}} + A^{-1} R p^k$$

<u>Example</u>

Consider $A = \begin{bmatrix} 59133654 & -10069961 \\ 7552448 & -1286118 \end{bmatrix}$ with $A^{-1} = \begin{bmatrix} \frac{643059}{322} & -\frac{10069961}{644} \\ \frac{1888112}{161} & -\frac{29566827}{322} \end{bmatrix}$

$$A^{-1} = (A^{-1} \text{ rem } 97^{1024}) + A^{-1} \overbrace{\begin{bmatrix} -29777071 & -33042015 \\ -3803076 & -4220069 \end{bmatrix}}^{R} 97^{1024}$$

High-order residue application: Unimodularity certification

<u>Fact:</u> The following are equivalent

- $\det A = \pm 1$ (i.e., A is unimodular)
- the p-adic expansion of A^{-1} is finite
- the high-order residue R is the zero matrix.

<u>Example:</u>

Is $A = \begin{bmatrix} 51 & 65 & \cdots & -50 \\ 76 & 86 & \cdots & -80 \\ -44 & 20 & \ddots & \vdots \\ 24 & -61 & \cdots & 25 \end{bmatrix} \in \mathbb{Z}^{8000 \times 8000}$ unimodular?

- Get $R \in \mathbb{Z}^{8000 \times 8000}$ s.t. $A^{-1} = (A^{-1} \text{ rem } 100^{16384}) + A^{-1}R \cdot 100^{16384}$.
- A is unimodular if and only if R is zero.

Cost proportional to $O(\log_2 16384 = 14)$ matrix multiplications.

High-order residue application: proper matrix fraction descriptions

Consider $A = \begin{bmatrix} 59133654 & -10069961 \\ 7552448 & -1286118 \end{bmatrix}$ with $A^{-1} = \begin{bmatrix} \frac{643059}{322} & -\frac{10069961}{644} \\ \frac{1888112}{161} & -\frac{29566827}{322} \end{bmatrix}$

$$A^{-1} = (A^{-1} \text{ rem } 97^{1000}) + A^{-1} \overbrace{\begin{bmatrix} -29777071 & -33042015 \\ -3803076 & -4220069 \end{bmatrix}}^{R} 97^{1000}$$

$$= (A^{-1} \text{ rem } 97^{1000}) + \overbrace{\begin{bmatrix} -\frac{171}{322} & -\frac{461}{644} \\ -\frac{26}{161} & -\frac{297}{322} \end{bmatrix}}^{A^{-1}R}$$

High-order residue for lattice reduction

Consider $A = \begin{bmatrix} 59133654 & -10069961 \\ 7552448 & -1286118 \end{bmatrix}$ with $A^{-1} = \begin{bmatrix} \frac{643059}{322} & -\frac{10069961}{644} \\ \frac{1888112}{161} & -\frac{29566827}{322} \end{bmatrix}$

1. Compute high-order residue R and the fraction $A^{-1}R = \begin{bmatrix} \frac{342}{644} & \frac{461}{644} \\ \frac{104}{644} & \frac{594}{644} \end{bmatrix}$.

2. Apply *Gradual sub-lattice reduction* [LATIN 2010, van Hoeij & Novocin]

$$\left[\begin{array}{cc|cc} 644 & & & \\ & 644 & & \\ \hline 342 & 461 & 1 & \\ 104 & 594 & & 1 \end{array}\right] \longrightarrow \left[\begin{array}{cc|cc} & & -26 & 5 \\ & & 4 & 24 \\ \hline 100 & -110 & 1 & 9 \\ 340 & 270 & 5 & 8 \end{array}\right]$$

3. A reduced lattice basis for A is $\left[\begin{array}{cc} 1 & 9 \\ \hline 5 & 8 \end{array}\right]$.

Ideas used for polynomial lattices [ISSAC 2003: Giorgi, Jeannerod & Villard]

Part III: Double-plus-one lifting for high-order residue computation

Linear lifting to compute $A^{-1} = C_0 + C_1 p + C_2 p^2 + C_3 p^3 + C_4 p^4 + \cdots$

Precompute C_0

- cost is one matrix multiplication at precision p

C_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Compute $C_1, C_2, C_3, \ldots$ in succession

- each iteration requires two matrix multiplications at precision p

C_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
1	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
2	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
3	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
4	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
5	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Linear lifting to compute $A^{-1} = C_0 + C_1 p + C_2 p^2 + C_3 p^3 + C_4 p^4 + \cdots$

- At the start of iteration i we have computed

$$A^{-1} = \overbrace{C_0 + C_1 p + \cdots + C_{i-1} p^{i-1}}^{\mathrm{Rem}(A^{-1}, p^i)} + A^{-1} R_i X^i$$

- Iteration i computes C_i and R_{i+1}.

Standard algorithm for linear lifting [1982, Dixon]

$C_0 := \mathrm{Rem}(A^{-1}, p)$
$R_0 := I_n$
for $i = 0$ **to** $k-1$ **do**
  $C_i := \mathrm{Rem}(C_0 R_i, p)$
  $R_{i+1} := (1/p)(R_i - AC_i)$

Quadratic lifting to compute $A^{-1} = C_0 + C_1 p + C_2 p^2 + C_3 p^3 + \cdots$ [Hensel/Newton iteration]

Precompute C_0:

- cost is one matrix multiplication modulo at precision p

C_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Now double the precision at each step:

- iteration i requires a matrix multiplication at precision p^{2^i}

C_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
1	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
2	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
3	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
4	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
5	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Quadratic lifting to compute $A^{-1} = C_0 + C_1 p + C_2 p^2 + C_3 p^3 + \cdots$

- At the start of iteration i we have computed

$$A^{-1} \equiv \overbrace{C_0 + C_1 p + \cdots + C_{i-1} p^{2^i - 1}}^{B = \mathrm{Rem}(A^{-1}, p^{2^i})} + A^{-1} R p^{2^i}$$

- Iteration i doubles the precision: $B = \mathrm{Rem}(A^{-1}, p^{2^{i+1}})$

Standard algorithm for quadratic lifting

$B := \mathrm{Rem}(A^{-1}, p)$
for $i = 0$ **to** $k-1$ **do**
  $R := (1/p^{2^i})(I - AB)$
  $B := \mathrm{Rem}(B(I + Rp^{2^i}), p^{2^{i+1}})$

- Can we optimize using loop unrolling / software pipelining?
- But the Rem operation (non arithmetic) is problematic!

Example of standard quadratic lifting

Quadratic lifting: standard

$B := \mathrm{Rem}(A^{-1}, p)$

for $i = 0$ **to** $k-1$ **do**

  $R := (1/p^{2^i})(I - AB)$

  $B := \mathrm{Rem}(B(I + Rp^{2^i}), p^{2^{i+1}})$

Example

$$\begin{aligned} 777^{-1} &= 3 + 777^{-1}(-233) \cdot 10 \\ &= 13 + 777^{-1}(-101) \cdot 10^2 \\ &= 8713 + 777^{-1}(-677) \cdot 10^4 \\ &= 12998713 + 777^{-1}(-101) \cdot 10^8 \end{aligned}$$

What happens if we omit the Rem?

Quadratic lifting: division free

$B := \mathrm{Rem}(A^{-1}, p)$

for $i = 0$ **to** $k-1$ **do**

$\quad R := (1/p^{2^i})(I - AB)$

$\quad B := B(I + Rp^{2^i})$

What happens if we omit the Rem?

Quadratic lifting: division free

$B := \mathrm{Rem}(A^{-1}, p)$
for $i = 0$ **to** $k-1$ **do**
  $R := (1/p^{2^i})(I - AB)$
  $B := B(I + Rp^{2^i})$

Example

$777^{-1} =$ $3 + 777^{-1}(-233) \cdot 10$

What happens if we omit the Rem?

Quadratic lifting: division free

$B := \mathrm{Rem}(A^{-1}, p)$
for $i = 0$ **to** $k-1$ **do**
  $R := (1/p^{2^i})(I - AB)$
  $B := B(I + Rp^{2^i})$

Example

$$\begin{aligned} 777^{-1} &= 3 + 777^{-1}(-233)\cdot 10 \\ &= -6987 + 777^{-1}(74289)\cdot 10^2 \end{aligned}$$

What happens if we omit the Rem?

Quadratic lifting: division free

$B := \mathrm{Rem}(A^{-1}, p)$
for $i = 0$ **to** $k-1$ **do**
  $R := (1/p^{2^i})(I - AB)$
  $B := B(I + Rp^{2^i})$

Example

$$
\begin{aligned}
777^{-1} &= 3 + 777^{-1}(-233)\cdot 10 \\
&= -6987 + 777^{-1}(74289)\cdot 10^2 \\
&= -37931731287 + 777^{-1}(2947295521)\cdot 10^4
\end{aligned}
$$

What happens if we omit the Rem?

Quadratic lifting: division free

$B := \mathrm{Rem}(A^{-1}, p)$
for $i = 0$ **to** $k-1$ **do**
  $R := (1/p^{2^i})(I - AB)$
  $B := B(I + Rp^{2^i})$

Example

$$
\begin{aligned}
777^{-1} &= 3 + 777^{-1}(-233)\cdot 10 \\
&= -6987 + 777^{-1}(74289)\cdot 10^2 \\
&= -37931731287 + 777^{-1}(2947295521)\cdot 10^4 \\
&= -111796021725954458700 1287 + 777^{-1}(8686550888106661441
\end{aligned}
$$

Division free quadratic lifting

Original version

$B := \mathrm{Rem}(A^{-1}, p)$
for $i = 0$ **to** $k-1$ **do**
$\quad R := (1/p^{2^i})(I - AB)$
$\quad B := B(I + Rp^{2^i})$

Optimization ideas:

- Apply loop unrolling and software pipelining.
- Avoid explicit computation of B.

Optimized version

$B := \mathrm{Rem}(A^{-1}, p)$
$R := (1/p)(I - AB)$
for $i = 0$ **to** $k-1$ **do**
$\quad R := R^2$

Straight line version of quadratic lifting

$$B := \mathrm{Rem}(A^{-1}, p)$$
$$R := (1/p)(I - AB)$$
$$A^{-1} = B(I + Rp)(1 + R^2p^2)(1 + R^4p^4)\cdots$$

Example

$$\begin{aligned} 777^{-1} &\equiv 3(1 \overbrace{-233}^{R} \cdot 10)(1 + \overbrace{74289}^{R^2} \cdot 10^2)(1 + \overbrace{2947295521}^{R^4} \cdot 10^4) \bmod 10^8 \\ &\equiv -1117960217259544587001287 \bmod 10^8 \\ &\equiv 12998713 \bmod 10^8 \end{aligned}$$

Question: How to alleviate the expression swell?

Answer: Interleave quadratic with linear lifting

Optimized version

$B := \text{Rem}(A^{-1}, p)$

$R := (1/p)(I - AB)$

for $i = 1$ **to** $k-1$ **do**

Loop invariant: $A^{-1} = * + A^{-1}R \cdot p^{2^{i-1}}$

$R := R^2$

Double-plus-one lifting

$C_0 := B := \text{Rem}(A^{-1}, p)$

$R := (1/p)(I - AB)$

for $i = 1$ **to** $k-1$ **do**

Loop invariant: $A^{-1} = * + A^{-1}R \cdot p^{2^{i}-1}$

$R := R^2$

$R := (1/p)(R - A\,\text{Rem}(C_0R, p))$

Example of double-plus-one lifting

Input: $A = 567$ and $p = 1000$.

Initialize:

$$567^{-1} = -97 + 567^{-1}(55) \cdot 1000$$

1. $1000 \to 1000^2 \to 1000^3$

$$\begin{aligned} 567^{-1} &= -5335097 + 567^{-1}(3025) \cdot 1000^2 \\ &= -430335097 + 567^{-1}(244) \cdot 1000^3 \end{aligned}$$

2. $1000^3 \to 1000^6 \to 1000^7$

$$\begin{aligned} 567^{-1} &= -105001763668430335097 + 567^{-1}(59536) \cdot 1000^6 \\ &= -970017636684303335097 + 567^{-1}(55) \cdot 1000^7 \end{aligned}$$

Implementation of double-plus-one lifting

Goals:

- reduce bulk of work reduced to level 3 BLAS
- reduce number of calls to GEMM (matrix×matrix multiply)

Key ideas:

- uses relatively prime lifting bases: $p = p_1 p_2 \ldots p_k$ and $q = q_1 q_2 \ldots q_l$
  $\rightarrow$ each $p_*, q_* < 2(n-1)2^{53}$
- make limited use of
  $\rightarrow$ IML (Integer Matrix Library) for inversion modulo a prime
  $\rightarrow$ GMP for large integer arithmetic

Precompute initial residue R in p-basis.
for $i = 0$ **to** $k-1$ **do**
  Compute $M := \mathrm{Rem}(C_0 \mathrm{Rem}(R, p)^2, p)$ in the p-basis.
  Use basis extension technques to obtain M in the q-basis.
  Compute $R := \mathrm{Rem}(p^{-1}(R^2 - AM, q)$ in the q-basis

Empirical results: high order-residue via double-plus-one lifting

- Intel 1.3 GHz Itanium2 with 192 GB RAM running GNU/Linux 2.4.21.
- gcc 4.1.2: linked against IML 1.0.3, ATLAS 3.6.0, and GMP 4.1.3.

Dimension	$\log_{10} \|\|A\|\|_\infty$	Time
1000	1	57 s
2000	1	454 s (≈ 7.6 minutes)
8000	1	41120 s (≈ 11.4 hours)
200	100	5 s
400	100	33 s
2000	100	4336 s (≈ 1.2 hours)

Empirical results: high order-residue via double-plus-one lifting

- Intel 1.3 GHz Itanium2 with 192 GB RAM running GNU/Linux 2.4.21.
- gcc 4.1.2: linked against IML 1.0.3, ATLAS 3.6.0, and GMP 4.1.3.

Dimension	$\log_{10} \|\|A\|\|_\infty$	Time
1000	1	57 s
2000	1	454 s (≈ 7.6 minutes)
8000	1	41120 s (≈ 11.4 hours)
200	100	5 s
400	100	33 s
2000	100	4336 s (≈ 1.2 hours)

Multi-core implementation:

Dimension	$\log_{10} \|\|A\|\|_\infty$	Time
2000	100	1073 s (≈ 18 minutes)