Near Optimal Algorithms for Computing Smith Normal Forms of Integer Matrices

Department of Computer Science
University of Waterloo, Ontario, Canada, N2L 3G1
Technical Report CS-06-03

Arne Storjohann
astorjoh@daisy.uwaterloo.ca

Abstract

We present new algorithms for computing Smith normal forms of matrices over the integers and over the integers modulo \(d\). For the case of matrices over \(\mathbb{Z}_d\), we present an algorithm that computes the Smith form \(S\) of an \(A \in \mathbb{Z}^{n \times m}_d\) in only \(O(n^{k-1}m)\) operations from \(\mathbb{Z}_d\). Here, \(\theta\) is the exponent for matrix multiplication over rings: two \(n \times n\) matrices over a ring \(R\) can be multiplied in \(O(n^\theta)\) operations from \(R\). We apply our algorithm for matrices over \(\mathbb{Z}_d\) to get an algorithm for computing the Smith form \(S\) of an \(A \in \mathbb{Z}^{n \times m}\) in \(O(n^{\theta-1}m \cdot M(n \log \|A\|))\) bit operations (where \(\|A\| = \max \|A_{ij}\|\) and \(M(c)\) bounds the cost of multiplying two \([c]\)-bit integers). These complexity results improve significantly on the complexity of previously best known Smith form algorithms (both deterministic and probabilistic) which guarantee correctness.

1 Introduction

The Smith normal form is a canonical diagonal form for equivalence of matrices over a principal ideal ring \(R\). For any \(A \in \mathbb{R}^{n \times m}\) there exist unimodular (square and invertible) matrices \(U\) and \(V\) over \(R\) such that

\[
S = UAV = \begin{bmatrix}
  s_1 & & \\
  & \ddots & \\
  & & s_r \\
  & & \phantom{0}
\end{bmatrix}
\]

with each \(s_i\) nonzero and with \(s_i|s_{i+1}\) for \(1 \leq i \leq r - 1\). \(S\) is called the Smith normal form of \(A\) and the unimodular \(U\) and \(V\) are called transforming matrices. The nonzero diagonal entries \(s_i\) of \(S\) are called the invariant factors of \(A\) and are unique up to units — uniqueness of \(S\) can be ensured by specifying that each \(s_i\) belong to a prescribed complete set of nonassociates of \(R\). The Smith normal form was first proven to exist by Smith [9, 1868] for matrices over the integers (in this case, each \(s_i\) is positive, \(r = \text{rank}(A)\) and \(\det(U), \det(V) = \pm 1\).

In this paper we consider the problem of computing Smith normal forms of matrices with entries from \(\mathbb{Z}\) and \(\mathbb{Z}_d\), the ring of integers modulo \(d\). Computing Smith normal forms over these domains is useful in many applications, including Diophantine analysis (see Newman [8, 1972]) computing the structure of finitely generated abelian groups (see Haves, Holt & Rees [6, 1993]) and computing the structure of the class group of a number field (see Hafner & McCurley [4, 1989] and Buchmann [1, 1988]).

In Section 3 we present our main result — an asymptotically fast algorithm for computing Smith normal forms over \(\mathbb{Z}_d\). Let \(A\) be an \(n \times m\) matrix over \(\mathbb{Z}_d\). We assume without loss of generality that \(n \leq m\) — the Smith normal form of the transpose of \(A\) will have the same invariant factors as that of \(A\). Our algorithm requires a near optimal \(O(n^{\theta-1}m)\) operations from \(\mathbb{Z}_d\) to compute the Smith normal form \(S\) of \(A\). Here, \(\theta\) is defined so that two \(n \times n\) matrices over a ring \(R\) can be multiplied in \(O(n^\theta)\) operations from \(R\). Using standard matrix multiplication \(\theta = 3\), while the best known algorithm of Coppersmith & Winograd [2, 1990] allows \(\theta = 2.38\). For the case \(n = m\), our complexity result for computing the Smith normal form matches that of the best known algorithm to compute \(\det(A)\) — which can be computed (up to a unit) as the product of the diagonal entries in \(S\). Although we do not prove it here, we remark that candidates for transforming matrices \(U\) and \(V\) can be recovered in \(O(n^{\theta-1}m)\) operations from \(\mathbb{Z}_d\). The asymptotically fast algorithm for computing transforming matrices over \(\mathbb{Z}_d\) is based on the approach we present here, but requires in addition a number of new results and will be the subject of a future paper.

In Section 4 we consider the problem of computing Smith normal forms of integer matrices. Let \(A\) be an \(n \times m\) input matrix over \(\mathbb{Z}\). We show how to apply the result of Section 3 to get an algorithm that requires \(O(n^{\theta-1}mM(n \log \|A\|))\) bit operations to produce the Smith normal form \(S\) of \(A\). The previously best deterministic algorithm of Hafner & McCurley [5, 1991] requires \(O(n^\theta m \log \|A\|M(n \log \|A\|))\) bit operations to produce \(S\); we have improved this worst case complexity bound by a factor of at least \(O(n \log \|A\|)\) bit operations — even assuming standard integer and matrix multiplication. The previously best Las Vegas probabilistic algorithm of Giesbrecht [3, 1996] computes \(S\) in an expected number of \(O(n^\theta mM(n \log \|A\|))\) bit operations.

The algorithm that we have presented for computing Smith normal forms over \(\mathbb{Z}\) does not compute unimodular transforming matrices \(U\) and \(V\) that satisfy \(UAV = S\). Since the transforming matrices are highly nonunique, the
goal is to produce candidates for $U$ and $V$ that have small entries. Heuristic methods have shown promising results, especially for large sparse input matrices with small entries (see Havas, Holt and Rees [6, 1993]), but are difficult to analyse. In the future, we will present deterministic algorithms that compute multiplier matrices $U$ and $V$. We mention one result: for a square nonsingular matrix $A$, there exists a candidate for $V$ that has total size (the sum of the bit lengths of the individual entries of $V$) bounded by $O'\left(n^2\log |A|\right)$ bits — this is on the same order of space as required to write down $A$.

2 Preliminaries and Previous Results

Two matrices $A$ and $B$ over a principal ideal ring $R$ are said to be equivalent if $B$ is related to $A$ via unimodular transformations $U$ and $V$, that is, with $B = UAV$ and $A = U^{-1}BV^{-1}$. It follows that two matrices $A$ and $B$ have the same Smith normal form if and only if they are equivalent. Recall that an $S = \text{diag}(s_1, s_2, \ldots, s_r, 0, \ldots, 0) \in R^{n \times m}$ is in Smith normal form if $s_i | s_{i+1}$ for $i = 1, 2, \ldots, r - 1$ and each $s_i$ belongs to a prescribed complete set of nonassociates of $R$. For the case $R = \mathbb{Z}_d$, we choose our prescribed set of nonassociates to be $N_d^* = \{x \mod d : x \in \mathbb{Z}, 0 < x \leq d, x \neq 0\}$, and for $a, b \in \mathbb{Z}_d$, write $\gcd_d(a, b)$ to denote the unique principal ideal generator of the ideal $(a, b) \subseteq \mathbb{Z}_d$ which belongs to $N_d^*$. Note that $\gcd_d(a, b)$ can be computed as $\gcd(\tilde{a}, \tilde{b}) \mod d$ where $\tilde{a}$ and $\tilde{b}$ are in $\mathbb{Z}$ with $\tilde{a} = a \mod d$ and $\tilde{b} = b \mod d$. For the case $a, b = 0$, we have $\gcd_d(0, 0) = 0$. Over the ring $R = \mathbb{Z}$, our prescribed complete set of nonassociates is simply $N^* = \{x : x \in \mathbb{Z}, x \neq 0\}$.

We present some of our complexity results in terms of the number of operations from $\mathbb{Z}_d$. Given $a, b \in \mathbb{Z}_d$, we consider a single operation from $\mathbb{Z}_d$ to be one of: (1) finding $a + b, a - b, ab \in \mathbb{Z}_d$; (2) if $a$ divides $b$, finding a $q \in \mathbb{Z}_d$ with $aq = b$; (3) finding elements $g, s, t, u, v \in \mathbb{Z}_d$ such that

$$\begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix}$$

with $g = \gcd_d(a, b)$ and $su - tv$ a unit in $\mathbb{Z}_d$. Let $B(\log d)$ be a function which bounds the number of bit operations required to perform a single operation from $\mathbb{Z}_d$. Using standard integer arithmetical, $B(\log d) \leq \log^2 d$, while fast integer arithmetical allows

$$B(\log d) \ll M(\log d) \log \log d.$$ 

In Section 4 we use the fact that $B(\log d)$ bounds the number of bit operations required to apply the Chinese remainder algorithm with moduli whose product has magnitude less than $d$.

Our work on this particular topic (asymptotically fast algorithms for diagonalizing matrices over rings) was motivated in part by the work of Hafner & McCurley in [5, 1991] where they give asymptotically fast algorithms for triangularizing matrices over rings. Theorem 1, which follows from their work, gives a key subroutine which we require.

**Theorem 1** (Hafner & McCurley [5, 1991]) There exists a deterministic algorithm that takes as input an $n \times m$ matrix $A$ over $\mathbb{Z}_d$, and produces as output two matrices $V$ and $T$ satisfying $AV = TA$, with $T$ lower triangular and $V$ unimodular. If $A$ has last $t$ columns zero, then $V$ can be written as

$$V = \begin{bmatrix} V_1 & 0 \\ 0 & I_t \end{bmatrix}.$$ 

If $nm \leq b$, then the cost of the algorithm is bounded by $O(b^n)$ operations from $\mathbb{Z}_d$.

3 Smith Normal Form over $\mathbb{Z}_d$

In this section we develop an asymptotically fast algorithm to compute the Smith normal form of an $A \in \mathbb{Z}_d^{n \times m}$. Our approach is to compute a succession of matrices $A = A_0, A_1, \ldots, A_{k} = D$ with $A_i$ equivalent to $A_{i-1}$ for $i = 1, 2, \ldots, k$, and with $D$ a diagonal matrix. The Smith normal form of $A$ can then be found quickly by computing the Smith normal form of the diagonal matrix $D$.

Our algorithm depends on a number of subroutines, two of which we present separately in Subsection 3.1 and 3.2. In Subsection 3.1 we present an algorithm that requires $O(n^3)$ operations from $\mathbb{Z}_d$ to transforms an upper triangular $B \in \mathbb{Z}_d^{n \times n}$ to an equivalent bidiagonal matrix $C$. In Subsection 3.2 we show how to compute the Smith normal form of a bidiagonal $C \in \mathbb{Z}_d^{n \times n}$ in $O(n^3)$ operations from $\mathbb{Z}_d$. In Subsection 3.3 we combine these results and give an algorithm that requires $O(n^3 - m)$ operations from $\mathbb{Z}_d$ to computing the Smith normal form of an $A$ in $\mathbb{Z}_d^{n \times m}$.

3.1 Reduction of Banded Matrices

A square matrix $A$ is upper $b$-banded if $A_{ij} = 0$ for $j < i$ and $j \geq i + b$, that is, if $A$ can be written as

$$A = \begin{bmatrix} * & * & \cdots & \cdots & \cdots \\ * & * & \cdots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \quad (1)$$

The main purpose of this subsection is to develop an algorithm which transforms $A$ to an equivalent matrix, also upper banded, but with band about half the width of the band of the input matrix. Our result is the following.

**Theorem 2** For $b > 2$, there exists a deterministic algorithm that takes as input an $n \times n$ upper $b$-banded matrix $A$ over $\mathbb{Z}_d$, and produces as output an equivalent $n \times n$ upper $\left(\lfloor b/2 \rfloor + 1\right)$-banded matrix $A'$. If $A$ has last $t$ columns zero, then $A'$ will have last $t$ columns zero. The cost of the algorithm is $O(n^2 b^n)$ operations from $\mathbb{Z}_d$.

Proof By augmenting $A$ with at most $b$ rows and columns of zeroes we may assume that $t \geq 2b$, that is, that $A$ has at least $2b$ trailing columns of zeroes. In what follows, we let write $\text{sub}[i, k] = \text{sub}[i, k]$ to denote the the symmetric $k \times k$ submatrix of $A$ comprised of rows and columns $i + 1, \ldots, i + k$.\]
Our work matrix, initially the input matrix $A$, has the form

\[
B = \begin{bmatrix}
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
* & * & * & * & * & * & * & * & * & * \\
\end{bmatrix}
\]

Our approach is to transform $A$ to $A'$ by applying (in place) a sequence of equivalence transformations to sub$[[s_1, r_1], \text{sub}[[i+1, s_1 + j s_2, r_2]$], where $i$ and $j$ are nonnegative integer parameters and

\[
s_1 &= \lfloor b/2 \rfloor, \\
r_1 &= b - 1, \\
s_2 &= b - 1, \\
r_2 &= 2(b - 1),
\]

The first step is to convert the work matrix to an equivalent matrix but with first $s_1$ rows in correct form. This transformation is accomplished using subroutine $\text{Triang}$, defined below by Lemma 3.

**Lemma 3** For $b > 2$, there exists a deterministic algorithm $\text{Triang}$ that takes as input an $r_1 \times r_1$ upper $b$-banded matrix and produces as output an equivalent matrix

\[
B' = \begin{bmatrix}
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
* & * & * & * & * & * & * & * & * & * \\
\end{bmatrix}
\]

if $B$ has last $t$ columns zero, then $B'$ will have last $t$ columns zero. The cost of the algorithm is $O(b^3)$ operations from $\mathbb{Z}_d$.

**Proof.** Using the algorithm of Theorem 1, compute an $s_2 \times s_2$ unimodular matrix $V$ which, upon post-multiplication, triangularizes the $s_1 \times s_2$ upper right hand block of $B$, and set

\[
B' = B \left[ I_{s_1}, V \right]
\]

Since $r_1 < 2b$, the cost is as stated.

Apply subroutine $\text{Triang}$ to sub$[[0, r_1]$ of our initial work matrix to effect the following transformation:

At this stage we can write the work matrix as

\[
C = \begin{bmatrix}
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
* & * & * & * & * & * & * & * & * & * \\
\end{bmatrix}
\]

where the focus of attention is now sub$[[s_1, r_2]$. Subsequent transformations will be limited to rows $s_1 + 1, s_1 + 2, \ldots, n-t$ and columns $s_1 + s_2 + 1, s_1 + s_2 + 2, \ldots, n-t$. The next step is to transform the work matrix back to an upper $b$-banded matrix. This is accomplished using subroutine $\text{Shift}$, defined below by Lemma 4.

**Lemma 4** For $b > 2$, there exists a deterministic algorithm $\text{Shift}$ that takes as input an $r_2 \times r_2$ matrix

\[
C = \begin{bmatrix}
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * & * \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
* & * & * & * & * & * & * & * & * & * \\
\end{bmatrix}
\]
over \( \mathbb{Z}_d \), where each block is \( s_2 \times s_2 \), and produces as output an equivalent matrix

\[
C' = \begin{bmatrix}
* & \cdots & * \\
\vdots & \ddots & \vdots \\
* & \cdots & *
\end{bmatrix}
\]

If \( C \) has last \( t \) columns and rows zero, then \( C' \) will have last \( t \) columns and rows zero. The cost of the algorithm is \( O(b^3) \) operations from \( \mathbb{Z}_d \).

**Proof.** Write the input matrix as

\[
C = \begin{bmatrix}
C_1 & C_2 \\
C_3 & C_5
\end{bmatrix}
\]

where each block is \( s_2 \times s_2 \). Use the algorithm of Theorem 1 to compute, in succession, a unimodular matrix \( U^T \) such that \( C_1^T U^T \) is lower triangular, and then a unimodular matrix \( V \) such that \( (UC_2) V \) is lower triangular. Set

\[
C' = \begin{bmatrix}
U \\
I_{s_2}
\end{bmatrix}
\begin{bmatrix}
C_1 & C_2 \\
C_3 & C_5
\end{bmatrix}
\begin{bmatrix}
I_{s_2} \\
V
\end{bmatrix}
\]

Since \( v_2 < 2b \), the cost is as stated.

The procedure just described is now recursively applied to the trailing \( (n - s_1) \times (n - s_1) \) submatrix of the work matrix, itself an upper \( b \)-banded matrix. For example, the next step is to apply subroutine **Triang** to submatrix \( s_1 \times n_1 \) to get the following sequence of transformations.

We get the following.

**Algorithm:** **BandReduction**

**Input:** An upper \( b \)-banded matrix \( A \in \mathbb{Z}^{n \times n} \) with \( b > 2 \) and last \( t \) columns zero. Note: We assume that \( t \geq 2b \). If
not, then augment $A$ with $2b-t$ rows and columns of zeros.

**Output:** An upper $([b]/2+1)$-banded matrix that is equivalent to $A$ and has last $t$ columns zero.

1. **[Initialize]**
   
   $s_1 \leftarrow [b]/2$;
   
   $n_1 \leftarrow [b]/2 + b - 1$;
   
   $s_2 \leftarrow b - 1$;
   
   $n_2 \leftarrow 2(b - 1)$;

2. **[Apply equivalence transformations]**
   
   for $i = 0$ to $([t-n]/s_1) - 1$ do,
   
   apply Triang to sub$_{s_1[i], s_1[i]}$;
   
   for $j = 0$ to $([t-n-(i+1)/s_2]) - 1$ do,
   
   apply Shift to sub$_{s_2[j], s_2[j]}$;

Let $T(n,b)$ be the cost of applying algorithm BandReduction to an $n \times n$ upper $b$-banded input matrix.

To prove the correctness of applying BandReduction, an $n \times n$ upper $b$-banded input matrix.

For all $i, j$, $1 \leq i, j \leq b$, the inner loop in step (2) is

$$L_i = \left[ (n - t)/s_1 \right] < \frac{2n}{b-1},$$

while the number of iterations, for any fixed value of $i$, of the inner loop in step (2) is

$$L_j = \left[ (n - t - (i+1)/s_2) \right] < \frac{n}{b-1}. $$

The number of applications of either subroutine Triang or Shift occurring during algorithm BandReduction is seen to be bounded by $L_i(1 + L_j)$. By Lemmas 3 and 4, we have

$$T(n,b) < L_i(1 + L_j)cb^b$$

for some absolute constant $c$. Substituting (2) and (3) into (4) yields

$$T(n,b) < \frac{2n}{b-1}(1 + \frac{n}{b-1})cb^b$$

$$< \frac{2n}{b-1}(\frac{2n}{b-1})\left(\frac{b}{2}\right)^b$$

which completes the proof.

**Corollary 5** There exists a deterministic algorithm that takes as input an $n \times n$ upper triangular matrix $A$ over $\mathbb{Z}_q$, and produces as output an upper 2-banded matrix $\hat{A}$ that is equivalent to $A$. The cost of the algorithm is $O(n^2)$ operations from $\mathbb{Z}_q$.

**Proof** By augmenting $A$ with at most $n$ rows and columns of zeros, we can assume that $n = 2^k + 1$ for some $k \in \mathbb{Z}$. We consider $A$ as an $n \times n$ upper $b$-banded matrix with $b = n/2$.

Let $D(n,b)$ be the cost of computing an upper 2-banded matrix equivalent to an $n \times n$ upper $b$-banded input matrix. It follows from Theorem 2 that

$$D(n,n) \leq D(n, [b]/2) + 1 + cn^2(b-1)^{b-2}$$

for some absolute constant $c$. Replace $b$ with $n$ in (5) and iterate to obtain

$$D(n,n) \leq D(n, [n]/2) + 1 + cn^2(n-1)^{n-2}$$

which completes the proof.

3.2 The Smith Normal Form of a Bidagonal Matrix

A square matrix $A$ is upper bidiagonal if $A_{ij} = 0$ for $j < i$ and $j > i + 1$, that is, if $A$ can be written as

$$A = \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & *
\end{bmatrix}.$$ 

In particular, $A$ is upper bidiagonal if $A$ is upper 2-banded and vice versa. Our result is the following.

**Theorem 6** There exists a deterministic algorithm that takes as input an upper bidiagonal matrix $A \in \mathbb{Z}_q^{n \times n}$ and produces as output the Smith normal form of $A$. The cost of the algorithm is $O(n^2)$ operations from $\mathbb{Z}_q$.

We require some intermediate results before proving Theorem 6.

**Lemma 7** Let $a,b$ be elements of $\mathbb{Z}_q$. There exists elements $x$ and $u$ of $\mathbb{Z}_q$, such that $xa + b = u \gcd_a(a,b)$.

**Proof** Follows from the fact that $\mathbb{Z}_q$ is a stable ring.

**Lemma 8** Let

$$A = \begin{bmatrix}
a & b \\
* & d \\
* & c
\end{bmatrix}$$

be over $\mathbb{Z}_q$, with $d$ a multiple of $b$. If $q_i$ is a solution to $a = q_1 \gcd_a(a,b)$, and $q_2$ is a solution to $\gcd_d(a,b) = q_2 \gcd_d(b,c), then A is equivalent to

$$\hat{A} = \begin{bmatrix}
\hat{a} & \hat{e} \\
* & \hat{d} \\
\hat{c} & \hat{e}
\end{bmatrix} \quad \text{where} \quad \hat{a} = \gcd_d(a,b,c), \quad \hat{d} = q_1 d, \quad \hat{c} = q_2 c, \quad \hat{e} = q_2 e.$$

**Proof** We show that $A$ can be transformed to $\hat{A}$ via a sequence of unimodular row and column transformations. To begin, let $x_i$ and $u_i$ be elements of $\mathbb{Z}_q$, with $u_i$ a unit, such
that \(x_1a + b = u_1\) \(\gcd_j(a,b)\). (We only require the existence of \(x_1\) and \(u_1\), as per Lemma 7, we don’t need to produce \(x_1\) and \(u_1\) explicitly.) Add \(x_1\) times column 1 of \(A\) to column 3 and then switch columns 1 and 3 to obtain the equivalent matrix

\[
A_1 = \begin{bmatrix}
g_1 & a \\
d & c \\
e & e
\end{bmatrix}
\]

where \(g_1 = u_1 \gcd_j(a,b)\). To zero out the entry in row 1 column 3 of \(A_1\), multiply column 3 of \(A_1\) by \(-u_1\) (a unit) and then add \(q_1\) times column 1 of \(A_1\) to column 3 to obtain the equivalent matrix

\[
A_2 = \begin{bmatrix}
g_1 & q_1d \\
d & q_1c \\
e & e
\end{bmatrix}.
\]

Since \(g_1\) is an associate of \(\gcd_j(a,b)\), and \(b\) divides \(d\), we can add a multiple of row 1 of \(A_2\) to row 2 to obtain the equivalent matrix

\[
A_3 = \begin{bmatrix}
g_1 & q_1d \\
* & q_1c \\
e & e
\end{bmatrix}.
\]

The second stage of the reduction is similar to the first. Let \(x_2\) and \(u_2\) be elements of \(\mathbb{Z}_d\), with \(u_2\) a unit, such that \(x_2g_1 + c = u_2 \gcd_j(g_1,c)\), and add \(x_2\) times the first row of \(A_3\) to row 3 and then switch rows 1 and 2 to obtain the equivalent matrix

\[
A_4 = \begin{bmatrix}
g_2 & q_1c \\
g_1 & q_1d \\
e & e
\end{bmatrix}.
\]

where \(g_2 = u_2 \gcd_j(g_1,c)\). To zero out the entry in row 3 column 1 of \(A_4\), multiply row 3 of \(A_4\) by \(-u_2v_1^{-1}\) (a unit) and then add \(q_2\) times row 1 of \(A_4\) to row 3 to obtain the equivalent matrix

\[
A_5 = \begin{bmatrix}
g_2 & q_1c \\
* & q_1d \\
e & e
\end{bmatrix}.
\]

To complete the transformation to \(\hat{A}\), transform the entry in row 1 column 1 to \(\gcd_j(a,b,c)\) by multiplying column 1 of \(A_5\) by a unit, then zero out the entry in row 1 column 3 by adding a multiple of column 1 to column 3.

**Corollary 9** There exists a deterministic algorithm that takes as input a \(3 \times 4\) matrix

\[
A = \begin{bmatrix}
a & b \\
d & c \\
e & e
\end{bmatrix}
\]

over \(\mathbb{Z}_d\), with \(d\) a multiple of \(b\), and produces as output an equivalent matrix that can be written as

\[
\hat{A} = \begin{bmatrix}
\hat{a} & e \\
\hat{d} & \hat{e}
\end{bmatrix}
\]

with \(\hat{e}\) a multiple of \(e\), and \(\hat{a}\) a divisor of both \(\hat{e}\) and \(\hat{d}\).

Furthermore, the matrix \(\hat{A}\) produced is equivalent to \(A\) under a sequence of unimodular row and column transformations limited to columns 1 and 3. The cost of the algorithm is \(O(1)\) operations from \(\mathbb{Z}_d\).

**Proof** Find solutions \(q_1\) and \(q_2\) to \(a = q_1 \gcd_j(a,b)\) and \(\gcd_j(a,b,c) = q_2 \gcd_j(a,b,c,d)\), then compute \(\hat{a}, \hat{d}, \hat{c}\) and \(\hat{e}\) according to the definitions in Lemma 8.

For our next result, we need some notation. For \(2 < k < n\) denote by \(T_k^n\) the set of all \(n \times n\) matrices over \(\mathbb{Z}_d\) which are upper bidiagonal except with the entry in row 1 column 2 zero and with the entry in row 1 column \(k\) possibly nonzero but dividing the entry in row \(k - 1\) column \(k\) — that is, matrices which can be written using a block decomposition as

\[
\begin{bmatrix}
a & b \\
d & e
\end{bmatrix},
\]

where \(b\) is in column \(k\) and divides \(d\).

**Lemma 10** There exists a deterministic algorithm that takes as input a matrix \(T\) over \(\mathbb{Z}_d\) and in \(T_k^n\) with \(2 < k < n\), and produces as output an equivalent matrix \(\hat{T}\) in \(T_{k+1}^n\). Furthermore, if \(T_{(1)}\) divides all entries in the first \(k - 1\) columns of \(T\), then \(T_{(1)}\) divides all entries in the first \(k\) columns of \(\hat{T}\). The cost of algorithm is \(O(1)\) operations from \(\mathbb{Z}_d\).

**Proof** Let \(T\) be written as in (7). The construction of Corollary 9 can be applied to the \(3 \times 4\) submatrix of \(T\) comprised of rows 1, \(k - 1\), \(k\) and columns 1, \(k - 1\), \(k\) at a cost of \(O(1)\) operations from \(\mathbb{Z}_d\) to produce the equivalent matrix

\[
\begin{bmatrix}
\hat{a} & * \\
\hat{d} & \hat{e}
\end{bmatrix}
\]

in \(T_{k+1}^n\). To prove the second part of the theorem, note that by Corollary 9 we have \(\hat{a} = \gcd_j(a,b,c,d)\), and in particular, \(\hat{a}d\). Thus, if \(a\) divides all entries in the first \(k - 1\) columns of \(T\), then \(\hat{a}\) divides all entries in the first \(k\) columns of \(\hat{T}\).

We now return to the proof of Theorem 6. Let \(R(n)\) be the number of operations required to compute the Smith normal form of an \(n \times n\) upper bidiagonal matrix over \(\mathbb{Z}_d\). We claim that

\[
R(n) \leq R(n - 1) + cn
\]

for some absolute constant \(c\). To prove (8), let \(A\) be an \(n \times n\) upper bidiagonal matrix over \(\mathbb{Z}_d\). We show how to produce
a matrix
\[ B = \begin{bmatrix}
  g & * & * \\
  * & * & * \\
  * & * & * \\
  \ddots & * & * \\
  * & & \\
  0 & & & \\
  & & & \\
  & & & \\
  & & & \\
  0 & & & \\
\end{bmatrix} \] (9)
which is equivalent to \( A \) and where \( g \) is the gcd of all entries in \( B \). The Smith normal form of \( A \) can now be found by computing recursively the Smith normal form of the trailing \((n-1) \times (n-1)\) submatrix of \( B \).

To begin, convert \( A \) to the \((n+2) \times (n+2)\) matrix
\[ \tilde{A}_3 = \begin{bmatrix}
  * & * & 0 & \cdots & * \\
  * & * & * & \ddots & * \\
  * & * & * & \ddots & * \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  * & * & * & \cdots & * \\
  0 & & & & \\
  & & & & \\
  & & & & \\
  & & & & \\
  & & & & \\
\end{bmatrix} \]
by inserting a row and column of zeros after the pivot entry and by augmenting with a single row and column of zeros. The Smith normal form of \( \tilde{A}_3 \) will have the same invariant factors as the Smith normal form of \( A \). Furthermore, \( \tilde{A}_3 \) is in \( T_{n+2} \) and the entry in row 1 column 1 of \( \tilde{A}_3 \) divides all entries in the first two columns of \( \tilde{A}_3 \). Starting with \( \tilde{A}_3 \), apply the algorithm of Lemma 10 for \( k = 3,4,\ldots,n+1 \) to compute a succession of equivalent matrices \( \tilde{A}_4, \tilde{A}_5, \ldots, \tilde{A}_{n+2} \), with \( \tilde{A}_1 \in T_{n+2} \). By Lemma 10, the cost of this is \( O(n) \) operations from \( Z_d \) and, since the last column of \( \tilde{A}_3 \) is all zero, \( \tilde{A}_{n+2} \) will have the form
\[ \tilde{A}_{n+2} = \begin{bmatrix}
  g & 0 & * & \cdots & * \\
  0 & * & * & \ddots & * \\
  * & * & * & \ddots & * \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  * & * & * & \cdots & * \\
  0 & & & & \\
  & & & & \\
  & & & & \\
  & & & & \\
  & & & & \\
\end{bmatrix} \]
where \( g \) divides all entries in the first \( k+1 \) columns of \( \tilde{A}_{n+2} \).
Finally, delete rows and columns 2 and \( n+2 \) of \( \tilde{A}_{n+2} \) (which contain only zero entries) to produce an \( n \times n \) matrix equivalent to \( A \) and which can be written as in (9). This proves the inequality (8). To complete the proof of Theorem 6, iterate (8) to obtain
\[ R(n) \leq R(n-1) + cn \]
\[ = R(0) + c \sum_{i=1}^{n} i \]
\[ \ll n^2 \]

3.3 The Smith Normal Form Algorithm

Theorem 11 There exists a deterministic algorithm that takes as input an \( n \times n \) matrix \( A \) over \( Z_d \), and produces as output the Smith normal form of \( A \). The cost of the algorithm is \( O(n^2 \log n) \) operations from \( Z_d \).

Proof By augmenting \( A \) with at most \( n-1 \) columns, we can assume that \( m = \ln n \) for some integer \( k \). The algorithm consists of three steps. First, find an \( n \times n \) upper triangular matrix \( \tilde{B} \) that has the same invariant factors as \( A \). This can be accomplished in \( O(n^2 \log n) \) operations from \( Z_d \) as follows. Find a lower triangular matrix \( T \) that is equivalent to \( A \) by applying the triangularization algorithm of Theorem 1, in succession for \( i = k-2, k-3, \ldots, 0 \), to the \( n \times 2n \) submatrix of \( A \) comprised of columns \( i+1, i+2, \ldots, (i+2)n \). Take \( B \) to be the transpose of the principal \( n \times n \) submatrix of \( T \). For the second step, apply the algorithm of Corollary 5 to transform \( B \) to an equivalent upper triangular matrix \( C \). Finally, apply the algorithm of Theorem 6 to transform \( C \) to Smith normal form \( S \), which will have the same diagonal entries as the Smith normal form of \( A \). By Corollary 5 and Theorem 6, each of these steps is bounded by \( O(n^2 \log n) \) operations from \( Z_d \).

4 Smith Normal Form over \( Z \)

In this section we show how to use the algorithm for Smith normal form over \( Z_d \) presented in section 3 to get an asymptotically fast algorithm for computing Smith normal forms over \( Z \). We follow the approach of many previous algorithms and compute over \( Z_d \), where \( d \) is chosen to be a positive multiple of the product invariant factors of \( A \) (see Hafner & McCurley [5, 1991]). To make this idea precise, we define homomorphisms \( \phi = \phi_d \) and \( \overline{\phi}^- = \phi_d^- \) which we use to move between the two domains \( Z \) and \( Z_d \). Define \( \phi : Z \to Z_d \) by \( \phi(a) = \overline{a} \) where \( \overline{a} = a \mod d \). Define the pullback homomorphism \( \overline{\phi}^- : Z_d \to Z \) by \( \overline{\phi}^-(\overline{a}) = \overline{a} \) where \( \overline{a} = a \mod d \) and \( 0 < a < d \). For the following theorem, we denote by \( \text{snf}(X) \) the Smith normal form of an input matrix \( X \) over the domain of entries of \( X \) (either \( X \) is over \( Z \) or \( X \) is over \( Z_d \)). We also write \( \phi(A) \) to denote the matrix obtained by applying \( \phi \) to each entry of \( A \).

Theorem 12 Let \( A \) be a matrix over \( Z \). If \( d = 2d' \) where \( d' \) is a positive multiple of the product of the invariant factors of \( A \), then
\[ \text{snf}(A) = \overline{\phi}^-(\text{snf}(\phi(A))) \]

Proof Let \( \text{snf}(A) = \text{diag}(s_1, s_2, \ldots, s_\tau, 0, \ldots, 0) \). Each \( s_i \) satisfies \( 1 \leq s_i \leq d' < d \), so we have \( s_i = \overline{\phi}^-(\phi(s_i)) \) for \( 1 \leq i \leq \tau \) and
\[ \text{snf}(A) = \overline{\phi}^-(\text{snf}(\phi(A))) \] (10)

Next, let \( U \) and \( V \) be unimodular matrices over \( Z \) such that \( UAV = \text{snf}(A) \). Then
\[ \phi(U) \phi(A) \phi(V) = \phi(\text{snf}(A)) \]
where \( \phi(U) \) and \( \phi(V) \) are unimodular over \( Z_d \). It is easily verified that \( \phi(\text{snf}(A)) \) is in Smith normal form over \( Z_d \). In particular, \( \phi(s_i) \) divides \( \phi(s_{i+1}) \) for \( 1 \leq i < \tau - 1 \) and \( \phi(s_\tau) \in N_d^+ \) for \( 1 \leq i < \tau \). Since the Smith normal form of \( \phi(A) \) is unique, we must have
\[ \phi(\text{snf}(A)) = \text{snf}(\phi(A)) \] (11)
The desired result follows by substituting (10) into (11).

Lemma 13 There exists a deterministic algorithm that takes as input an \( n \times n \) matrix \( A \) over \( Z \), and produces as output the determinant \( \det \) of a nonsingular maximal rank minor of \( A \). The cost of the algorithm is \( O(n^2 \log n) \) bit operations.
Proof We apply the standard homomorphic imaging scheme. Compute a number \( z \) such that \( \Pi_p \leq z \) p \( \leq n^{1/2} ||A||^n \).

By Hadamard's bound every minor of \( A \) has magnitude bounded by \( b \). Next, find a maximal rank nonsingular submatrix \( A^* \) of \( A \). This can be accomplished using an algorithm of Ibarrá, Moran \& Hui [7, 1982] to compute the rank of \( A \) over \( \mathbb{Z}_p \) for each prime \( p \leq z \), since their algorithm returns also a maximal set of linearly independent rows and columns of \( A \) over \( \mathbb{Z}_p \). The cost of their algorithm for a single prime \( p \) is \( O(n^{r-1}m) \) operations form \( \mathbb{Z}_p \). Compute \( \det(A^*) \mod p \) for each prime \( p \leq z \), again using the algorithm [7, 1982], and reconstruct \( d_{\text{last}} = \det(A^*) \) using the Chinese remainder algorithm.

Theorem 14 There exists a deterministic algorithm that takes as input an \( n \times n \) matrix \( A \) over \( \mathbb{Z} \), and produces as output the Smith normal form \( S \) of \( A \). The cost of the algorithm is bounded by \( O(n^{r-1} mB(n \log n ||A||)) \) bit operations.

Proof It is well known fact is that the invariant factors \( s_1, s_2, \ldots, s_r \) are given by \( s_i = d_i/d_{i-1} \) where \( d_0 = 1 \) and for \( 1 \leq i \leq r \), \( d_i \) is the gcd of all \( i \times i \) minors of \( A \). In particular, the determinant \( d' \) of a nonzero maximal rank minor of \( A \) will be a multiple of \( d_m \), and \( d_i = (d_i/d_m)(d_{i-1}/d_{m-1}) \cdots (d_1/d_0) = s_1 s_2 \cdots s_i \). Set \( d = 2d' \) where \( d' = |d'| \) and compute \( S \) according to Theorem 12 as \( \hat{A}^{-1} \text{snf}(\phi_d(A)) \). By Theorem 11 the cost of this is \( O(n^{r-1} mB(\log d)) \) operations from \( \mathbb{Z}_n \). By Lemma 13, \( d' \) can be found in the allotted time and will be bounded in length by \( \lceil \log_2 d \rceil = O(n \log n ||A||) \) bits.

References


