Near Optimal Algorithms for Computing Smith Normal Forms of Integer Matrices

Arne Storjohann

Department of Computer Science
University of Waterloo, Ontario, Canada
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 $\mathbf{Z}_d$, we present an algorithm that computes the Smith form S of an $A \in \mathbf{Z}_d^{n\times m}$ in only $O(n^{\theta-1}m)$ operations from $\mathbf{Z}_d$. Here, θ 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 $\mathbf{Z}_d$ to get an algorithm for computing the Smith form S of an $A \in \mathbf{Z}^{n\times m}$ in $O\tilde{~}(n^{\theta-1}m \cdot \mathsf{M}(n \log ||A||))$ bit operations (where $||A|| = \max|A_{i,j}|$ and $\mathsf{M}(t)$ bounds the cost of multiplying two $\lceil t\rceil$-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 $A \in \mathsf{R}^{n\times m}$ there exist unimodular (square and invertible) matrices U and V over R such that

$$S = UAV = \begin{bmatrix} s_1 & & & & & \\ & \ddots & & & & \\ & & s_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}$$

with each s_i nonzero and with $s_i | s_{i+1}$ for $1 \le i \le r-1$. 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 [13, 1869] for matrices over the integers (in this case, each s_i is positive, $r = \mathrm{rank}(A)$ and $\det(U)$, $\det(V) = \pm 1$).

In this paper we consider the problem of computing Smith normal forms of matrices with entries from $\mathbf{Z}$ and $\mathbf{Z}_d$, the ring of integers modulo d. Computing Smith normal forms over these domains is useful in many applications, including Diophantine analysis (see Newman [12, 1972]), computing the structure of finitely generated abelian groups (see Haves, Holt & Rees [7, 1993]) and computing the structure of the class group of a number field (see Hafner & McCurley [5, 1989] and Buchmann [1, 1988]).

In Section 3 we present our main result — an asymptotically fast algorithm for computing Smith normal forms over $\mathbf{Z}_d$. Let A be an $n \times m$ matrix over $\mathbf{Z}_d$. We assume without loss of generality that $n \le m$ — 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 $\mathbf{Z}_d$ to compute the Smith normal form S of A. Here, θ 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 [3, 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 $O\tilde{~}(n^{\theta-1}m)$ [1] operations from $\mathbf{Z}_d$. The asymptotically fast algorithm for computing transforming matrices over $\mathbf{Z}_d$ 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. The first polynomial time algorithm for computing Smith normal forms over $\mathbf{Z}$ was given by Kannan & Bachem [11, 1979] and later improved by Chou & Collins [2, 1982]. More recently, Iliopoulos [9, 1989] has given an algorithm that performs all arithmetic modulo the determinant of a square nonsingular input matrix. The modular approach, which effectively controls intermediate expression swell, was extended to singular input matrices by Iliopoulos [10, 1989] and Hafner & McCurley [6, 1991]. Let A be an $n \times m$ input matrix over $\mathbf{Z}$. We show how to apply the result of Section 3 to get an algorithm that

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee. ISSAC'96, Zurich, Switzerland; ©1996 ACM 0-89791-796-0/96/07...$3.50

[1] To summarize results we use "soft-Oh" notation: for any $f, g : \Re^s \mapsto \Re$, $f = O\tilde{~}(g)$ if and only if $f = O(g \cdot \log^c g)$ for some constant $c > 0$.

requires $O\tilde{}(n^{\theta-1}m\mathsf{M}(n\log||A||))$ bit operations to produce the Smith normal form S of A. The previously best deterministic algorithm of Iliopoulos [10, 1989] (see also Hafner & McCurley [6, 1991]) requires $O\tilde{}(n^3 m\log||A||\mathsf{M}(n\log||A||))$ 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 [4, 1995] computes S in an expected number of $O\tilde{}(n^2m\mathsf{M}(n\log||A||))$ bit operations.

The algorithm that we have presented for computing Smith normal forms over $\mathbf{Z}$ 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 [7, 1993]), but are difficult to analyze. 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 $O\tilde{}(n^2\log||A||)$ 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 $S = \mathrm{diag}(s_1, s_2, \ldots, s_r, 0, \ldots, 0) \in \mathsf{R}^{n\times m}$ 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 $\mathsf{R} = \mathbf{Z}_d$, we choose our prescribed set of nonassociates to be $N_d^* = \{x \bmod d : x \in \mathbf{Z}, 0 < x \le d, x|d\}$, and for $a, b \in \mathbf{Z}_d$, write $\gcd_d(a,b)$ to denote the unique principal generator of the ideal $(a,b) \subseteq \mathbf{Z}_d$ which belongs to N_d^*. Note that $\gcd_d(a,b)$ can be computed as $\gcd(\bar{a}, \bar{b}, d) \bmod d$ where $\bar{a}$ and $\bar{b}$ are in $\mathbf{Z}$ with $\bar{a} = a \bmod d$ and $\bar{b} = b \bmod d$. For the case $a, b = 0$, we have $\gcd_d(0,0) = 0$. Over the ring $\mathsf{R} = \mathbf{Z}$, our prescribed complete set of nonassociates is simply $N^* = \{x : x \in \mathbf{Z}, x \ge 0\}$.

We present some of our complexity results in terms of the number of operations from $\mathbf{Z}_d$. Given $a, b \in \mathbf{Z}_d$, we consider a single operation from $\mathbf{Z}_d$ to be one of: (1) finding $a+b, a-b, ab \in \mathbf{Z}_d$; (2) if a divides b, finding a $q \in \mathbf{Z}_d$ with $aq = b$; (3) finding elements $g, s, t, u, v \in \mathbf{Z}_d$ 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 $sv - tu$ a unit in $\mathbf{Z}_d$. Let $\mathsf{B}(\log d)$ be a function which bounds the number of bit operations required to perform a single operations from $\mathbf{Z}_d$. Using standard integer arithmetic, $\mathsf{B}(\log d) \ll \log^2 d$, while fast integer arithmetic allows

$$\mathsf{B}(\log d) \ll \mathsf{M}(\log d)\log\log d.$$

In Section 4 we use the fact that $\mathsf{B}(\log d)$ 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 [6, 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 [6, 1991]) *There exists a deterministic algorithm that takes as input an $n \times m$ matrix A over $\mathbf{Z}_d$, and produces as output two matrices V and T satisfying $AV = T$, 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 $n, m \le b$, then the cost of the algorithm is bounded by $O(b^\theta)$ operations from $\mathbf{Z}_d$.

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

In this section we develop an asymptotically fast algorithm to compute the Smith normal form of an $A \in \mathbf{Z}_d^{n\times m}$. 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 $O(n^\theta)$ operations from $\mathbf{Z}_d$ to transforms an upper triangular $B \in \mathbf{Z}_d^{n\times n}$ to an equivalent bidiagonal matrix C. In Subsection 3.2 we show how to compute the Smith normal form of a bidiagonal $C \in \mathbf{Z}_d^{n\times n}$ in $O(n^2)$ operations from $\mathbf{Z}_d$. In Subsection 3.3 we combine these results and give an algorithm that requires $O(n^{\theta-1}m)$ operations from $\mathbf{Z}_d$ to computing the Smith normal form of an A in $\mathbf{Z}_d^{n\times m}$.

3.1 Reduction of Banded Matrices

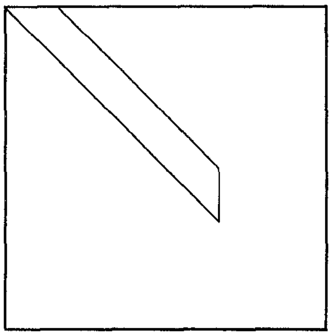
A square matrix A is upper b-banded if $A_{ij} = 0$ for $j < i$ and $j \ge i + b$, that is, if A can be written as

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

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 $\mathbf{Z}_d$, and produces as output an equivalent $n \times n$ upper $(\lfloor b/2\rfloor + 1)$-banded matrix A'. If A has last t columns zero, then A' will have last t columns zero. The cost of the algorithm is $O(n^2b^\theta)$ operations from $\mathbf{Z}_d$.*

Proof By augmenting A with at most $2b$ 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 write $\mathrm{sub}[i,k] = \mathrm{sub}_A[i,k]$ 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



Our approach is to transforms A to A' by applying (in place) a sequence of equivalence transformations to $\mathrm{sub}[is_1, n_1]$ and $\mathrm{sub}[(i+1)s_1 + js_2, n_2]$, where i and j are nonnegative integer parameters and

$$\begin{aligned} s_1 &= \lfloor b/2 \rfloor, \\ n_1 &= \lfloor b/2 \rfloor + b - 1, \\ s_2 &= b - 1, \\ n_2 &= 2(b-1). \end{aligned}$$

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 `Triang`, defined below by Lemma 3.

Lemma 3 *For $b > 2$, there exists a deterministic algorithm* `Triang` *that takes as input an $n_1 \times n_1$ upper b-banded matrix*

$$B = \left[\begin{array}{ccc|cccccc} * & \cdots & * & * & \cdots & * & * & & \\ & \ddots & \vdots & \vdots & & \vdots & \vdots & \ddots & \\ & & * & * & \cdots & * & * & \cdots & * \\ \hline & & & * & \cdots & * & * & \cdots & * \\ & & & & \ddots & & & & \vdots \\ & & & & & * & & & * \\ & & & & & & * & & * \\ & & & & & & & \ddots & \vdots \\ & & & & & & & & * \end{array}\right],$$

over $\mathbf{Z}_d$, where the principal block is $s_1 \times s_1$, and produces as output an equivalent matrix

$$B' = \left[\begin{array}{ccc|cccccc} * & \cdots & * & * & & & & & \\ & \ddots & \vdots & \vdots & \ddots & & & & \\ & & * & * & \cdots & * & & & \\ \hline & & & * & \cdots & * & * & \cdots & * \\ & & & \vdots & & & & & \vdots \\ & & & * & & & & & * \\ & & & * & & & & & * \\ & & & \vdots & & & & & \vdots \\ & & & * & \cdots & * & * & \cdots & * \end{array}\right]$$

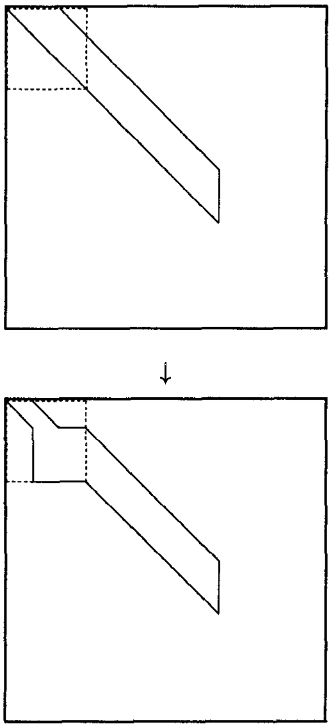
If B has last t columns zero, then B' will have last t columns zero. The cost of the algorithm is $O(b^\theta)$ operations from $\mathbf{Z}_d$.

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

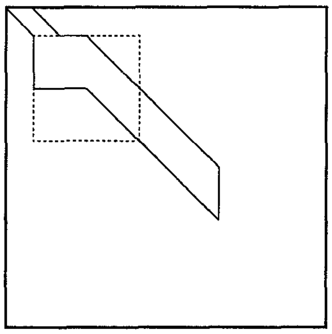
$$B' = B \left[\begin{array}{c|c} I_{s_1} & \\ \hline & V \end{array}\right].$$

Since $n_1 < 2b$, the cost is as stated. ∎

Apply subroutine `Triang` to $\mathrm{sub}[0, n_1]$ of our initial work matrix to effect the following transformation:



At this stage we can write the work matrix as



where the focus of attention is now $\mathrm{sub}[s_1, n_2]$. 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 `Shift`, defined below by Lemma 4.

Lemma 4 *For $b > 2$, there exits a deterministic algorithm* `Shift` *that takes as input an $n_2 \times n_2$ matrix*

$$C = \left[\begin{array}{ccc|ccc} * & \cdots & * & * & & \\ \vdots & & \vdots & \vdots & \ddots & \\ * & \cdots & * & * & \cdots & * \\ \hline & & & * & \cdots & * \\ & & & & \ddots & \vdots \\ & & & & & * \end{array}\right]$$

over $\mathbf{Z}_d$, where each block is $s_2 \times s_2$, and produces as output an equivalent matrix

$$C' = \left[\begin{array}{ccc|ccc} * & \cdots & * & * & & \\ & \ddots & \vdots & \vdots & \ddots & \\ & & * & * & \cdots & * \\ \hline & & & * & \cdots & * \\ & & & \vdots & & \vdots \\ & & & * & \cdots & * \end{array}\right]$$

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 $O(b^\theta)$ operations from $\mathbf{Z}_d$.

Proof Write the input matrix as

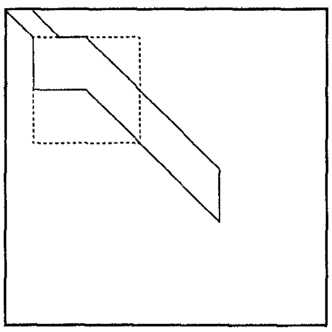
$$C = \left[\begin{array}{c|c} C_1 & C_2 \\ \hline & C_3 \end{array}\right]$$

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

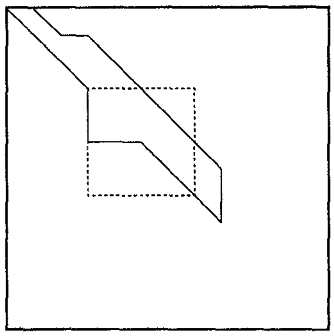
$$C' = \left[\begin{array}{c|c} U & \\ \hline & I_{s_2} \end{array}\right] \left[\begin{array}{c|c} C_1 & C_2 \\ \hline & C_3 \end{array}\right] \left[\begin{array}{c|c} I_{s_2} & \\ \hline & V \end{array}\right].$$

Since $n_2 < 2b$, the cost is as stated. ∎

Apply subroutine `Shift` to sub$[s_1 + js_2, n_2]$ for $j = 0, 1, 2, \ldots, \lfloor (n - s_1)/n_2 \rfloor$ to get the following sequence of transformations.



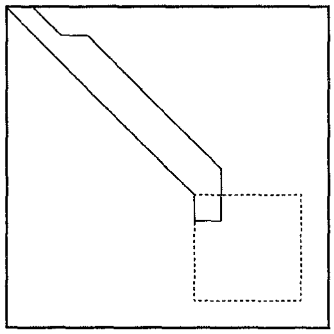
$\downarrow$



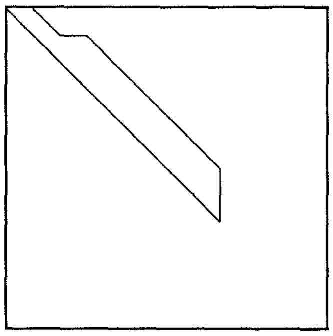
$\downarrow$

$\vdots$

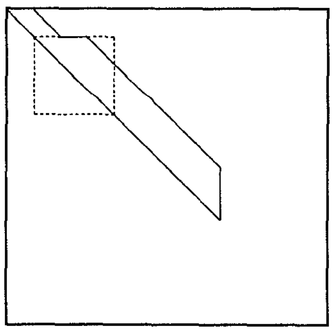
$\downarrow$



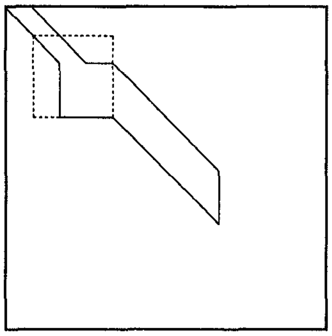
$\downarrow$



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 sub$[s_1, n_1]$ to get the following transformation.



$\downarrow$



We get the following.

Algorithm: `BandReduction`
Input: An upper b-banded matrix $A \in \mathbf{Z}_d^{n\times n}$ 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 $(\lfloor b/2 \rfloor + 1)$-banded matrix that is equivalent to A and has last t columns zero.

(1) [Initialize:]
 $s_1 \leftarrow \lfloor b/2 \rfloor$;
 $n_1 \leftarrow \lfloor b/2 \rfloor + b - 1$;
 $s_2 \leftarrow b - 1$;
 $n_2 \leftarrow 2(b-1)$;
(2) [Apply equivalence transformations:]
 for $i = 0$ to $\lceil (n-t)/s_1 \rceil - 1$
  apply `Triang` to $\mathrm{sub}_A[is_1, n_1]$;
  for $j = 0$ to $\lceil (n - t - (i+1)s_1)/s_2 \rceil - 1$
   apply `Shift` to $\mathrm{sub}_A[(i+1)s_1 + js_2, n_2]$;

Let $T(n, b)$ be the the cost of applying algorithm `BandReduction` to an $n \times n$ upper b-banded input matrix. To complete the proof of Theorem 2 we derive a bound on $T(n, b)$ in terms of number of operations from $\mathbf{Z}_d$. The number of iterations of the outer loop in step (2) is

$$L_i = \lceil (n-t)/s_1 \rceil < \frac{2n}{b-1} \tag{2}$$

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

$$L_j = \lceil (n - t - (i+1)s_1)/s_2 \rceil < \frac{n}{b-1}. \tag{3}$$

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 Lemma 3 and 4, we have

$$T(n, b) < L_i(1 + L_j)cb^\theta \tag{4}$$

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

$$\begin{aligned} T(n,b) &< L_i(1+L_j)cb^\theta \\ &\leq \left(\frac{2n}{b-1}\right)\left(1 + \frac{n}{b-1}\right)cb^\theta \\ &\leq \left(\frac{2n}{b-1}\right)\left(\frac{2n}{b-1}\right)4c(b-1)^\theta \\ &\ll n^2 b^{\theta-2} \end{aligned}$$

which completes the proof. ∎

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

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

$$D(n, b) \leq D(n, \lceil b/2 \rceil + 1) + cn^2(b-1)^{\theta-2} \tag{5}$$

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

$$\begin{aligned} D(n,n) &\leq D(n, \lfloor n/2 \rfloor + 1) + cn^2(n-1)^{\theta-2} \\ &= D(n, 2^{k-1} + 1) + cn^2(2^k)^{\theta-2} \\ &= D(n, 2^{k-2} + 1) + cn^2((2^k)^{\theta-2} + (2^{k-1})^{\theta-2}) \\ &\vdots \\ &= cn^2 \sum_{i=1}^{\log_2(2^k)} (2^i)^{\theta-2} \\ &= cn^2 \sum_{i=1}^{\log_2(n-1)} \left(\frac{n-1}{2^i}\right)^{\theta-2} \\ &= cn^2(n-1)^{\theta-2} \sum_{i=1}^{\log_2(n-1)} \left(\frac{1}{2^{\theta-2}}\right)^i \\ &\ll n^\theta \end{aligned}$$

which completes the proof. ∎

3.2 The Smith Normal Form of a Bidiagonal 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

$$A = \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & \\ & & & \ddots & * \\ & & & & * \end{bmatrix}. \tag{6}$$

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 $A \in \mathbf{Z}_d^{n\times n}$, and produces as output the Smith normal form of A. The cost of the algorithm is $O(n^2)$ operations from $\mathbf{Z}_d$.*

We require some intermediate results before proving Theorem 6.

Lemma 7 *Let a, b be elements of $\mathbf{Z}_d$. There exist elements x and u of $\mathbf{Z}_d$, with u a unit, such that $xa + b = u \gcd_d(a, b)$.*

Proof Follows from the fact that $\mathbf{Z}_d$ is a stable ring. ∎

Lemma 8 *Let*

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

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

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

Proof We show that A can be transformed to $\hat{A}$ via a sequence of unimodular row and column transformations. To begin, let x_1 and u_1 be elements of $\mathbf{Z}_d$, with u_1 a unit, such that $x_1 a + b = u_1 \gcd_d(a,b)$. (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

$$A_1 = \begin{bmatrix} g_1 & & a & \\ d & * & & \\ c & & & e \end{bmatrix}$$

where $g_1 = u_1 \gcd_d(a,b)$. 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

$$A_2 = \begin{bmatrix} g_1 & & & \\ d & * & q_1 d & \\ c & & q_1 c & e \end{bmatrix}.$$

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

$$A_3 = \begin{bmatrix} g_1 & & & \\ & * & q_1 d & \\ c & & q_1 c & e \end{bmatrix}.$$

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

$$A_4 = \begin{bmatrix} g_2 & & q_1 c & e \\ & * & q_1 d & \\ g_1 & & & \end{bmatrix}$$

where $g_2 = u_2 \gcd_d(g_1, c)$. To zero out the entry in row 3 column 1 of A_4, multiply row 3 of A_4 by $-u_2 u_1^{-1}$ (a unit) and then add q_2 times row 1 of A_1 to row 3 to obtain the equivalent matrix

$$A_5 = \begin{bmatrix} g_2 & & q_1 c & e \\ & * & q_1 d & \\ & & q_1 q_2 c & q_2 e \end{bmatrix}.$$

To complete the transformation to $\hat{A}$, transform the entry in row 1 column 1 to $\gcd_d(a,b,c)$ 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* 3×4 *matrix*

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

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

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

with $\hat{e}$ *a multiple of* e, *and* $\hat{a}$ *a divisor of both* $\hat{c}$ *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* $\mathbf{Z}_d$.

Proof Find solutions q_1 and q_2 to $a = q_1 \gcd_d(a,b)$ and $\gcd_d(a,b) = q_2 \gcd_d(a,b,c)$, 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 $2 < k \leq n$ denote by $\mathcal{T}_k^n$ the set of all $n \times n$ matrices over $\mathbf{Z}_d$ 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

$$\left[\begin{array}{c|cccc|ccc|cccc} a & & & & & & b & & & & & \\ \hline & * & * & & & & & & & & & \\ & & * & & & & & & & & & \\ & & & \ddots & * & & & & & & & \\ & & & & * & * & & & & & & \\ \hline & & & & & * & d & & & & & \\ & & & & & & c & e & & & & \\ \hline & & & & & & & * & * & & & \\ & & & & & & & & * & & & \\ & & & & & & & & & \ddots & * & \\ & & & & & & & & & & * & \end{array}\right], \qquad (7)$$

where b is in column k and divides d.

Lemma 10 *There exists a deterministic algorithm that takes as input a matrix* T *over* $\mathbf{Z}_d$ *and in* $\mathcal{T}_k^n$ *with* $2 < k < n$, *and produces as output an equivalent matrix* $\hat{T}$ *in* $\mathcal{T}_{k+1}^n$. *Furthermore, if* T_{11} *divides all entries in the first* $k-1$ *columns of* T, *then* $\hat{T}_{11}$ *divides all entries in the first* k *columns of* $\hat{T}$. *The cost of algorithm is* $O(1)$ *operations from* $\mathbf{Z}_d$.

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

$$\hat{T} = \left[\begin{array}{c|cccc|ccc|cccc} \hat{a} & & & & & & e & & & & & \\ \hline & * & * & & & & & & & & & \\ & & * & & & & & & & & & \\ & & & \ddots & * & & & & & & & \\ & & & & * & * & & & & & & \\ \hline & & & & & * & \hat{d} & & & & & \\ & & & & & & \hat{c} & \hat{e} & & & & \\ \hline & & & & & & & * & * & & & \\ & & & & & & & & * & & & \\ & & & & & & & & & \ddots & * & \\ & & & & & & & & & & * & \end{array}\right]$$

in $\mathcal{T}_{k+1}^n$. To prove the second part of the theorem, note that by Corollary 9 we have $\hat{a} = \gcd_d(a,b,c,d)$, and in particular,

$\hat{a}|a$. 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 $\mathbf{Z}_d$. We claim that

$$R(n) \leq R(n-1) + cn \tag{8}$$

for some absolute constant c. To prove (8), let A be an $n \times n$ upper bidiagonal matrix over $\mathbf{Z}_d$. We show how to produce a matrix

$$B = \begin{bmatrix} g & & & & \\ & * & * & & \\ & & * & * & \\ & & & * & \\ & & & \ddots & * \\ & & & & * \end{bmatrix} \tag{9}$$

which is equivalent to A and where g is the $\gcd_d$ 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

$$\bar{A}_3 = \begin{bmatrix} * & & * & & & \\ & 0 & & & & \\ & & * & * & & \\ & & & * & * & \\ & & & & * & \\ & & & & \ddots & * \\ & & & & & * \\ & & & & & & 0 \end{bmatrix}.$$

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 $\bar{A}_3$ will have the same invariant factors as the Smith normal form of A. Furthermore, $\bar{A}_3$ is in $\mathcal{T}_3^{n+2}$ and the entry in row 1 column 1 of $\bar{A}_3$ divides all entries in the first two columns of $\bar{A}_3$. Starting with $\bar{A}_3$, apply the algorithm of Lemma 10 for $k = 3, 4, \ldots, n+1$ to compute a succession of equivalent matrices $\bar{A}_4, \bar{A}_5, \ldots, \bar{A}_{n+2}$, with $\bar{A}_k \in \mathcal{T}_k^{n+2}$. By Lemma 10, the cost of this is $O(n)$ operations from $\mathbf{Z}_d$ and, since the last column of $\bar{A}_3$ is all zero, $\bar{A}_{n+2}$ will have the form

$$\bar{A}_{n+2} = \begin{bmatrix} g & & & & & 0 \\ & 0 & & & & \\ & & * & * & & \\ & & & * & * & \\ & & & & * & \\ & & & & \ddots & * \\ & & & & & * \\ & & & & & 0 \end{bmatrix}$$

where g divides all entries in the first $k+1$ columns of $\bar{A}_{n+2}$. Finally, delete rows and columns 2 and $n+2$ of $\bar{A}_{n+2}$ (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

$$\begin{aligned} R(n) &\leq R(n-1) + cn \\ &= R(0) + c\sum_{i=1}^{n} i \\ &\ll n^2 \end{aligned}$$

∎

3.3 The Smith Normal Form Algorithm

Theorem 11 *There exists a deterministic algorithm that takes as input an $n \times m$ matrix A over $\mathbf{Z}_d$, and produces as output the Smith normal form of A. The cost of the algorithm is $O(n^{\theta-1}m)$ operations from $\mathbf{Z}_d$.*

Proof By augmenting A with at most $n-1$ columns, we can assume that $m = kn$ for some integer k. The algorithm consists of three steps. First, find an $n \times n$ upper triangular matrix B that has the same invariant factors as A. This can be accomplished in $O(n^{\theta-1}m)$ operations from $\mathbf{Z}_d$ 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 $in+1, in+2, \ldots, (i+2)n$. 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 bidiagonal 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 each of these steps is bounded by $O(n^{\theta-1}m)$ operations from $\mathbf{Z}_d$. ∎

4 Smith Normal Form over Z

In this section we show how to use the algorithm for Smith normal form over $\mathbf{Z}_d$ presented in section 3 to get an asymptotically fast algorithm for computing Smith normal forms over $\mathbf{Z}$. We follow the approach of many previous algorithms and compute over $\mathbf{Z}_d$, where d is chosen to be a positive multiple of the product invariant factors of A (see Hafner & McCurley [6, 1991]). To make this idea precise, we define homomorphisms $\phi = \phi_d$ and $\bar{\phi}^{-1} = \bar{\phi}_d^{-1}$ which we use to move between the two domains $\mathbf{Z}$ and $\mathbf{Z}_d$. Define $\phi : \mathbf{Z} \to \mathbf{Z}_d$ by $\phi : a \mapsto \bar{a}$ where $\bar{a} = a \bmod d$. Define the pullback homomorphism $\bar{\phi}^{-1} : \mathbf{Z}_d \to \mathbf{Z}$ by $\bar{\phi}^{-1} : \bar{a} \mapsto a$ where $\bar{a} = a \bmod d$ and $0 \leq \bar{a} < d$. For the following theorem, we denote by $\mathsf{snf}(X)$ the Smith normal form of an input matrix X over the domain of entries of X (either X is over $\mathbf{Z}$ or X is over $\mathbf{Z}_d$). We also write $\phi(A)$ to denote the matrix obtained by applying ϕ to each entry of A.

Theorem 12 *Let A be a matrix over $\mathbf{Z}$. If $d = 2d'$ where d' is a positive multiple of the product of the invariant factors of A, then*

$$\mathsf{snf}(A) = \bar{\phi}^{-1}(\mathsf{snf}(\phi(A))).$$

Proof Let $\mathsf{snf}(A) = \mathrm{diag}(s_1, s_2, \ldots, s_r, 0, \ldots, 0)$. Each s_i satisfies $1 \leq s_i \leq d' < d$, so we have $s_i = \bar{\phi}^{-1}(\phi(s_i))$ for $1 \leq i \leq r$ and

$$\mathsf{snf}(A) = \bar{\phi}^{-1}(\phi(\mathsf{snf}(A))). \tag{10}$$

Next, let U and V be unimodular matrices over $\mathbf{Z}$ such that $UAV = \mathsf{snf}(A)$. Then

$$\phi(U)\,\phi(A)\,\phi(V) = \phi(\mathsf{snf}(A))$$

where $\phi(U)$ and $\phi(V)$ are unimodular over $\mathbf{Z}_d$. It is easily verified that $\phi(\mathsf{snf}(A))$ is in Smith normal form over $\mathbf{Z}_d$. In particular, $\phi(s_i)$ divides $\phi(s_{i+1})$ for $1 \leq i \leq r-1$ and $\phi(s_i) \in N_d^*$ for $1 \leq i \leq r$. Since the Smith normal form of $\phi(A)$ is unique, we must have

$$\phi(\mathsf{snf}(A)) = \mathsf{snf}(\phi(A)). \tag{11}$$

The desired result follows by substituting (10) into (11). ■

Lemma 13 *There exists a deterministic algorithm that takes as input an $n \times m$ matrix A over $\mathbf{Z}$, and produces as output the determinant d^* of a nonsingular maximal rank minor of A. The cost of the algorithm is $O(n^{\theta-1} m \mathsf{B}(n \log n ||A||))$ bit operations.*

Proof We apply the standard homomorphic imaging scheme. Compute a number z such that $\Pi_{p \leq z}^{p \text{ prime}} p > n^{n/2} ||A||^n$. 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 Ibarra, Moran & Hui [8, 1982] to compute the rank of A over $\mathbf{Z}_p$ for each prime $p \leq z$, since their algorithm returns also a maximal set of linearly independent rows and columns of A over $\mathbf{Z}_p$. The cost of their algorithm for a single prime p is $O(n^{\theta-1} m)$ operations form $\mathbf{Z}_p$. Compute $\det(A^*) \bmod p$ for each prime $p \leq z$, again using the algorithm [8, 1982], and reconstruct $d^* = \det(A^*)$ using the Chinese remainder algorithm. ■

Theorem 14 *There exists a deterministic algorithm that takes as input an $n \times m$ matrix A over $\mathbf{Z}$, and produces as output the Smith normal form S of A. The cost of the algorithm is bounded by $O(n^{\theta-1} m \mathsf{B}(n \log n ||A||))$ 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_r, and $d_r = (d_1/d_0)(d_2/d_1) \cdots (d_r/d_{r-1}) = s_1 s_2 \cdots s_r$. Set $d = 2d'$ where $d' = |d^*|$ and compute S according to Theorem 12 as $\bar{\phi}_d^{-1}(\mathsf{snf}(\phi_d(A)))$. By Theorem 11 the cost of this is $O(n^{\theta-1} m \mathsf{B}(\log d))$ operations from $\mathbf{Z}_d$. 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

[1] Buchmann, J. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. In *Séminaire de théorie des nombres* (Paris, 1988).

[2] Chou, T.-W. J., and Collins, G. E. Algorithms for the solutions of systems of linear diophantine equations. *SIAM Journal of Computing 11* (1982), 687–708.

[3] Coppersmith, D., and Winograd, S. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation 9* (1990), 251–280.

[4] Giesbrecht, M. Fast computation of the Smith normal form of an integer matrix. In *Proceedings of ISSAC'95* (Montreal, Canada, 1995), pp. 110–118.

[5] Hafner, J. L., and McCurley, K. S. A rigorous subexponential algorithm for computation of class groups. *J. Amer. Math. Soc. 2* (1989), 837–850.

[6] Hafner, J. L., and McCurley, K. S. Asymptotically fast triangularization of matrices over rings. *SIAM Journal of Computing 20*, 6 (Dec. 1991), 1068–1083.

[7] Havas, G., Holt, D. F., and Rees, S. Recognizing badly presented Z-modules. *Linear Algebra and its Applications 192* (1993), 137–163.

[8] Ibarra, O., Moran, S., and Hui, R. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms 3* (1982), 45–56.

[9] Iliopoulos, C. S. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. *SIAM Journal of Computing 18*, 4 (Aug. 1989), 658–669.

[10] Iliopoulos, C. S. Worst-case complexity bounds on algorithms for computing the canonical structure of infinite abelian groups and solving systems of linear diophantine equations. *SIAM Journal of Computing 18*, 4 (Aug. 1989), 670–678.

[11] Kannan, R., and Bachem, A. Polynomial algorithms for computing the Smith and Hermite normal forms of and integer matrix. *SIAM Journal of Computing 8*, 4 (Nov. 1979), 499–507.

[12] Newman, M. *Integral Matrices.* Academic Press, 1972.

[13] Smith, H. J. S. On systems of linear indeterminate equations and congruences. *Phil. Trans. Roy. Soc. London 151* (1861), 293–326.

Arne Storjohann is an Assistant in the Institute for Scientific Computation at ETH-Zurich. He holds B.Math. and M.Math. degrees from the University of Waterloo, Canada. His research interests are in Symbolic Computation, Linear Algebra, Analysis of Algorithms, and Matrix Normal Forms.