1 Introduction

biblio ? checker Bernstein.

Notations

- $\mathsf{M}(n)$ polynomial arithmetic and $\mathsf{I}(n)$ multiprecision integer arithmetic.
- We denote resp. by $(a \text{ rem } p)$ and $(a \text{ quo } p)$ the remainder and quotient of the Euclidean division of $a \in \mathbb{Z}$ by $p \in \mathbb{N}$ where $0 \leqslant (a \operatorname{rem} b) < b$.
- For any $a = n / d$ with d coprime to p, we let $[a]_p$ be the unique representative in $\{0, \ldots, p-1\}$ of a modulo p.
- We call informally a pseudo-reduction of a modulo p the computation of b such that $a = b \bmod p$ and b "not too big" compared to p. In practice, we often will have that $b = \mathcal{O}(p^2)$.
- Say that our complexity model is bit complexity.
- Should we take β as a constant and simplify complexity ?
- Only give costs for the typical case $p_i \simeq \beta, s \simeq B, r \gg s$ because of $\beta^B < p_1 \cdots p_s$, which implies that $B < s$?

Bibliography Bibliography on reductions and pseudo-reductions. Recall Barett, Montgomery results ?

Cost of $a \bmod p$ when :

1. $\log(a) = \Theta(\log(p))$. **Cas classique, on en a vraiment besoin – $\mathcal{O}(\mathsf{I}(\log(p)))$?**
2. $\log(a) \gg \Theta(\log(p))$ **Servira à prouver l'algo naïf de multi-réduction – $\mathcal{O}(\log(a) / \log(p) \, \mathsf{I}(\log(p)))$?**
3. $\log(a / p) \ll \Theta(\log(p))$ **Sert à montrer que la finalisation des pseudos-réductions est peu couteuse**

   Polynomial analog suggests $\mathcal{O}(\mathsf{I}(c) \log(p) / c)$ where $c := \log(a / p)$. Maybe we don't need to be so specific

2 Conversions with Residue Number System

2.1 Residue Number System

2.2 Naive approach

In order to convert an integer a to a residue number system $(m_1, \ldots m_k)$, one can of course apply an Euclidean division of a by m_i for $i \in \{1, \ldots, k\}$.

In this section, we want to reduce the integers $n \in \mathbb{Z}$ modulo each of the positive integers $p_1, \ldots, p_s \in \mathbb{N}$. Let us assume that $p_1, \ldots, p_s < \beta$ and that $n_i < \beta^B$. In practice, β will be related to a certain number of machine words.

Algorithm 1

Input: $n = \sum_{j=0}^{B-1} c_j \beta^j, p$
Output: $n \bmod p$

Algo:

$c = c_{B-1}$
for $i = B - 2 \ldots 0$ do
  $r = c \, \beta \bmod \text{p}$
  $c = r + c_i$
return $c \bmod p$

The bit complexity for computing $n \bmod p$ is $\mathcal{O}(B \, \mathsf{I}(\log \beta))$. The conversion to the RNS thus costs $\mathcal{O}(s \, B \, \mathsf{I}(\log \beta))$. **$\mathcal{O}(s^2 \, \mathsf{I}(\log \beta))$**

Mention Barett/Montgomery optimizations ? Precomputation of floating number β / p ?

2.3 Quasi-linear approach

Classic binary tree approach. Precomputation of binary tree : $\mathcal{O}(\mathsf{I}(s \log \beta) \log s)$. [MCA, 3rd edition, Th 9.17]

Cost in typical case : $\mathcal{O}(\mathsf{I}(s \log \beta) \log s)$

3 Simultaneous RNS conversions

3.1 Straighforward

Advantage of simultaneous reductions with naive algorithm : can benefit from (SIMD) vectorized instructions. **$\mathcal{O}(r\, s^2\, \mathsf{I}(\log \beta))$**.

Straighforward simultaneous reduction using quasi-linear approach : $\mathcal{O}(r\ \mathsf{I}(s \log \beta) \log s)$. However, do not benefit from SIMD.

3.2 Linear Algebra

3.2.1 Linear algebra reductions

Simultaneous pseudo-reductions In this section, we want to simultaneously reduce the integers $n_1, \ldots, n_r \in \mathbb{Z}$ modulo each of the positive integers $p_1, \ldots, p_s \in \mathbb{N}$.

Let us assume that $p_1, \ldots, p_s < \beta$ and that $n_i < \beta^B$. In practice, β will be related to a certain number of machine words.

The first thing to do is to write the expansion in base β of

$$n_i = \sum_{j=0}^{B-1} c_{i,j}\, \beta^j$$

for $1 \leqslant i \leqslant r$. Let's precompute the values $r_{i,j} := \beta^i \operatorname{rem} p_j$ for $0 \leqslant i < B$ and $1 \leqslant j \leqslant s$.

Let $n_{i,\ell} := \sum_{j=0}^{B-1} c_{i,j}\, r_{j,\ell}$ then we have $n_i = n_{i,\ell} \bmod p_\ell$. The value $n_{i,\ell}$ is bounded by $B\, \beta^2$, whereas n_i was of size β^B. We say that $n_{i,\ell}$ is a pseudo-reduction of n_i modulo p_ℓ.

The values $n_{i,\ell}$ can be computed by linear algebra : $(n_{i,\ell}) \in \mathcal{M}_{r \times s}(k)$ is the product of $(c_{i,j}) \in \mathcal{M}_{r \times B}(k)$ and $(r_{j,\ell}) \in \mathcal{M}_{B \times s}(k)$.

Cost. In the case where we want to represent our integers in the RNS representation, we will choose $p_1, \ldots, p_s$ such that $\beta^B < p_1 \cdots p_s$, which implies that $B < s$. Then the matrix product to compute $(n_{i,\ell})$ can be done in bit complexity $\mathcal{O}(r / s \cdot s^\omega\ \mathsf{I}(\log(\beta))) = \mathcal{O}(r\ s^{\omega-1}\ \mathsf{I}(\log(\beta)))$. The precomputation of the residues $(r_{i,j})$ costs $\mathcal{O}(s^2\, \mathsf{I}(\log(\beta)))$.

Simultaneous reductions Now the cost of computing the remainder $(a \operatorname{rem} p)$ when $a < B\, \beta^2$ and $p < \beta$ is $\mathcal{O}(\mathsf{I}(\log(\beta\, B)))$. Therefore, our final step to compute our simultenous reductions costs $\mathcal{O}(r\, s\, \mathsf{I}(\log(\beta\, B)))$.

3.2.2 Linear algebra reconstructions

Hypothesis for the reconstruction : $p_1, \ldots, p_s$ are pairwise coprime.

Simultaneous pseudo-reconstructions Let $P = p_1 \cdots p_s$, $P_i = P / p_i$ for $1 \leqslant i \leqslant s$. Let $l_i := \sum_{j=1}^{s} n_{i,j} P_j\, [P_j^{-1}]_{p_j}$ so that $n_i = l_i \bmod P$ with $l_i < P\beta$. Then l_i are pseudo-reconstructions of $(n_{i,\ell})$ modulo $p_1, \ldots, p_s$.

Once again, we perform the computation of l_i using linear algebra. Let $P_j\ [P_j^{-1}]_{p_j} = \sum_{k=0}^{s-1} e_{j,k}\, \beta^k$ be the expansion in base β of $P_\ell\, [P_\ell^{-1}]_{p_\ell}$. Put together, we have

$$l_i := \sum_{j=1}^{s} n_{i,j} P_j\, [P_j^{-1}]_{p_j} = \sum_{j=1}^{s} n_{i,j} \sum_{k=0}^{s-1} e_{j,k}\, \beta^k = \sum_{k=0}^{s-1} \left(\sum_{j=1}^{s} n_{i,j} \cdot e_{j,k} \right) \beta^k.$$

Let $(d_{i,j}) \in \mathcal{M}_{r\times s}$ be the product of the matrices $(n_{i,j}) \in \mathcal{M}_{r\times s}$ and $(e_{j,k}) \in \mathcal{M}_{s\times s}$. Then $l_i = \sum_{k=0}^{s-1} d_{i,j}\,\beta^k$.

Note that $(d_{i,j})$ are not the exact coefficients of the β-expansion of l_i since $d_{i,j} \leqslant s\,\beta^2$. But the correction's cost is $\mathcal{O}(s\,\mathsf{I}(\log\beta))$.

say something about $d_{i,j}$ not being the β-expansion, but close.

Cost. The matrix product to compute $(d_{i,j})$ can be done in bit complexity $\mathcal{O}(r\,/\,s \cdot s^{\omega}\,\mathsf{I}(\log(\beta))) = \mathcal{O}(r\,s^{\omega-1}\,\mathsf{I}(\log(\beta)))$. Precomputation of $(e_{j,k})$ costs $\mathcal{O}(s\,\mathsf{I}(s\log\beta))$.

Simultaneous reconstructions The final step of the reconstruction consists in reducing l_i modulo P. This step is relatively cheap since l_i is almost reduced.

Using the reduction when $\log(a\,/\,p) \ll \Theta(\log(p))$ in our case $l_i = \mathcal{O}(s\,\beta^{s+1})$ and $P = \mathcal{O}(\beta^s)$, the last reduction step costs ??$\mathcal{O}(s\ \log(\beta)\ \mathsf{I}(\log(s\ b))\,/\,\log(s\ b))$** per l_i. Thus a total cost of $\tilde{\mathcal{O}}(r\,s\log\beta)$.

3.3 Hybrid approach ?

Linear algebra up to intermediate sizes. Asymptotic complexity (binary tree) equivalent (not even a change of the constant).

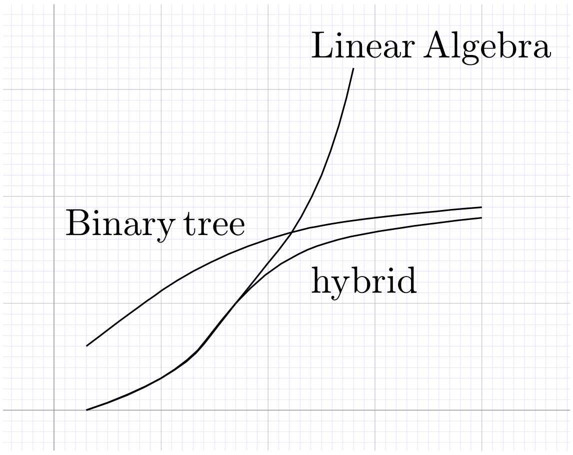

Figure 1.

4 Matrix Multiplication with multi-precision integer coefficients

5 Implementation

5.1 Reduction to word-size matrix multiplication

5.2 Kronecker substitution

5.2.1 From integer to β-adic

5.3 Linear storage for multi-modular matrix

6 Benchmarks

6.1 Conversion to and from the residue number system