JOURNAL OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 2, Number 4, October 1989

A RIGOROUS SUBEXPONENTIAL ALGORITHM FOR COMPUTATION OF CLASS GROUPS

JAMES L. HAFNER AND KEVIN S. McCURLEY

I. INTRODUCTION

In 1801, Gauss published the monumental work *Disquisitiones Arithmeticæ*, in which he developed among other things a theory involving binary quadratic forms. A large part of Gauss' investigations was concerned with computational questions, and in particular, Gauss devoted a great deal of attention to the problem of calculating equivalence classes of binary quadratic forms of a fixed determinant. In modern notation, let $C(-d)$ be the group, under composition, of $SL(2, \mathbf{Z})$-equivalence classes of positive definite binary quadratic forms of discriminant $-d$, and let $h(-d)$ be its order (see [12] for more precise definitions; note that this differs slightly from Gauss' original definition). Properties of $C(-d)$ and $h(-d)$ are important in many areas, including the theory of factorization in quadratic fields and the distribution of prime numbers in arithmetic progressions. Over the last 188 years, the problem of determining all values of d for which $h(-d)$ has a given value has come to be known as "Gauss' class number problem," and has been studied in depth. The origin of this problem can be traced to [3, Articles 304–305], where Gauss made a conjecture that implies $\lim_{d\to\infty} h(-d) = \infty$. Goldfeld, Gross, and Zagier (see [4]) only recently managed to give a constructive proof for the bound $h(-d) > c \log d$ for some constant c. It is only as a consequence of this theorem that we know of an algorithm for calculating all d for which $h(-d) = k$ (although this algorithm has a running time that is exponential in k).

In this paper, we shall be concerned with the related computational problem of calculating $h(-d)$ and the invariants of the finite abelian group $C(-d)$ for a given d. By the invariants, we mean positive integers $m_1, \ldots, m_k$ such that $m_1 | m_2 | \cdots | m_k$, $h(-d) = \prod_{i=1}^{k} m_i$ and

$$C(-d) \cong \bigoplus_{i=1}^{k} \mathbf{Z}/m_i\mathbf{Z}.$$

These are sometimes refered to as the torsion coefficients. Gauss [3, Article 185] gave at least two algorithms for computing only $h(-d)$ that used the construction of a set of representatives for the equivalence classes. A related

Received by the editors April 14, 1989.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 11Y16, 11E41; Secondary 11R11, 11R29.

0894-0347/89 $1.00 + $.25 per page

algorithm of Gauss appears in [3, Article 305], in which he describes a technique based on what we would now recognize as Lagrange's theorem for a finite group. Since it is conjectured that $h(-d) \gg d^{1/2}/\log\log d$, these algorithms of Gauss will have complexity at least $d^{1/2}$, and for the first two, the complexity is at least d.

Approximately 170 years after the time of Gauss, Shanks [19, 20] discovered an algorithm to compute $h(-d)$ that has running time of $O(d^{1/4+\varepsilon})$, or $O(d^{1/5+\varepsilon})$ under the assumption of the extended Riemann hypothesis (see Lenstra [10] or Schoof [16] for the complexity analysis). Shanks' algorithm depends on the Dirichlet class number formula and the fact that the equivalence classes form a group.

The purpose of the present paper is to prove the following theorem.

Theorem 1. *Assuming the extended Riemann hypothesis, there exists a Las Vegas algorithm that will compute the invariants of $C(-d)$, and so compute $h(-d)$ in an expected running time of $L(d)^{\sqrt{2}+o(1)}$ bit operations, where*

$$L(d) = \exp(\sqrt{\log d \log\log d}).$$

The appearance of the function $L(d)$ will not be surprising to those who are familiar with the current state of integer factoring algorithms, since the best algorithms known for factoring an integer n require $L(n)^c$ bit operations. In fact, the algorithm that we shall describe is closely related to the factoring algorithm of Seysen [18]. The connection between integer factoring and calculation of class numbers is well known, since in the same paper where he presented his class number algorithm, Shanks [19] described its relation to factoring.

The technique used in our algorithm is to construct a basis for the $\mathbf{Z}$-module Λ of relations on a set of n generators for the Gauss class group $C(-d)$. Then $C(-d) \cong \mathbf{Z}^n/\Lambda$. From this basis for Λ, it is easy to calculate the desired invariants of $C(-d)$ by computing the Smith normal form of the basis matrix.

In a previous paper by the second author [11], an algorithm for computing $h(-d)$ was presented along with a heuristic argument that a running time of $L(d)^{3/\sqrt{8}+o(1)}$ should be possible. The idea of constructing a generating set for the lattice of relations was also suggested previously in a paper by A. K. Lenstra and H. W. Lenstra, Jr. [9]. The algorithm presented in this paper is closely related to that of [11], but is structured in such a way as to allow a proof for the running time and extracting the extra information about the invariants. A practical version of the algorithm might take yet another form.

At present, we see no way to remove the assumption of the extended Riemann hypothesis (ERH) entirely, but it is probably possible to prove unconditionally that the algorithm described below will perform as claimed in the theorem for almost all inputs d. It would be interesting to see if the techniques used here can also be extended to the case of positive discriminants, or to class groups of arbitrary number fields. We have not investigated this.

In passing, we mention that the problem of computing the structure of the class group $C(-d)$ belongs to the complexity class NP under the assumption of ERH. We shall omit the details for this, but the essential ingredients for

the proof are implicit in [11] and the methods of this paper. The argument in [11] involved guessing a basis matrix for a certain lattice. From this basis matrix, we need only verify that its columns belong to the lattice and that its determinant has the correct value. This gives a proof for the correctness of the class number, and all that is required to obtain the structure of $C(-d)$ is to compute the Smith normal form of the matrix.

Our presentation here is not fully self-contained, for which we apologize. Occasional references will be made to Seysen [18] and [11], and definitions and notation for quadratic forms and class groups may be found in several sources, including [12, 6, 10, 16, and 11].

2. Description of the algorithm

Throughout this paper, we shall assume that d satisfies $d \equiv 0$ or $3 \pmod 4$ and $d > 4$, since these represent the only interesting values for our problem. We shall use the notation that $\langle g_1, \ldots, g_k \rangle$, is the group generated by $g_1, \ldots, g_k$, and $W_n(t)$ is the box $\{x : x \in \mathbf{Z}^n, \|x\|_\infty \leq t\}$. If $\mathscr{L}$ is a full-dimensional lattice in $\mathbf{Z}^n$, then $\operatorname{Det}\mathscr{L}$ refers to the absolute value of the determinant of a basis matrix for $\mathscr{L}$, or, equivalently, the index of $\mathscr{L}$ in $\mathbf{Z}^n$ viewed as abelian groups. Also, if A is a (not necessarily square) matrix, then $\Lambda(A)$ refers to the lattice generated by the columns of A.

Our model for the class group $C(-d)$ is the set of $SL(2, \mathbf{Z})$-equivalence classes of quadratic forms $ax^2 + bxy + cy^2$ of a given negative discriminant $b^2 - 4ac = -d$. We shall use the notation (a, b, c) to denote such a form, and $[(a, b, c)]$ to denote its equivalence class. Under the assumption of the extended Riemann hypothesis, a set of generators for $C(-d)$ is provided by the following result from Schoof [16].

Theorem 2. *Let p_i be the ith prime with $(-d/p_i) = 1$, and let*

$$b_i = \min\{b \in \mathbf{Z}^+ : b^2 \equiv -d \pmod{4p_i}\}. \tag{2.1}$$

If ERH is true, then there exists an effectively computable constant c_0 such that the classes $[(p_i, b_i, \cdot)]$, $1 \leq i \leq n_0$, generate $C(-d)$, where n_0 is the largest integer such that $p_{n_0} \leq c_0 \log^2 d$.

In what follows, we shall use f_i to denote the equivalence class $[(p_i, b_i, \cdot)]$, and as in [18] and [11], we define a homorphism $\varphi : \mathbf{Z}^n \to C(-d)$ by

$$\varphi(x_1, \ldots, x_n) = \prod_{i=1}^{n} f_i^{x_i}.$$

We shall use Λ to denote the lattice of integer relations on $f_1, \ldots, f_n$, i.e., $\ker \varphi$.

The following is an outline of the algorithm. We will be sketchy in the description and analysis of Steps 1 and 2 as these are given in detail in [11]. The remaining steps will be described in more detail later. In the outline below, the constants c_0 and n_0 are the same as in Theorem 2, and the parameters z and m are left unspecified until the proof of correctness and the analysis of the

running time. Later, it will be shown that with high probability the algorithm will terminate with the correct output if $m = n^{1+o(1)}$. Note that it follows from [11] that the algorithm is Las Vegas under the assumption of ERH, i.e., the output is correct.

Class Group Algorithm. Input $d > 4$ with $d \equiv 0$ or $3 \pmod 4$, and output $h(-d)$ and the invariants of the class group $C(-d)$.

Step 1. Compute a rational number B with $B/2 \leq h(-d) < B$ (see [11]).

Step 2. Set $N = \lfloor L(d)^z \rfloor$. Compute the classes $f_i = [(p_i, b_i, \cdot)]$ with $p_i \leq N$. Let n be the number with $p_i \leq N$. Note that $n = N^{1+o(1)} = L(d)^{z+o(1)}$.

Step 3. Generate n sparse relations $w_1, \ldots, w_n$ on the generators $f_1, \ldots, f_n$ from the box $W_n(n^2 d)$, using the method of Seysen [18].

Step 4. Set $h_0 := |\det(A_0)|$, where A_0 is the $n \times n$ matrix with columns $w_1, \ldots, w_n$.

Step 5. Generate new relations $v_1, \ldots, v_m$ with seeds from the box $W_n(d^2)$.

Step 6. Calculate the Hermite normal form H of the $n \times (n+m)$ matrix A_1 whose columns are $w_1, \ldots, w_n, v_1, \ldots, v_m$.

Step 7. Let h be the product of the diagonal entries of H. If $h \geq B$, then return to Step 3. Otherwise, calculate the Smith normal form S of H, and output $h(-d) = h$ and the diagonal entries of S which are greater than 1.

In the rest of this section, we will present details for Steps 3–7. Later, we will give a rigorous proof of correctness and the analysis for the running time. In §5, we will describe variations of these steps that may lead to faster algorithms.

We first describe the procedure for Step 3. This is simply a translation of Seysen's method [18] into our notation. The following procedure is used to generate the column $w_k = (w_{ik}, \ldots, w_{nk})^t$ of A_0.

Seysen Algorithm.

Step 3.1. Choose integers $x_1, \ldots, x_{n_0}$ randomly from a uniform distribution on the interval $[0, d]$. Set $x_i = 0$ for $i = n_0 + 1, \ldots, n$.

Step 3.2. Compute the reduced form $(a, b, \cdot)$ in the class $f_k^{2nd} \prod_{j=1}^n] f_j^{x_j}$.

Step 3.3. Attempt to factor $a = \prod_{j=1}^n p_j^{u_j}$ using trial division. If this fails, then return to Step 3.1.

Step 3.4. Compute $y_j = \pm u_j$ such that $[(a, b, \cdot)] = \prod_{j=1}^n f_j^{y_j}$ (see [18, Theorem 3.1]).

Step 3.5. For $j = 1, \ldots, n$, let $w_{jk} = x_j - y_j + \delta_{jk} 2nd$, where δ_{jk} is the Kronecker delta.

Note that if d is sufficiently large, then all of the entries in w_k are trivially bounded in absolute value by $n^2 d$ and that the w_k's are sparse.

Next, we describe the procedure for calculating $|\det(A_0)|$ in Step 4. We begin by calculating a set $\mathscr{P}$ of primes whose product exceeds $n^{5n/2} d^n$. Then we use

row elimination to calculate $|\det(A_0)|$ modulo p for each prime p in the set $\mathscr{P}$. Finally, we use the Chinese remainder theorem to calculate $|\det(A_0)|$ modulo the product of the primes in $\mathscr{P}$. The Hadamard inequality implies that $|\det(A_0)| < n^{5n/2}d^n$, so the Chinese remainder theorem will, in fact, give us the integer $|\det(A_0)|$.

In Step 5, we can generate relations as follows. We choose random integers (the seeds) x_j, $1 \le j \le n$, from the interval $[-d^2, d^2]$, and compute and reduce the form $\prod_{j=1}^n f_j^{x_j}$. We then try to factor the lead coefficient of the reduced form as a product of the primes $p_1, \ldots, p_n$ using trial division. If we succeed in factoring the lead coefficient, then from this we derive a relation of the form $v_i = x - y$, where $\prod_{j=1}^n f_j^{x_j} = \prod_{j=1}^n f_j^{y_j}$, and the y_j's are small, i.e., bounded in absolute value by $\log d$. This is similar to the method above, except that our relations need not lead to a diagonally dominate sparse matrix. We also note that this procedure may require inverting forms, but this is trivial since the inverse of the reduced form (a, b, c) is $(a, -b, c)$.

The method that we shall describe for computing the Hermite normal form in Step 6 is based on the familar technique of unimodular elementary column operations, except that the entries are reduced modulo h_0. We carry out the arithmetic modulo h_0 to ensure that the numbers encountered during column operations remain manageable in size. This idea of using modular arithmetic was described in the work of Hu [5], Domich, Kannan, and Trotter [2], and Schrijver [17], although they suggested using the determinant of the lattice as the modulus and worked only with square matrices. In our case, we will not have the determinant of the lattice until we produce the Hermite normal form of the matrix, so we use the arithmetic modulo a (possibly large) multiple of the determinant to finally arrive at the determinant.

The first step is to convert A_1 into a lower triangular matrix L using unimodular column operations followed by reduction modulo h_0. This proceeds in the standard order, moving from top to bottom and left to right. In order to introduce a zero in the i, j location (where $i < j$), we use the extended Euclidean algorithm to calculate integers r and t such that $ra_{ii} + ta_{ij} = g$, where $g = \gcd(a_{ii}, a_{ij})$ and $|r|, |t| \le h_0$. We then replace column i by $r\cdot$(column i) $+t\cdot$(column j), and we replace column j by $-a_{ij}/g\cdot$(column i) $+a_{ii}/g\cdot$(column j). This is equivalent to postmultiplication by the $(n+m)\times(n+m)$ unimodular matrix constructed by embedding the matrix

$$\begin{bmatrix} r & -a_{ij}/g \\ t & a_{ij}/g \end{bmatrix}$$

into the $(n+m)\times(n+m)$ identity matrix, and will replace a_{ii} by $\gcd(a_{ii}, a_{ij})$ and a_{ij} by 0. After each such column operation, we reduce the entries in the two columns modula h_0.

From this lower triangular matrix L, we compute the Hermite normal form $H = (H_{ij})$ as follows. First, we reconstruct the diagonal of H. Let L_{ij}, i, $j = 1, \ldots, n$, be the entries of L. Set $D_1 = h_0$. Then for $i = 1, \ldots, n$, use the extended Euclidean algorithm to find integers r_i and t_i such that $r_iD_i + t_iL_{ii} = H_{ii}$, where $H_{ii} = \gcd(D_i, L_{ii})$ and put $D_{i+1} = D_i/H_{ii}$. In §3.3, we

prove that the H_{ii} are the main diagonal entries of H. The rest of the entries of H are computed as follows. For $i = 1, \ldots, n$ multiply the ith column of L by t_i and reduce modula D_i. This produces a matrix whose columns generate Λ and whose diagonal is the diagonal of H. The final step in computing H is to use unimodular column operations to reduce each of the entries below the diagonal modulo the diagonal entry in its corresponding row. The order of operations is to work from top to bottom and from left to right, and after each column operation we reduce the column entries modulo h.

Finally, we describe a procedure for computing the Smith normal form S of H. Once again, we use modular arithmetic, but now we work modulo $h(-d) = \det(H)$. In the first stage of the algorithm, we perform unimodular row and column operations on H followed by reduction modulo $h(-d)$ to construct a matrix of the form

$$B = \left[\begin{array}{c|ccc} B_{11} & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & B^* & \\ 0 & & & \end{array}\right], \tag{2.2}$$

with the further condition that B_{11} divides every entry of the matrix B^*. We begin by setting $B = H$. We then perform unimodular column operations and reduction modulo $h(-d)$ in order to replace B_{11} by the gcd of the entries in the first row, and to introduce zeros into the rest of the first row. We then perform unimodular *row* operations so as to replace the new B_{11} by the gcd of the entries in the first column. This may destroy the zeros in the first row, but if so, then it reduces B_{11} by a multiplicative factor of at least 2. Hence, if we repeat the procedure at most $O(\log h(-d)) = O(\log d)$ times, then we arrive at a matrix of the form (2.2). If now there exists an entry B^*_{ij} of B^* for which $B_{11} \nmid B^*_{ij}$, then we add row i back to row 1 and repeat the procedure. At each stage, we will reduce B_{11} by a multiplicative factor of at least 2, so in at most $O(\log d)$ iterations, we will finally arrive at a matrix of the form (2.2) with B_{11} dividing every entry of B^*. We now apply essentially the same procedure to B^* (note that this will not change the first row or column of B). Continuing in this way, we eventually arrive at a diagonal matrix $\tilde{S}$ with

$$\tilde{S}_{11} | \tilde{S}_{22} | \cdots | \tilde{S}_{nn}.$$

Then we use a reconstruction procedure to produce the Smith normal form S from $\tilde{S}$. To do this, we set $D_1 = h(-d)$, and then for $i = 1, \ldots, n$, we set $S_{ii} = \gcd(D_i, \tilde{S}_{ii})$, and $D_{i+1} = D_i / S_{ii}$. The matrix $S = \operatorname{diag}(S_{11}, \ldots S_{nn})$ is now the Smith normal form of H.

This completes the description of the algorithm. In the next section, we prove the correctness of each of these steps.

3. Why it works

Let $\Lambda_i = \langle w_1, \ldots, w_n, v_1, \ldots, v_i \rangle$. Ultimately, our goal is to construct a set of generators for the lattice Λ in $\mathbf{Z}^n$ such that $\mathbf{Z}^n/\Lambda \cong C(-d)$ and $\operatorname{Det}\Lambda = h(-d)$. From this set of generators, we will construct a basis in triangular form, and from this, we derive the invariants of the group $C(-d) \cong \mathbf{Z}^n/\Lambda$.

At the completion of Step 3, we have generated a lattice Λ_0 which is contained in Λ and is of finite index. Thus, $G = \Lambda/\Lambda_0$ is a finite abelian group of order $\operatorname{Det}\Lambda_0/\operatorname{Det}\Lambda = h_0/h(-d) \leq h_0$. The new relations from Step 5 then generate a tower of lattices

$$\Lambda_0 \subseteq \Lambda_1 \subseteq \cdots \subseteq \Lambda_m \subseteq \Lambda.$$

Alternately, we can view Step 5 as generating elements of G given by coset representatives of Λ_0 in Λ.

Our claim is that if the elements of G are chosen almost uniformly at random, then with high probability $\Lambda = \Lambda_m$ for some finite m, and in this case the algorithm produces the correct output and terminates. Thus, we will need to give a bound for m, demonstrate that the selection process of Step 5 simulates random selection of elements of G, and prove that the procedure of Steps 6 and 7 will produce the correct diagonal elements of the Hermite and Smith normal forms of A_1, respectively. This will prove the correctness of the algorithm.

3.1. **Geometry.** In this section, we justify our claim that the selection process of Step 5 does, in fact, simulate the selection of random elements of our group $G = \Lambda/\Lambda_0$. To do this, we need to show that the coset representatives for Λ_0 in Λ are well distributed in the box $W_n(d^2)$. We begin by stating the following lemma.

Lemma 1. *Let F be a fundamental domain for a lattice $\mathscr{L}$ of rank n in $\mathbf{Z}^n$, let $w \in \mathbf{Z}^n$, and let $D = D(\mathscr{L})$ be an upper bound for the diameter of F (in the Euclidean norm). Let $N_{\mathscr{L}}(t, w)$ be the cardinality of the set $\{v : v \in (w + \mathscr{L}) \cap W_n(t)\}$. Then*

$$N_{\mathscr{L}}(t, w) = \frac{(2t)^n}{\operatorname{Det}\mathscr{L}}\{1 + O(nD/t)\},$$

provided $t > nD$.

Proof. Let $\widetilde{W}_n(t)$ be the box $\{x : x \in \mathbf{R}^n, \|x\|_\infty \leq t\}$. Note that $\operatorname{vol} F = \operatorname{Det}\mathscr{L}$.

Each lattice point of $w + \mathscr{L}$ corresponds to a translation of F by the vector associated to that lattice point. Hence, we can count the number of lattice points by counting the number of copies of F inside the box, with care taken near the boundaries of the box. Clearly, the union of all these translates of F by the lattice points in $(w + \mathscr{L}) \cap W_n(t)$ is a subset of $\widetilde{W}_n(t + D)$ and contains $\widetilde{W}_n(t - D)$. Thus, we have

$$\operatorname{vol}\widetilde{W}_n(t - D) \leq \operatorname{Det}\mathscr{L} \cdot N_{\mathscr{L}}(t, w) \leq \operatorname{vol}\widetilde{W}_n(t + D).$$

But

$$\operatorname{vol}\widetilde{W}_n(t \pm D) = (2(t \pm D))^n = (2t)^n = \{1 + O(nD/t)\},$$

and this proves the lemma. □

A relation generated by Step 5 is of the form $x - y(x)$, where x is randomly selected from $W_n(d^2)$, and $y = y(x)$ satisfies $0 \le |y_i| \le \log d$. Let $\mathscr{S}$ be the set of classes containing "smooth" reduced forms. In order to prove our claim that the relations selected in Step 5 simulate the selection of random elements in G, we shall require the following lemma.

Lemma 2. *Let $w \in \Lambda$ be fixed. Then with the notation above,*

$$\Pr(x - y(x) \in w + \Lambda_0 |\ \varphi(x) \in \mathscr{S}) = \frac{h(-d)}{h_0}\left\{1 + O\left(\frac{n^3}{d}\right)\right\}, \tag{3.1}$$

where $\Pr(\cdot|\cdot)$ denotes conditional probability.

Proof. The probability in (3.1) is

$$\frac{\Pr(x - y(x) \in w + \Lambda_0,\ \varphi(x) \in \mathscr{S})}{\Pr(\varphi(x) \in \mathscr{S})}$$

which can be rewritten as

$$\frac{\#\{x \in W_n(d^2) :\ \varphi(x) \in \mathscr{S},\ x - y(x) \in w + \Lambda_0\}}{\#\{x :\ x \in W_n(d^2),\ \varphi(x) \in \mathscr{S}\}}. \tag{3.2}$$

We may view the vector y as being determined either by x or else the class $f \in \mathscr{S}$ with $\varphi(x) = f$. Hence, the numerator of (3.2) is

$$\begin{aligned}\sum_{f \in \mathscr{S}} \#\{x \in W_n(d^2),\ x \in y(f) + w + \Lambda_0\} &= \sum_{f \in \mathscr{S}} \frac{(2d^2)^n}{h_0}\left(1 + O\left(\frac{nD}{d^2}\right)\right) \\ &= \frac{\#\mathscr{S} \cdot (2d^2)^n}{h_0}\left(1 + O\left(\frac{n^3}{d}\right)\right),\end{aligned}$$

since the diameter $D = D(\Lambda_0)$ of the fundamental domain of Λ_0 is $O(n^2 d)$ by the triangle inequality. Furthermore, the denominator of (3.2) is just

$$\sum_{f \in \mathscr{S}} \#\{x \in W_n(d^2),\ x \in y(f) + \Lambda\} = \frac{\#\mathscr{S} \cdot (2d^2)^n}{h(-d)}\left(1 + O\left(\frac{n^3}{d}\right)\right),$$

by the same argument, and this proves the desired result. □

3.2. $2n$ **vectors suffice.** We require a simple lemma involving finite groups.

Lemma 3. *Let G be a finite abelian group of order M with*

$$G \cong \bigoplus_{p|M} \bigoplus_{i=1}^{\alpha_p} \mathbf{Z}/p^{\beta_i(p)}\mathbf{Z}.$$

Let $g_1, \ldots, g_k$ be elements of G that are chosen with replacement according to a probability distribution with density p satisfying $p(g) \le \frac{\alpha}{M}$ for every $g \in G$. Then the probability that $\langle g_1, \ldots, g_k\rangle = G$ is at least $1 - \alpha^k 2^{I-k}$, where $I = \sum_{p|M} \alpha_p$ is the number of primary invariants of G.

Proof. We give an upper bound for $\Pr(\langle g_1, \ldots, g_k\rangle \ne G)$. Clearly,

$$\Pr\{\langle g_1, \ldots, g_k\rangle \ne G\} \le \left(\tfrac{\alpha}{M}\right)^k \#\{(g_1, \ldots, g_k) :\ \langle g_1, \ldots, g_k\rangle \ne G\}.$$

Now $\langle g_1, \ldots, g_k \rangle \neq G$ if and only if $\{g_1, \ldots, g_k\}$ is a subset of some maximal proper subgroup H of G. Thus,

$$\#\{(g_1, \ldots, g_k): \langle g_1, \ldots, g_k \rangle \neq G\} \leq \sum_{H \text{ maximal}} |H|^k.$$

Now each maximal subgroup $H < G$ is uniquely determined by its factor group G/H which has order a prime $p|M$. Thus, our sum over maximal subgroups can be split as

$$\sum_{p|M} \sum_{\substack{H \text{ maximal} \\ |G/H|=p}} |H|^k. \tag{3.3}$$

An elementary counting argument shows that the number of subgroups of order p is $(p^{\alpha_p} - 1)/(p-1)$. If $I = \sum_{p/M} \alpha_p \leq k$, then the sum (3.3) is bounded by

$$\sum_{p/M} \left(\frac{M}{p}\right)^k \frac{p^{\alpha_p} - 1}{p-1} \leq M^k \sum_{p/M} 2^{\alpha_p - k} \leq M^k 2^{I-k}.$$

This concludes the proof, since the result is trivial if $k < I$. □

Note that the lemma is meaningful only if $\alpha < 2$. This is consistent with the statement that a proper subgroup of G contains at most $|G|/2$ elements. Note also that for an arbitrary group, $I \leq \log|G|/\log 2$.

Now we have already noted that if $G = \Lambda/\Lambda_0$, then

$$|G| = h_0/h(-d) \leq h_0 \leq \exp\{n^{1+o(1)}\}.$$

Also by Lemma 2, we can take $\alpha = 1 + O(n^3/D)$. Thus, it is possible to choose $m = n^{1+o(1)}$ so that if d is sufficiently large, then m satisfies

$$m \geq \frac{\log|G| + \log d}{\log(2/\alpha)}.$$

Consequently, with this choice of m, the probability is at least $1 - 1/d$ of computing enough relations in Step 5 to generate Λ after m iterations.

3.3. **Hermite and Smith normal forms.** In this section, we prove a lemma concerning the problem of computing the Hermite normal form of an integral matrix using modular arithmetic. Our argument is adapted from the paper by Domich, Kannan, and Trotter [2], in which they described a procedure for computing the Hermite normal form of a square matrix using arithmetic modulo the determinant. We generalize their statement in a rather trivial way so as to allow nonsquare matrices and arithmetic modulo a (large) multiple of the determinant of the lattice spanned by the columns of the matrix. The result is the following.

Lemma 4. *Let A be an integral matrix of size $n \times s$ and rank n, $H = (H_{ij})$ its Hermite normal form (in lower triangular form), and D a positive integer multiple of* $\mathrm{Det}(\Lambda(A))$. *Let $D_1 = D$, and $D_{i+1} = D_i/H_{ii}$ $i = 1, \ldots, n-1$.*

If $L = (L_{ij})$ *is any lower triangular matrix obtained from* A *by unimodular column operations followed by reduction modulo* D, *then* $H_{ii} = \gcd(D_i, L_{ii})$.

Proof. This proof is almost exactly as in [2]. Let $\gamma_i(M)$, $1 \le i \le n$, denote the greatest common divisor of all $i \times i$ subdeterminants obtained from the first i rows of a matrix M. Then it is easy to check that unimodular column operations leave this quantity unchanged. Moreover, reduction modulo D changes these γ_i's only by some multiple of D since it affects determinants linearly. Now $\gamma_i(H) = H_{11} \cdots H_{ii}$ divides $\mathrm{Det}(\Lambda(A))$ and so also D. Hence,

$$\begin{aligned} H_{11} \cdots H_{ii} &= \gcd(D, \gamma_i(H)) \\ &= \gcd(D, \gamma_i(A)) \\ &= \gcd(D, \gamma_i(L)) \\ &= \gcd(D, L_{11} \cdots L_{ii}). \end{aligned} \tag{3.4}$$

Thus, the claim of the lemma holds for $i = 1$. But $D_i = D/H_{11} \cdots H_{i-1,i-1}$, so dividing the $(i-1)$st instance of the above relation by $H_{11} \cdots H_{i-1,i-1}$ leaves

$$1 = \gcd(D_i, (L_{11} \cdots L_{i-1,i-1})/(H_{11} \cdots H_{i-1,i-1})), \tag{3.5}$$

for each $2 \le i \le n$. Consequently, dividing the ith instance of (3.4) by $H_{11} \cdots H_{i-1,i-1}$ and using (3.5), we deduce the lemma for each $i > 1$. □

The preceding lemma shows that Step 6 of the algorithm will correctly compute the diagonal entries of the Hermite normal form of A_1. It remains to see why the rest of the Hermite normal form is computed correctly in Step 6. This can be accomplished with the simple observations that after the columns of L are multiplied by the t_i's and reduced modulo $h(-d)$, we have a lower triangular matrix $\tilde{L}$ whose columns belong to $\Lambda(A_1)$ (see [2, Corollary 2.6]), and whose determinant equals $\mathrm{Det}(\Lambda(A_1))$. Hence, it follows that the columns of $\tilde{L}$ form a basis for $\Lambda(A_1)$. The rest of the procedure simply reduces the entries below the diagonal using elementary unimodular column operations, so it clearly produces the Hermite normal form.

It remains only to prove correctness of the algorithm for computing the Smith normal form of H. The proof follows very closely the arguments given above for the Hermite normal form, but we replace $\gamma_i(M)$ by $\tilde{\gamma}_i(M)$, which denotes the gcd of the determinants of *all* $i \times i$ submatrices of a matrix M. The crucial fact is that the $\tilde{\gamma}_i$'s are invariant under both elementary unimodular row and column operations. Thus,

$$\begin{aligned} S_{11} \cdots S_{ii} &= \gcd(h(-d), \tilde{\gamma}_i(S)) \\ &= \gcd(h(-d), \tilde{\gamma}_i(H)) \\ &= \gcd(h(-d), \tilde{\gamma}_i(\tilde{S})) \\ &= \gcd(h(-d), \tilde{S}_{11} \cdots \tilde{S}_{ii}). \end{aligned}$$

This observation is sufficient to complete the proof.

4. Running time

In this section,we analyze the running time of the algorithm given in §2. Our analysis is mostly self-contained, but any details that are omitted are contained in [11] or [18]. For example, in [11] it is shown that Steps 1 and 2 take at most $n^{1+o(1)}$ bit operations. Furthermore, it was shown that Step 4 can be done in $n^{4+o(1)}$ bit operations using Gaussian elimination and the Chinese remainder theorem.

We now estimate the running time for generating relations in Steps 3 and 5. If we use the binary method of exponentiation, then the arguments in [18] will show that the expected running time to test for a single relation in either Step 3 or Step 5 is $n^{1+o(1)}$. As the probability that a vector produces a relation is $L(d)^{-1/4z+o(1)}$, the expected running time to generate all the relations in Steps 3 and 5 is

$$(n+m)n^{1+o(1)}L(d)^{1/4z+o(1)}. \tag{4.1}$$

Since we argued in §3.2 that we can take $m = n^{1+o(1)}$, this gives an expected running time for Steps 3 and 5 of $n^{2+o(1)}L(d)^{1/4z+o(1)}$.

We next consider the running time for Step 6. In total, we need to perform at most $(n+m)n$ column operations to introduce zeros above the diagonal. Each of these column operations requires an application of the Euclidean algorithm on integers that are $\leq h_0$, followed by $O(n)$ arithmetic operations modulo h_0. Using fast multiplication methods, we can carry out the extended Euclidean algorithm in $O(\log h_0 \log\log^3 h_0)$ bit operations [15]. Each column operation can therefore be carried out in

$$O(\log h_0 \log\log^3 h_0 + n \log h_0 \log\log^2 h_0)$$

bit operations using fast integer multiplication [14]. Since $\log h_0 \leq n^{1+o(1)}$, we get a running time of $O(n^{4+o(1)})$ bit operations producing the Hermite normal form of A_1.

Finally, we consider the running time for Step 7. In order to create a matrix of the form (2.2), it requires $O(n \log d)$ unimodular row or column operations, each of which takes at most $O(n)$ operations modulo $h(-d)$ (or on integers of at most $O(\log d)$ bits). After doing this $O(\log d)$ times, we can guarantee that B_{11} divides every entry of B^*, so in order to produce the first diagonal entry of $\tilde{S}$, it takes at most $O(n^2 \log^4 d)$ bit operations. Hence, producing the full matrix $\tilde{S}$ takes at most $O(n^3 \log^4 d)$ bit operations. Since the reconstruction phase takes much less time than this, we construct the Smith normal form S in Step 7 in at most $O(n^3 \log^4 d)$ bit operations.

Hence, the total number of bit operations for the algorithm is bounded by

$$n^{2+o(1)}L(d)^{1/4z+o(1)} + n^{4+o(1)}.$$

As $n = L(d)^{z+o(1)}$, the optimal choice of z is $z = 1/\sqrt{8}$, and this proves Theorem 1.

5. Conjectural improvements

The above bound on the running time of our algorithm is not as good as the conjectured running time of Algorithm CN2 of [11]. However, there are a number of specific areas where the current rigorous algorithm might be improved.

First, any improvement would require a speedup in the running time for the generation of relations in Steps 3 and 5. This is due to the fact that, as it stands, these steps required $L(d)^{2z+1/4z+o(1)}$ operations, and this exponent has a minimum of $\sqrt{2}+o(1)$. Let ω_1 be the maximum number of nonzero elements in each of the first n relations generated in Step 3 and ω_2 the corresponding number for the m relations generated in Step 5. Clearly, then, we have $\omega_1 = O(\log^2 d)$ and $\omega_2 = O(n)$. If instead of trial division, we use the rigorous version of the elliptic curve factorization algorithm due to Pomerance [13], then the expected running time to test for a single relation in Step 3 should be $\omega_1 L(d)^{o(1)}$, and for a relation in Step 5 should be $\omega_2 L(d)^{o(1)}$. Then in place of (4.1), we would get an expected running time to generate all the relations in Steps 3 and 5 of

$$(n\omega_1 + n\omega_2)L(d)^{1/4z+o(1)}.$$

Hence, if one could show that the relations in Step 5 could be taken to be sparse (so that $\omega_2 = n^{o(1)}$), then by the analysis above, the running time would improve to $L(d)^{2/\sqrt{3}+o(1)}$. (Note that $2/\sqrt{3} \approx 1.155$.)

Similarly, we might be able to show that only $m = n^{o(1)}$ extra relations from Step 5 would be required. This would be possible if we could show that the group G described in §3 had significantly fewer than $O(\log|G|)$ primary invariants. There are at least two possible approaches to this. One would be to show that this group G by construction had this property. The other would be to show that G, in some quantitative sense, was random, in which case, with positive probability, it would have the desired property. In any case, this improvement would speed up the running time to $L(d)^{2/\sqrt{3}+o(1)}$ just as above. (Note that implementing both improvements together would not improve the overall running time because the time required for generating the first n relations would dominate.)

The two previous remarks have assumed no changes in the rest of the algorithm. However, if we could make either of the above improvements, then it seems likely that fast matrix multiplication techniques might be adapted to give a faster running time for computing $\det(A_0)$ and the determinant of the lattice $\Lambda(A_1)$. Using these ideas, we would obtain a running time of $L(d)^{c+o(1)}$ for computing $h(-d)$, where $c = (\theta+1)/2\sqrt{\theta}$, and θ is the exponent for matrix multiplication (currently, it is known that $\theta \leq 2.376$; see [1]).

Again with either of the above improvements, if it were possible to compute the determinant of a lattice generated by the columns of a nonsquare sparse matrix (perhaps by adapting Wiedemann's method) in $n^{2+\varepsilon}$ ring operations,

then the running time for computing $h(-d)$ would be reduce to $L(d)^{1+o(1)}$. This is highly speculative, but it is perhaps reasonable to conjecture this as the best running time that can be achieved with the framework of smooth forms. Note that this is a stronger conjecture that was made previously in [11].

REFERENCES

1. D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, preprint, 1987. Extended abstract in Proc. 19th ACM Sympos. on Theory of Comput. ACM, New York, 1987, pp. 1–6.
2. P. D. Domich, R. Kannan, and L. E. Trotter, *Hermite normal form computation using modulo determinant arithmetic*, Math. Oper. Res. **12** (1987), 50–59.
3. K. F. Gauss,*Disquisitiones arithmeticæ*, Fleischer, Lepzig, 1801. Translation into English by Arthur A. Clarke, S. J., reprinted by Springer-Verlag, New York, 1985.
4. D. Goldfeld, *Gauss' class number problem for imaginary quadratic fields*, Bull. Amer. Math. Soc. **13** (1985), 23–37.
5. T. C. Hu, *Integer programming and network flows*, Addison-Wesley, Reading, MA, 1969.
6. Hua Loo Keng, *Introduction to number theory*. Translation into English by Peter Shiu, Springer-Verlag, New York, 1982.
7. Ravindran Kannan and Achim Bachem, *Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix*, SIAM J. Comput. **8** (1979), 499–507.
8. D. E. Knuth, *The art of computer programming*, Vol. 2: *Seminumerical algorithms*, 2nd ed., Addison-Wesley, Reading, MA, 1981.
9. A. K. Lenstra and H. W. Lenstra, Jr., *Algorithms in number theory*, Technical Report 87–008, Dep. of Computer Science, Univ. of Chicago, 1987.
10. H. W. Lenstra, Jr., *On the calculation of regulators and class numbers of quadratic fields*, Journées Arithmétiques 1980 (J. V. Armitage, ed.), Cambridge Univ. Press, New York, 1982.
11. Kevin S. McCurley, *Cryptographic key distribution and computation in class groups*, Number Theory and Applications (Proc. NATO Advanced Study Inst. on Number Theory and Applications, Banff, 1988) (Richard A. Molin, ed.), Kluwer, Boston, 1989.
12. W. Narkiewicz, *Classical problems in number theory*, PWN, Polish Sci. Publ., Warsaw, 1986.
13. Carl Pomerance, *Fast, rigorous factorization and discrete logarithm algorithms*, Discrete Algorithms and Complexity (Proc. Japan-US Joint Seminar, June 1986, Kyoto, Japan), Academic Press, Orlando, 1987, pp. 119–143.
14. A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing **7** (1971), 281–292.
15. A. Schönhage, *Schnelle Berechnung von Kettenbruchenentwicklungen*, Acta Inform. **1** (1971), 139–144.
16. R. Schoof, *Quadratic fields and factorization*, Computation Methods in Number Theory (R. Tijdeman and H. W. Lenstra, Jr., eds.), Math. Centrum Tract 154, Amsterdam, 1982, pp. 235–286.
17. A. Schrijver, *Theory of linear and integer programming*, Wiley, New York, 1985.
18. Martin Seysen, *A probabilistic factorization algorithm with quadratic forms of negative discriminant*, Math. Comp. **48** (1987), 757–780.
19. Daniel Shanks, *Class number, a theory of factorization, and genera*, Proc. Sympos. Pure Math., Vol. 20, Amer. Math. Soc., Providence, RI, 1971, pp. 415–440.
20. ——, *Five number-theoretic algorithms*, Proc. Second Manitoba Conference on Numerical Mathematics, Univ. of Manitoba, Congressus Numerantium, No. VII, Utilitas. Math., Winnipeg, 1973, pp. 51–70.
21. Douglas H. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Trans. Inform. Theory **32** (1986), 54–62.

ABSTRACT. Let $C(-d)$ denote the Gauss Class Group of quadratic forms of a negative discriminant $-d$ (or equivalently, the class group of the imaginary quadratic field $Q(\sqrt{-d})$). We give a rigorous proof that there exists a Las Vegas algorithm that will compute the structure of $C(-d)$ with an expected running time of $L(d)^{\sqrt{2}+o(1)}$ bit operations, where $L(d) = \exp(\sqrt{\log d \log\log d})$. Thus, of course, also includes the computation of the class number $h(-d)$, the cardinality of $C(-d)$.

IBM RESEARCH DIVISION, ALMADEN RESEARCH CENTER, K53/802, 650 HARRY ROAD, SAN JOSE, CALIFORNIA 95120-6099

Current address (K. McCurley): Organization 1423, Sandia National Laboratories, Albuquerque, New Mexico 87185