Modular multiplication and division algorithms based on continued fraction expansion

Mourad Gouicem

Intel Corporation

June 23, 2015

Problem

$$\text{Given } a, b, n \in \mathbb{N}, \text{ compute } \begin{cases} a \cdot b \mod n \\ a^{-1} \cdot b \mod n \end{cases}$$

- These operations are heavily used in number theory.
- And are frequently the most time consuming operation.

Problem

Gauss elimination of a matrix in $(\mathbb{Z}/n\mathbb{Z})^{d\times d}$ (n prime)

for $i = 0$ *to* $d-1$ **do**
  for $j = i+1$ *to* $d-1$ **do**
    $q = m[i,i]^{-1} \cdot m[j,i] \mod n$;
    for $k = i+1$ *to* $d-1$ **do**
      $m[j,k] = a[j,k] - q \cdot a[i,k] \mod n$

Problem

Gauss elimination of a matrix in $(\mathbb{Z}/n\mathbb{Z})^{d\times d}$ (n prime)

for $i = 0$ *to* $d-1$ **do**
  for $j = i+1$ *to* $d-1$ **do**
    $q = m[i,i]^{-1} \cdot m[j,i] \mod n$;
    for $k = i+1$ *to* $d-1$ **do**
      $m[j,k] = a[j,k] - q \cdot a[i,k] \mod n$

In this talk, we focus on division, but we provide a similar algorithm for multiplication in the paper.

Modular division

1. Compute modular inverse of a modulo n
2. Compute multiplication of a^{-1} by b modulo n

$\Rightarrow$ High latency due to the inverse computation.

Question

How do we integrate the modular multiplication with the modular inversion?

Modular inversion

The extended Euclidean algorithm (ExtGCD(a, n))

Let $a, n \in \mathbb{N}$ with $\mathrm{GCD}(a, n) = 1$.

$$\begin{array}{lll} \theta_{-1} = n & \theta_0 = a & \theta_{i+1} = \theta_{i-1} - k_{i+1}\theta_i \\ p_{-1} = 1 & p_0 = 0 & p_{i+1} = p_{i-1} + k_{i+1}p_i \\ q_{-1} = 0 & q_0 = 1 & q_{i+1} = q_{i-1} + k_{i+1}q_i \end{array}$$

$$\text{with} \begin{cases} k_{i+1} & = \lfloor \theta_{i-1}/\theta_i \rfloor, \\ \theta_i & = (-1)^i(q_i a - p_i n). \end{cases}$$

As a and n are coprime, the q_l cofactor corresponding to the last non-zero θ_l is a^{-1}.

Number systems based on the Euclidean algorithm

Ostrowski $(\theta_i)_{i\in\mathbb{N}}$ integer number system

Given $(\theta_i)_{i\in\mathbb{N}}$ from the ExtGCD(a, n) with a, n coprime integers and $a < n$, any integer $b < n$ can be uniquely written as

$$b = a + \sum_{i=1}^{l} b_i \theta_{i-1}$$

where $\begin{cases} 0 \leq b_1 \leq k_1 - 1, \\ 0 \leq b_i \leq k_i \text{ for } i \geq 2, \\ b_{i+1} = 0 \text{ if } b_i = k_i. \end{cases}$ (Markovian condition)

Number systems based on the Euclidean algorithm

Ostrowski $((-1)^i q_i)_{i\in\mathbb{N}}$ integer number system

Given $(q_i)_{i\in\mathbb{N}}$ from the ExtGCD(a, n) with a, n coprime integers and $a < n$, any integer $b < n$ can be uniquely written as

$$b = 1 + \sum_{i=1}^{l} b_i (-1)^{i-1} q_{i-1}$$

where $\begin{cases} 0 \leq b_1 \leq k_1 - 1, \\ 0 \leq b_i \leq k_i \text{ for } i \geq 2, \\ b_{i+1} = 0 \text{ if } b_i = k_i. \end{cases}$ (Markovian condition)

Ostrowski $(\theta_i)_{i\in\mathbb{N}}$ and $((-1)^i q_i)_{i\in\mathbb{N}}$ integer number systems share the same Markovian condition.

The proposed algorithm for modular division

Algorithm

1. Compute the sequences $(q_i)_{i\in\mathbb{N}}$ and $(\theta_i)_{i\in\mathbb{N}}$ from ExtGCD(a, n)
2. Compute the sequence $(b_i)_{i\in\mathbb{N}}$ such that
$$b = a + \sum_{i=1}^{l} b_i\theta_{i-1}$$
3. Return $1 + \sum_{i=1}^{l} b_i(-1)^{i-1}q_{i-1}$

$$b \quad = a + \sum_{i=1}^{l} b_i\theta_{i-1}$$

As $\theta_{i-1} = (-1)^{i-1}q_{i-1} \cdot a \mod n$. Then

$$\begin{aligned} b &= a + \sum_{i=1}^{l} b_i((-1)^{i-1}q_i \cdot a) && \mod n \\ &= a \cdot (1 + \sum_{i=1}^{l} b_i(-1)^{i-1}q_i) && \mod n \\ a^{-1} \cdot b &= 1 + \sum_{i=1}^{l} b_i((-1)^{i-1}q_i) && \mod n \end{aligned}$$

The proposed algorithm for modular division

Algorithm

1. Compute the sequences $(q_i)_{i\in\mathbb{N}}$ and $(\theta_i)_{i\in\mathbb{N}}$ from ExtGCD(a, n)
2. Compute the sequence $(b_i)_{i\in\mathbb{N}}$ such that
$$b = a + \sum_{i=1}^{l} b_i\theta_{i-1}$$
3. Return $1 + \sum_{i=1}^{l} b_i(-1)^{i-1}q_{i-1}$

$$b \quad = a + \sum_{i=1}^{l} b_i\theta_{i-1}$$

As $\theta_{i-1} = (-1)^{i-1}q_{i-1} \cdot a \mod n$. Then

$$\begin{aligned} b &= a + \sum_{i=1}^{l} b_i((-1)^{i-1}q_i \cdot a) && \mod n \\ &= a \cdot (1 + \sum_{i=1}^{l} b_i(-1)^{i-1}q_i) && \mod n \\ a^{-1} \cdot b &= 1 + \sum_{i=1}^{l} b_i((-1)^{i-1}q_i) \end{aligned}$$

Example 1: a=17, b=30, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	30		0
0		17	1	30		0

Example 1: a=17, b=30, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	30		0
0		17	1	30		0
1	2	11	2	13	1	1

Example 1: a=17, b=30, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	30		0
0		17	1	30		0
1	2	11	2	13	1	1
2	1	6	3	2	1	-1

Example 1: a=17, b=30, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	30		0
0		17	1	30		0
1	2	11	2	13	1	1
2	1	6	3	2	1	-1
3	1	5	5	2	0	-1

Example 1: a=17, b=30, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	30		0
0		17	1	30		0
1	2	11	2	13	1	1
2	1	6	3	2	1	-1
3	1	5	5	2	0	-1
4	1	1	8	2	0	-1

Example 1: a=17, b=30, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	30		0
0		17	1	30		0
1	2	11	2	13	1	1
2	1	6	3	2	1	-1
3	1	5	5	2	0	-1
4	1	1	8	2	0	-1
5	5	0	45	0	2	15

$$
\begin{aligned}
b &= 1 \cdot 17 + 1 \cdot 11 + 0 \cdot 6 + 0 \cdot 5 + 2 \cdot 1 = 30 && \bmod n \\
a^{-1} \cdot b &= 1 \cdot 1 - 1 \cdot 2 + 0 \cdot 3 - 0 \cdot 5 + 2 \cdot 8 && \bmod n \\
&= 15 && \bmod n
\end{aligned}
$$

Example 2: a=17, b=23, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	23		0
0		17	1	23		0

Example 2: a=17, b=23, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	23		0
0		17	1	23		0
1	2	11	2	6	1	1

Example 2: a=17, b=23, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	23		0
0		17	1	23		0
1	2	11	2	6	1	1
2	1	6	3	6	0	1

Example 2: a=17, b=23, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	23		0
0		17	1	23		0
1	2	11	2	6	1	1
2	1	6	3	6	0	1
3	1	5	5	0	1	4

Example 2: a=17, b=23, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	23		0
0		17	1	23		0
1	2	11	2	6	1	1
2	1	6	3	6	0	1
3	1	5	5	0	1	4
4	1	1	8			

Example 2: a=17, b=23, n=45

i	k_i	θ_i	q_i	${}_r b_i$	b_i	c
-1		45	0	23		0
0		17	1	23		0
1	2	11	2	6	1	1
2	1	6	3	6	0	1
3	1	5	5	0	1	4
4	1	1	8			
5	5	0	45			

$$
\begin{aligned}
b &= 1 \cdot 17 + 0 \cdot 11 + 1 \cdot 6 = 23 && \bmod n \\
a^{-1} \cdot b &= 1 \cdot 1 - 0 \cdot 2 + 1 \cdot 3 && \bmod n \\
&= 4 && \bmod n
\end{aligned}
$$

Complexity and pleasant facts

- Euclidean algorithm has quadratic binary complexity
- All partial results are of size $O(\log(n))$.
- Quotients k_i and b_i are very small on average (geomean ≈ 2.69)
  $\Rightarrow$ we can compute them by repeated subtractions
- First straightforward 64-bits implementation show an encouraging speedup of $2.44\times$ over GMP low-level functions.

Thank you for your attention.
Any questions?