MATHEMATICS OF OPERATIONS RESEARCH
Vol. 12, No. 3, August 1987
Printed in U.S.A.

MINKOWSKI'S CONVEX BODY THEOREM AND INTEGER PROGRAMMING*†

RAVI KANNAN

Carnegie-Mellon University

The paper presents an algorithm for solving Integer Programming problems whose running time depends on the number n of variables as $n^{O(n)}$. This is done by reducing an n variable problem to $(2n)^{5i/2}$ problems in $n - i$ variables for some i greater than zero chosen by the algorithm. The factor of $O(n^{5/2})$ "per variable" improves the best previously known factor which is exponential in n. Minkowski's Convex Body theorem and other results from Geometry of Numbers play a crucial role in the algorithm. Several related algorithms for lattice problems are presented. The complexity of these problems with respect to polynomial-time reducibilities is studied.

0. Introduction. The Integer Programming (feasibility) Problem is the problem of determining whether there is a vector of integers satisfying a given system of linear inequalities. In settling an important open problem, H. W. Lenstra (1979, 1983) showed in an elegant way that when n the number of variables is fixed, there is a polynomial time algorithm to solve this problem. He accomplishes this by giving a polynomial time algorithm that for any polytope P in $\mathscr{R}^n$ of nonzero volume either finds an integer point (point with all integer coordinates) in P or finds an integer vector v so that the maximum value of the linear functional (v, x) and the minimum value of (v, x) over the polytope P differ by less than c^{n^2} where c is a constant independent of n. Every integer point must lie on a hyperplane of the form $(v, x) = z$ for some integer z, and there are at most $c^{n^2} + 1$ such hyperplanes intersecting P. It obviously suffices to determine for each such hyperplane H, whether $H \cap P$ contains an integer point. Lenstra uses this to show that an n variable problem can be reduced to $c^{n^2} + 1$ problems each in $n - 1$ variables. This raises two questions: Can this factor of c^{n^2} be replaced by a polynomial? Can the reduction be done efficiently so as to achieve a better complexity for Integer Programming? Both these questions are, in effect, answered affirmatively in this paper.

The paper presents an algorithm which either finds an integer point in the given polytope P in $\mathscr{R}^n$ or finds for some i, $1 \leqslant i \leqslant n$, an $n - i$ dimensional subspace V with the following property: the number of translates of V containing integer points that intersect P is at most $(2n)^{5i/2}$. Each such translate leads to a $n - i$ dimensional problem. So, it can be shown that there is a factor of $O(n^{5/2})$ per variable in the running time. The algorithm for finding the subspace V uses at most $O(n^n s)$ arithmetic operations where s is the length of description of the polytope. The dependence on n of the complete integer programming algorithm is shown to be $O(n^{cn})$.

*This report was prepared with the support of a National Science Foundation grant ECS-8418392.
†Received March 10, 1986; revised November 25, 1986.
AMS 1980 subject classification. Primary: 90C10; Secondary: 68C25.
IAOR 1973 subject classification. Main: Programming/Integer.
OR/MS Index 1978 subject classification. Primary: 639 Programming/Integer/Algorithms.
Key words. Integer programming, computational complexity, geometry of numbers, lattice problems.

0364-765X/87/1203/0000$01.25

Several concepts and results from Geometry of Numbers are used, the most crucial of them being Minkowski's convex body theorem. This elegant classical theorem turns out to be crucial in reducing the factor per variable to a polynomial. §1 contains a brief introduction to Geometry of Numbers to make the paper self-contained.

The integer programming algorithm will be presented after two other algorithms: one for finding the shortest (in Euclidean length) nonzero integer linear combination of a given set of vectors and the other for finding the integer linear combination of a set of vectors that is closest (in Euclidean distance) to another given vector. These are called the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) respectively. The algorithms for both problems take $O(n^n s)$ arithmetic operations on n dimensional problems where s is the length of the input. The algorithm for the SVP is needed as a subroutine in the integer programming whereas the algorithm for the CVP is not directly needed, but has ideas that will be useful in integer programming. The integer programming algorithm extends to mixed integer programs too.

It is well known that Integer Programming is NP-hard. It has been shown recently that CVP is NP-hard. At present, it is not known whether SVP is NP-hard or admits a polynomial time algorithm (or both!). The last section of the paper provides another, more natural proof that CVP is NP-hard. Further, it relates the complexity of the SVP to an approximate version of the CVP. It is hoped that this is a beginning towards proving the NP-hardness of the SVP which remains an important open problem.

Operations Researchers are usually interested in solving the Integer Programming Optimality problem—i.e., the problem of maximizing a linear function over the set of integer solutions (solutions with all integer coordinates) to a system of linear inequalities. This question can be reduced by elementary means to the Integer Programming feasibility question which is the problem of determining whether there is an integer point inside a given poyhedron. This paper deals only with the feasibility question and this will be called **the Integer Programming Problem**. Computationally it can be stated as: Given $m \times n$ and $m \times 1$ matrices A and b respectively of integers, find whether there exists an $n \times 1$ vector x of integers satisfying the m inequalities $Ax \leqslant b$. The case of $n = 1$ can be trivially solved in polynomial time. For the case of $n = 2$, Hirschberg and Wong (1976), Kannan (1980) and Scarf (1981) gave polynomial time algorithms.

Central to H. W. Lenstra's algorithm for general n is an algorithm for finding a "reduced basis" of a "lattice" (both terms to be explained later). Lenstra's (1979) original basis reduction algorithm takes polynomial-time only when the number of dimensions is fixed. After his result, Lovász devised a basis reduction algorithm which runs in polynomial time even when n the number of dimensions varies. This algorithm combined with an earlier result of A. K. Lenstra's (1981) that reduced factoring a polynomial to finding a short vector in a lattice yields a polynomial time algorithm for factoring polynomials over the rationals. All these ideas were first published in an important paper of Lenstra, Lenstra and Lovász (1983). This paper is referred to henceforth as the LLL paper. Here, the following result from the LLL paper is used: Given a set of vectors $b_1, b_2, \ldots, b_n$, we can find in polynomial time a nonzero integer linear combination of them whose length is at most $2^{n/2}$ times the length of any (other) nonzero integer linear combination. §2 of this paper gives an algorithm for finding a better reduced basis of a lattice than the one Lovász's algorithm finds, but in time bounded by a polynomial only for fixed n. The reduced basis solves the SVP and is used in other algorithms of the paper.

A preliminary version of this paper appeared as Kannan (1983). Since then, Helfrich (1985) has made some improvements in the running time of some of the algorithms. Schnorr (1984) uses the algorithm presented here for solving the SVP to obtain *polynomial time* algorithms for finding better approximations to the shortest vector than the LLL paper. Lagarias, Lenstra and Schnorr (1986) prove some properties of the successive minima of lattices for the concept of "reduced basis" used in this paper. They have traced the concept back to Korkhine and Zolotoreff (1873). Hastad (1985) has observed using their results that for any polytope P of positive volume in $\mathscr{R}^n$, if P does not contain an integer point, then there *exists* an integer vector v so that the maximum and minimum of (v, x) over P differ by at most $O(n^{5/2})$. But there is no finite algorithm known that finds this vector. Babai (1985) is an interesting related development to some of the algorithmic questions discussed in this paper.

Notation. $\mathscr{R}^n$ is Euclidean n-space. $\mathscr{Z}^n$ is the set of n-vectors with integer components. (a, b) is the dot product of the two vectors a, b. $|a| = |a|_2 =$ the Euclidean length of the vector a.

$L(b_1, b_2, \ldots, b_n) =$ the lattice generated by the vectors $b_1, b_2, \ldots, b_n$ (i.e., the set of all integer linear combinations of these vectors). For any lattice L, $\Lambda_1(L)$ denotes the length of the shortest nonzero vector in L.

For any set of vectors $b_1, b_2, \ldots, b_n$, we reserve the notation $b_i(j)$ for the real numbers defined in (1.5) and $b(i, j)$ for the vectors defined in (1.5)$'$.

Suppose $L(b_1, b_2, \ldots, b_n)$ is a lattice. Then for $j = 1, 2, \ldots, n$, $L_j(b_1, b_2, \ldots, b_n)$ will denote the projection of $L(b_1, b_2, \ldots, b_n)$ orthogonal to the vector space spanned by $b_1, b_2, \ldots, b_{j-1}$. By convention we take the space spanned by the empty set to be the singleton $\{0\}$ and hence the orthogonal complement of it is the whole space. Thus, $L_1(b_1, b_2, \ldots, b_n) = L(b_1, b_2, \ldots, b_n)$. Clearly $L_j(b_1, b_2, \ldots, b_n)$ depends on the basis $b_1, b_2, \ldots, b_n$ of the lattice we choose.

The programs in this paper will be written in "pidgin" ALGOL. The language is close enough to English that the reader should have no problem with it. I adopt the convention that the statement "Return x" means Stop execution and output x.

1. Basic definitions and facts about lattices.

A *lattice* L in $\mathscr{R}^n$ is the set of all integer linear combinations of a set of linearly independent vectors in $\mathscr{R}^n$. The independent vectors are called a *basis* of the lattice.

If $b_1, b_2, \ldots, b_n$ are independent vectors in $\mathscr{R}^m$, $m \geqslant n$, the basis matrix of the lattice $L(b_1, b_2, \ldots, b_n)$ is the $n \times m$ matrix B with $b_1, b_2, \ldots, b_n$ as its n rows. Now suppose U is any $n \times n$ unimodular matrix (integer matrix with determinant ± 1). Clearly, the inverse of U exists and has integer entries. Then for any y in $\mathscr{R}^m$, y is in $L(b_1, b_2, \ldots, b_n)$ iff $\exists x \in \mathscr{Z}^n$: $y = xB \Leftrightarrow \exists x' \in \mathscr{Z}^n$: $x'(UB) = y$ (because U, U^{-1} have integer entries) $\Leftrightarrow y \in$ the lattice generated by the rows of UB. Thus making a unimodular transformation of the basis leaves the lattice unchanged. Indeed the converse is also true.

LEMMA 1.1. *Suppose B and B' are $n \times m$ and $k \times m$ matrices each with independent rows and suppose the rows of B and B' generate the same lattice. Then k equals n and there is a unimodular matrix U such that $UB = B'$.*

The dimension of a lattice is the number of basis vectors that generate it. If a lattice is full dimensional, i.e., it is a lattice in $\mathscr{R}^n$ of dimension n and is generated by the rows of an $n \times n$ matrix B, the determinant of the lattice is defined to be the absolute value of the determinant of B (by the lemma above it is an invariant of the lattice). Geometrically, it is the volume of the parallelepiped spanned by $b_1, b_2, b_3, \ldots, b_n$. We also have to deal with lattices which are not full dimensional. Thus suppose $b_1, b_2, \ldots, b_n$ are independent vectors in $\mathscr{R}^m$, $m \geqslant n$. Then the determinant of $L(b_1, b_2, \ldots, b_n)$ is defined to be the n volume of the n-dimensional parallelepiped spanned by $b_1, b_2, \ldots, b_n$. To make this definition computationally more explicit as well for other purposes, we introduce the familiar Gram-Schmidt orthogonalization process for finding $b_1^*, b_2^*, \ldots, b_n^*$ where,

$$b_1^* = b_1 \quad \text{and} \tag{1.2}$$

$$b_{i+1}^* = b_{i+1} - \sum_{j=1}^{i} \mu_{i+1,j} b_j^* \quad \text{for } i = 1, \ldots, n-1, \quad \text{where} \tag{1.3}$$

$$\mu_{kl} = (b_k, b_l^*)/(b_l^*, b_l^*) \quad \text{for } 1 \leqslant l < k \leqslant n. \tag{1.4}$$

We see that if $b_1, \ldots, b_n$ have rational coordinates, so do the b_i^* and they can be computed in polynomial time from $b_1, b_2, \ldots, b_n$. Letting $u_i = b_i^*/|b_i^*|$, there exist real numbers $b_i(j)$ such that

$$b_i = \sum_{j=1}^{n} b_i(j) u_j. \tag{1.5}$$

In fact, from (1.3), we see that $b_i(j) = \mu_{ij}|b_j^*|$ for $1 \leqslant j < i \leqslant n$, $b_i(i) = |b_i^*|$ for all i and $b_i(j) = 0$ for $1 \leqslant i < j \leqslant n$. So with rational inputs, $b_i(j)^2$ is always rational (even though $b_i(j)$ may not be). This observation will be useful because I will have occasion to compare $|b_i(j)|$ with other real numbers. The definition of $b_i(j)$ in (1.5) will be used repeatedly and so it is part of the notation. I will also use occasionally the vectors $b(i, j)$ defined by (1.5)′ below:

$$b(i, j) = \sum_{k=j}^{i} b_i(k) u_k \quad \text{for } 1 \leqslant j \leqslant i \leqslant n. \tag{1.5$'$}$$

In many parts of the paper, it will be extremely useful to think of $b_1, b_2, \ldots, b_n$ as being represented in a coordinate system with $u_1, u_2, \ldots, u_n$ as the axes vectors. In this coordinate system, the matrix with the basis vectors as its rows is lower triangular and has the $b_i(j)$ of (1.5) as its entries. Reminder: $b_i(i)$ is the length of b_i^*.

$$\begin{pmatrix}
b_1(1) & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\
b_2(1) & b_2(2) & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & b_i(i) & 0 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & b_{i+1}(i) & b_{i+1}(i+1) & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\
b_n(1) & b_n(2) & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & b_n(n)
\end{pmatrix}.$$

The lower triangular representation of the basis matrix. I caution that these entries may be irrational and cannot be exactly computed in general. So, in the algorithms I do not change the coordinate system, but conceptually it is easier to think of the basis matrix being written in this form.

The determinant of $L(b_1, b_2, \ldots, b_n)$ denoted $d(L(b_1, b_2, \ldots, b_n))$ is defined to be the absolute value of the determinant of the lower triangular $n \times n$ matrix whose entries are $b_i(j)$. Clearly, this equals the product of the lengths of the b_i^*, $i = 1, 2, \ldots, n$.

We are often interested in "projecting" and "lifting" vectors. Projecting a vector b onto the hyperplane through the origin with v as the normal yields the vector

$b - ((b, v)/(v, v))v$. The projection of b in the direction of v is the vector $((b, v)/(v, v))v$. To project perpendicular to a subspace we find an orthogonal basis of the subspace and project perpendicular to each basis vector successively—this is the Gram-Schmidt procedure described in (1.2) and (1.3). To project into a subspace means to project perpendicular to its orthogonal complement. The projection of a set is the set of projections of its elements. Suppose v is a nonzero element of a lattice L and $\hat{L}$ is the projection of L perpendicular to v. If $\hat{w}$ is any vector in the $\hat{L}$, we may "lift" it to a vector in L as follows: it is easy to see that there is a unique vector w in L such that w projects into $\hat{w}$ and $(w, v) \in (-(v, v)/2, (v, v)/2]$. To see this note that we may take any vector u which projects into $\hat{w}$ and add a suitable integer multiple of v into u to get a u' whose projection in the direction of v has length at most $|v|/2$ which is exactly what the dot product condition above stipulates. Indeed, let $r = [(u, v)/(v, v)]$ where $[x]$ stands for the integer nearest to the real number x. Let $u' = u - rv$. Then $|(u', v)| \leqslant (1/2)(v, v)$. The process described here will be called *lifting* $\hat{w}$ to w.

We need two facts from basic Geometry of Numbers.

PROPOSITION 1.6. *Suppose v is a nonzero element of the lattice L, such that λv does not belong to the lattice for any λ in $(0, 1)$. Then there is a basis of the lattice containing v. (Such a vector v is called primitive.)*

PROPOSITION 1.7. *The following "algorithm" yields a basis $b_1, b_2, \ldots, b_n$ of the lattice L.*

```
Procedure. Input lattice $L$ of dimension $n$.
$b_0 = 0$
do for $i = 1$ to $n$ by 1:
    Pick any nonzero $v$ such that $(v, b_j) = 0$ for $j = 0, 1, \ldots, i - 1$.
    Find the smallest positive real $\lambda$ such that $\lambda v$ is in the lattice $\hat{L}$ obtained by
    projecting $L$ into the orthogonal complement of the span of $\{b_1, \ldots, b_{i-1}\}$.
    Find $w$ in $L$ such that $w$ projects into $\lambda v$ in $\hat{L}$.
    $b_i = w$.
end
return $(b_1, \ldots, b_n)$. ■
```

Of course the method presented here is not quite an algorithm—we do not know how the input is specified etc. I will later describe a more rigorous version of this algorithm called *SELECT–BASIS in* §2.

THEOREM 1.8 (Minkowski). *If S is any convex set in $\mathscr{R}^n$ which is symmetric about the origin $(x \in S \Rightarrow -x \in S)$ and has volume greater than 2^n, then S contains a nonzero point of $\mathscr{Z}^n$.*

PROOF. Define $S/2 = \{x\colon\ x \in \mathscr{R}^n,\ 2x \in S\}$. Clearly, $S/2$ has volume greater than 1. Consider the convex bodies $v + S/2 = \{x\colon\ x \in \mathscr{R}^n, x = v + s$ for some $s \in S/2\}$ as v ranges over $\mathscr{Z}^n$. There is one such body for each point of $\mathscr{Z}^n$ and their volumes are strictly greater than 1. Therefore, two of them must intersect. (I leave it to the reader to make this intuitive argument into a rigorous one.) Suppose $v + S/2$ and $u + S/2$ intersect, then so do $S/2$ and $(u - v) + S/2$. Let y be in their intersection. Then y and $y - u + v$ both belong to $S/2$. So, $2y$, $2(y - u + v)$ both belong to S. The symmetry of S implies that $-2y$ belongs to it, the convexity then implies that the average of $-2y$ and $2(y - u + v)$ which is $v - u$ belongs to S. Of course, $v - u$ is in $\mathscr{Z}^n$ proving the theorem. ■

I will generally only use the following direct consequence of Minkowski's theorem.

THEOREM 1.9. *If L is an n-dimensional lattice with determinant $d(L)$, there is a nonzero element v of L with $|v| \leqslant \sqrt{n}\,(d(L))^{1/n}$.*

PROOF. Let $L = \mathscr{Z}^n B = \{x\colon x = yB \text{ for some } y \in \mathscr{Z}^n\}$, B an $n \times n$ matrix with a basis of L as its rows. Consider the solid sphere T with the origin as center and radius $\sqrt{n}\,(d(L))^{1/n}$. T has volume $\pi^{n/2}/\Gamma(n/2+1)R^n$ where R is its radius. So the volume of T is greater than $2^n d(L)$. T is convex and symmetric about the origin. Hence so is $TB^{-1} = \{x\colon \exists y \in T \text{ s.t. } x = yB^{-1}\}$. TB^{-1} has volume greater than $2^n d(L) \cdot \det(B^{-1}) = 2^n$. Thus there is a y in $\mathscr{Z}^n - \{0\} \cap TB^{-1}$. $v = yB$ is then a nonzero element of $T \cap L$. Clearly v is short enough to prove the theorem. ∎

REMARK 1.10. The factor $\sqrt{n}$ in the theorem can be improved by reckoning the volume of the n-sphere more accurately. In fact, more sophisticated upper bounds on $\Lambda_1(L)/(d(L))^{1/n}$ are known. This is of course the ratio theorem, 1.9 is bounding from above. The supremum value of the square of this ratio over all n-dimensional lattices is called Hermite's constant and the best known asymptotic upper bound on it is $n(1+o(1))/e\pi$ due to Blichfeldt (1929). See also Lekkerkerker (1969, §38).

The general references on the subject of Geometry of Numbers are Cassels (1959) and Lekkerkerker (1969). An expository survey of lattice algorithms can be found in Kannan (1987).

2. The algorithm for finding the shortest vector.

In this section, I describe an algorithm to find a shortest nonzero[1] vector in a lattice L given by a basis $b_1, b_2, \ldots, b_n$. This algorithm actually finds a "reduced basis" of the lattice of which the first vector will be the shortest vector in the lattice (the definition of a reduced basis used in this paper is found in 2.6). Here is how the algorithm works: Using polynomially (in n alone) recursive calls to lower dimensional subroutines, the algorithm finds a basis $a_1, a_2, \ldots, a_n$ for the lattice L which satisfies the following properties:

$$\text{For } j = 2,3,\ldots,n, \quad a_j(j) = \Lambda_1\big(L_j(a_1, a_2, \ldots, a_n)\big),^2 \tag{2.1}$$

$$|a_1| \leqslant \frac{2}{\sqrt{3}}|a_2|, \tag{2.2}$$

$$|a_2(1)| \leqslant |a_1|/2. \tag{2.3}$$

Intuitively, these conditions can be understood by appealing to the representation of the basis $a_1, a_2, \ldots, a_n$, as a lower triangular matrix. In such a representation, condition (2.1) says that for $j = 2,3,\ldots,n$, the jth diagonal entry is the length of the shortest vector in the lattice generated by the rows of the $(n-j+1) \times (n-j+1)$ matrix consisting of the last $n-j+1$ rows and columns of the basis matrix.

Whereas in the reduced basis of LLL (1982), the length of the first vector is guaranteed to be at most $2^{n/2}d(L)^{1/n}$, for the basis here, one can prove (2.4) below using Minkowski's theorem, conditions (2.1), (2.2) and (2.3) and the fact that $d(L) = |a_1| d(L_2(a_1, a_2, \ldots, a_n))$.

$$|a_1| \leqslant \sqrt{2n}\, d(L)^{1/n}. \tag{2.4}$$

[1]Henceforth, I will use the phrase "shortest vector" for "shortest nonzero vector" when the meaning is clear.

[2]See notation in the introduction for the definition of $L_j(a_1, a_2, \ldots, a_n)$ and Λ_1 (a lattice).

In other words, our a_1 is much a shorter vector than theirs—but of course we will spend more time finding it. (This inequality will be proved later.)

Having obtained such a basis $a_1, a_2, \dots, a_n$, I show that the shortest vector in the lattice must be of the form $y = \sum_{i=1}^{n} \alpha_i a_i$ where $(\alpha_1, \alpha_2, \dots, \alpha_n) \in T$ where T is a subset of $\mathscr{Z}^n$ of cardinality at most

$$\frac{3^n |a_1|^n}{d(L)}. \tag{2.5}$$

(2.4) is used to bound the expression (2.5) in terms of n alone. We enumerate all elements of T, find the corresponding y and take the shortest of these which must then be the shortest vector in the lattice. We find an entire reduced basis instead of just the shortest vector to facilitate the recursion. First, here is the definition of reduced basis with which we will work.

DEFINITION 2.6. A basis $v_1, v_2, \dots, v_n$ of the lattice $L(v_1, v_2, \dots, v_n)$ is called a *reduced basis* if (2.7) and (2.8) below are satisfied.

$$\text{For } j = 1, 2, \dots, n, \quad v_j(j) = \Lambda_1\big(L_j(v_1, v_2, \dots, v_n)\big),^3 \tag{2.7}$$

$$|v_i(j)| \leqslant v_j(j)/2 \quad \text{for } i \geqslant j + 1 \geqslant 2. \tag{2.8}$$

Note the difference between (2.1) and (2.7) is that (2.7) includes $j = 1$ also whereas (2.1) does not. Thus in the lower triangular representation, every diagonal entry is the length of the shortest vector in the lattice generated by the rows of the square submatrix of which it is the top left entry. The essential feature of the LLL reduced basis is that in the lower triangular representation, the jth diagonal entry is the length of the shortest vector in the 2-dimensional lattice generated by the rows of the submatrix containing the rows $j, j + 1$ and columns $j, j + 1$ of the basis matrix. Schnorr (1984) generalizes the LLL reduced basis to allow $k \times k$ submatrices for any fixed k. (Schnorr's algorithm uses the algorithm *SHORTEST* of this section as a subroutine to make the $k \times k$ matrices reduced in the sense defined here.)

A detailed description of the algorithm *SHORTEST* is given below followed by a proof of correctness and bounds on the running time.

```
Procedure SHORTEST ($n; b_1, b_2, \dots, b_n$).
  Comment. The preceding paragraphs explain what the algorithm accomplishes.
$L = L(b_1, b_2, \dots, b_n)$. The procedure finds a basis of $L$ satisfying (2.7) and (2.8).
    1. If $n = 1$ then return $\{b_1\}$.
    2. Use the basis reduction algorithm from Lenstra, Lenstra and Lovász to make
       the basis reduced in their sense.
    3. $\bar{b}'_i \leftarrow$ projection of $b_i$ perpendicular to $b_1$ for $i = 2, 3, \dots, n$.
    4. $\bar{b}_2, \bar{b}_3, \dots, \bar{b}_n \leftarrow SHORTEST(n - 1; \bar{b}'_2, \bar{b}'_3, \dots, \bar{b}'_n)$.
    5. For $i = 2$ to $n$, lift $\bar{b}_i$ to $b_i$ in $L$. (Cf. §1. The proof of Theorem 3.9 contains
       the mundane details of how the lifting is done.)
  Comment. We now have a basis of $L$ satisfying conditions (2.1) and (2.3).
    6. If $|b_2| < \sqrt{3}|b_1|/2$ then swap $b_1$ and $b_2$ and Goto 3; otherwise continue.
  Comment. We now satisfy condition (2.2) also. Caution: $|b_1|, |b_2|$ may be irra-
tional, but their squares are not. So we use them instead.
```

³Thus v_1 is a shortest vector in the whole lattice.

```
7. If $|b_j(j)| \geqslant |b_1|$ for some $j > 1$, then $j_0 \leftarrow$ minimum such $j$, else $j_0 \leftarrow n + 1$.
8. $BASIS \leftarrow \{b_1, b_2, \ldots, b_{j_0 - 1}\}$.
Comment. I show later that some nonzero shortest vector of $L$ is in $L(BASIS)$.
9. Call $ENUMERATE(BASIS)$ to obtain a shortest nonzero vector $v_1$ in
   $L(BASIS)$.
Comment. This procedure is explained later.
10. $\{b_1, b_2 \ldots b_n\} \leftarrow SELECT\text{-}BASIS(n; v_1, b_1, b_2 \ldots b_n)$.
Comment. Procedure explained later. It returns a basis of $L$ containing $v_1$ as the first vector—cf. Proposition 1.6.
11. Execute steps 3, 4 and 5 and return $\{b_1, b_2, \ldots b_n\}$.
    end SHORTEST
```

Procedure $SELECT\text{-}BASIS(n; b_1, b_2, \ldots, b_{n+1})$.

Comment. $b_1, b_2, \ldots, b_{n+1}$ are vectors in $\mathcal{Q}^k$ for some $k \geqslant n$ and span an n-dimensional subspace of $\mathcal{R}^k$. The procedure returns a basis $a_1, a_2, \ldots, a_n$ of $L = L(b_1, b_2, \ldots, b_{n+1})$. It first finds a shortest lattice vector in the direction of b_1—call this a_1. Then it projects L orthogonal to a_1 to get a lattice $\bar{L}$. It works by recursively finding a basis of $\bar{L}$ (cf. Proposition 1.7).

```
1. If $n = 0$ or $b_1 = 0$ then do the obvious.
2. If $b_1$ is linearly independent of $b_2, \ldots, b_{n+1}$
   then $a_1 \leftarrow b_1$
   else do:
        3. Find $\alpha_2, \ldots, \alpha_{n+1}$ (rationals—these are unique) such that $\Sigma_{j=2}^{n+1}\alpha_j b_j = b_1$.
        4. $M \leftarrow$ least common multiples of the denominators of $\alpha_2, \ldots, \alpha_{n+1}$.
        5. $\gamma \leftarrow GCD(M\alpha_2, M\alpha_3, \ldots, M\alpha_{n+1})$.
        6. Let $M/\gamma = p/q$ where $p, q$ are relatively prime integers. $a_1 \leftarrow (1/q)b_1$.
   end
7. $\bar{b}_i \leftarrow$ projection of $b_i$ perpendicular to $a_1$ for $i = 2, 3, \ldots, n + 1$,
8. $\{c_2, c_3, \ldots, c_n\} \leftarrow SELECT\text{-}BASIS(n - 1; \bar{b}_2, \ldots, \bar{b}_{n+1})$.
9. Lift $c_i$ to $a_i$ in $L$ for $i = 2, 3, \ldots, n$ and return $\{a_1, a_2, \ldots, a_n\}$.
end SELECT-BASIS.
```

PROPOSITION 2.9. *The basis $a_1, a_2, \ldots, a_n$ returned by the above procedure SELECT–BASIS is a basis of $L = L(b_1, b_2, \ldots, b_{n+1})$ assuming that $b_1, b_2, \ldots, b_{n+1}$ span an n-dimensional subspace.*

PROOF. By Proposition 1.7, it suffices to show that a_1 is a shortest vector of L in the direction of b_1 provided b_1 is not equal to zero. This is easily seen to be true. ∎

PROPOSITION 2.10. *The vectors returned by the procedure SHORTEST $(n; b_1, b_2, \ldots, b_n)$ form a basis of $L(b_1, b_2, \ldots, b_n)$.*

PROOF. Follows easily by induction on n on the lines of Proposition 1.7. ∎

PROPOSITION 2.11. *Let j_0 be as defined in step 7 of SHORTEST. Then a shortest vector of $L(b_1, b_2, \ldots, b_{j_0-1})$ is also a shortest vector of $L(b_1, b_2, \ldots, b_n)$.*

PROOF. Suppose $v = \Sigma_{i=1}^{n}\alpha_i b_i$ is a shortest nonzero vector of $L(b_1, b_2, \ldots, b_n)$ and one of $\alpha_{j_0}, \alpha_{j_0+1}, \ldots, \alpha_n$ is nonzero. Then the projection v' of v into the vector space $V =$ the orthogonal complement of Span $(b_1, b_2, \ldots, b_{j_0-1})$ is nonzero. Therefore by (2.1), we must have

$$|v'| \geqslant \Lambda_1\left(L_{j_0}(b_1, b_2, \ldots, b_n)\right) = b_{j_0}(j_0).$$

Then clearly, $|v| \geqslant |v'| \geqslant b_{j_0}(j_0) \geqslant |b_1|$. Thus b_1 is a shortest vector of L. ∎

PROPOSITION 2.12. *The procedure SHORTEST executes recursive call of step* 4 *at most* $(5/2)n$ *times when started on an n-dimensional lattice.*

PROOF. By Lenstra, Lenstra and Lovász, the execution of their basis reduction algorithm in step 2 of procedure *SHORTEST* yields a basis of L with $|b_1| \leqslant 2^{n/2}\Lambda_1(L)$. Each execution but the first of the loop steps 3–6 of *SHORTEST* cuts down $|b_1|$ by a factor of at least $\sqrt{3}/2$. Thus each 5 iterations of the loop cuts it down by a factor of at least 2. We cannot reduce $|b_1|$ further once it reaches $\Lambda_1(L)$. Thus at most $5n/2$ executions of the loop suffice. ∎

Description of procedure ENUMERATE. The crucial reason that we can complete the recursion is that we can enumerate relatively few candidates to determine the shortest vector. This fact is proved now. Suppose $j_0 - 1 = m$ in step 7 of procedure *SHORTEST* and suppose a shortest vector of $L(b_1, b_2, \dots, b_m)$ is $y = \sum_{i=1}^{m}\alpha_i b_i$. Then since y must be of length at most $|b_1|$, the projection of y into V_m, the orthogonal complement in $\mathbf{R}^n$ of the span of $\{b_1, b_2, \dots, b_{m-1}\}$ must be of length at most $|b_1|$. This projection has length $|\alpha_m b_m(m)|$, so we must have

$$|\alpha_m| \leqslant |b_1| / |b_m(m)|.$$

More generally, we have the following proposition. The reader might want to use the lower triangular representation of the basis matrix to understand the proposition.

PROPOSITION 2.13. *With the above notation, suppose $\beta_{i+1}, \beta_{i+2}, \dots, \beta_m$ are fixed integers. Then there is an easily computed number β_i^0 such that for all integers $\alpha_1, \alpha_2, \dots, \alpha_{i-1}$ and β_i,*

$$\left|\sum_{j=1}^{i-1}\alpha_j b_j + \beta_i b_i + \sum_{j=i+1}^{m}\beta_j b_j\right| \leqslant |b_1| \Rightarrow \beta_i \in \left[\beta_i^0 \quad \beta_i^0 + 2\frac{|b_1|}{|b_i(i)|}\right].$$

PROOF. For any vector v, I denote by $\hat{v}$, the projection of v along the direction of b_i^* in this proof. Let $u = \sum_{j=i+1}^{m}\beta_j b_j$ and $w = \sum_{j=1}^{i-1}\alpha_j b_j + \beta_i b_i + u$. Clearly, $\hat{w} = \beta_i b_i^* + \hat{u} = \beta_i b_i^* + t b_i^*$ (*say*) where t is some fixed real number (since $\beta_{i+1}, \beta_{i+2}, \dots, \beta_m$ and hence u are fixed). So we have

$$|w| \leqslant |b_1| \Rightarrow |\hat{w}| \leqslant |b_1| \Rightarrow |(\beta_i + t)| \leqslant |b_1|/|b_i^*| \Rightarrow -t - \frac{|b_1|}{|b_i^*|} \leqslant \beta_i \leqslant -t + \frac{|b_1|}{|b_i^*|}.$$

So the proposition follows with

$$\beta_i^0 = -t - \frac{|b_1|}{|b_i^*|}.$$ ∎

It is clear that we can write a procedure to determine a list T of candidates $(\alpha_1, \alpha_2, \dots, \alpha_n)$ for the shortest vector $\sum\alpha_i b_i$.

LEMMA 2.14. *At the end of the procedure, $|T|$ is at most $\prod_{i=1}^{m}(1 + 2|b_1|/b_i(i))$ which is at most $(18n)^{n/2}$.*

PROOF. The first part follows from the last proposition. For the second part, we note that using $|b_1| \geqslant b_i(i)$,

$$\prod_{i=1}^{m}(1 + 2|b_1|/b_i(i)) \leqslant 3^m\prod_{i=1}^{m}(|b_1|/b_i(i)).$$

The denominator $\prod_{i=1}^{m} b_i(i)$ is of course the determinant of the lattice $L(b_1, b_2, \dots, b_m)$—call it L_1 for short in the rest of the proof. Also, let $L_2 = L_2(b_1, b_2, \dots, b_m)$ for this proof. Since $b_2(2) = \Lambda_1(L_2)$,

$$\frac{b_2(2)^{m-1}}{d(L_2)} \leqslant \left(\sqrt{m-1}\right)^{m-1}$$

(by Minkowski's Theorem 1.9).

Further, $|b_1| \leqslant 2|b_2|/\sqrt{3}$ and $|b_2(1)| \leqslant |b_1|/2$ imply that $\sqrt{2}\,b_2(2) \geqslant |b_1|$. Hence, using $d(L_1) = d(L_2)|b_1|$, we have (cf. (2.4))

$$|b_1|^m \leqslant (2m)^{m/2} d(L_1).$$

Thus the lemma follows by a simple calculation.

PROPOSITION 2.15. *The basis $b_1, b_2, \dots, b_n$ returned by SHORTEST satisfies the conditions* (2.7) *and* (2.8).

PROOF. Induction on n. $n = 1$ is obvious. By Proposition 2.11, the shortest vector of $L(BASIS)$ in step 9 is also the shortest vector of the whole n-dimensional lattice. It is clear that the vector v_1 at the end of step 9 is indeed a shortest vector of the lattice. Using proposition 2.9 and the inductive assumption on step 11, the current lemma follows. ■

This completes the proof of correctness. As for the time bound, I will split it into two parts: a bound on the number of arithmetic operations—additions, subtractions, multiplications, divisions and comparisons with operands that are rational numbers, and a bound on the operand sizes. The number of arithmetic operations will depend on the dimension n of the problem as well as the length s of the input. However, going through the procedure *SHORTEST* step by step, we see that the total number of arithmetic operations performed *while the procedure is not inside a call to LLL basis reduction algorithm* is bounded by a function of n alone—it does not depend on s. This is seen by an inductive proof using Proposition 2.12. Unfortunately, the same does not hold for LLL. In the next section (Proposition 3.8), I show that the total number of arithmetic operations performed by *SHORTEST* in all the calls to LLL in $n^n s$. For now, I will assume this proposition.

THEOREM 2.16. *$SHORTEST(n; \dots)$ finds a reduced basis satisfying* (2.7) *and* (2.8) *in $O(n^n s)$ arithmetic operations where s is the length of the input.*

Note. In common usage, we might call this a $O(n^n s)$-algorithm. This, however, counts only the number of arithmetic operations, and ignores the size of the operands. In an algorithm such as this one which manipulates numbers and keeps them all precisely, it is important to prove bounds on the size of the numbers. I do so in the next section.

PROOF. Let $T(n)$ be the maximum number of arithmetic operations performed by $SHORTEST(n; \dots)$ while not inside a call to LLL. It is easily seen that all steps of the algorithm except recursive calls to shortest, the enumeration and calls to LLL call for a number of arithmetic operations bounded by a polynomial in n alone. Thus we have (by Proposition 2.12 and Lemma 2.14),

$$T(n) \leqslant \frac{5n}{2} T(n-1) + (18n)^{n/2} q(n),$$

(q-a polynomial). I will derive the bound $T(n) \in O(n^n)$. $\mathrm{Lim}_{n\to\infty}(1 - 1/n)^{n-1} = 1/e$

and $5/2e$ is less than 0.93, so there exists an N_1 such that for all $n > N_1$, we have $(5/2)(1 - 1/n)^{n-1} \leqslant 0.95$. Further, let N_2 be a natural number so that $\forall n \geqslant N_2$, $(18n)^{n/2}q(n) \leqslant 0.05n^n$. Let N be the maximum of N_1, N_2. Choose a constant $c \geqslant 1$ such that $T(n) \leqslant cn^n$ $\forall n \leqslant N$. Now, I argue by induction on n that $T(n) \leqslant cn^n$ for all n. For $n \leqslant N$, this is true by definition. So, assume $n > N$ and suppose it is true for $n - 1$. Then

$$T(n)/(cn^n) \leqslant (5/2)(1-(1/n))^{n-1} + (18n)^{n/2}q(n)/(cn^n) \leqslant 0.95 + 0.05 = 1.$$

This completes the inductive proof. The total number of arithmetic operations performed by the algorithm is $T(n)$ + the number of operations performed while executing calls to LLL. From Proposition 3.8, then the current theorem follows. ■

REMARK 2.17. Here, I considered the shortest vector in the Euclidean (L_2) norm. We can also define the shortest vector according to other norms in the obvious fashion. To find the L_1 shortest vector in a lattice, we proceed as follows: We apply *SHORTEST* to the basis. Then, analogous to Proposition 2.11, I claim now that if we choose $j_0 = \operatorname{Min}\{j\colon b_j(j) \geqslant \sqrt{n}\, b_1(1)\}$, then a L_1 shortest vector of $L(b_1, b_2, \ldots, b_{j_0-1})$ is also an L_1 shortest vector of the whole lattice. This is because any vector in $L(b_1, b_2, \ldots, b_n) \setminus L(b_1, b_2, \ldots, b_{j_0-1})$ must have L_2 norm at least $\sqrt{n}\, b_1(1)$ and therefore L_1 norm at least $\sqrt{n}\, b_1(1)$ which is clearly at least the L_1 norm of b_1. Let $m = j_0 - 1$. In any candidate, $\sum_{j=1}^{m} \lambda_j b_j$ for the L_1 shortest vector, we must have $|\lambda_m| b_m(m) \leqslant |b_1|_1 \leqslant \sqrt{n}\, b_1(1)$. Thus there are at most $1 + 2\sqrt{n}\, b_1(1)/b_m(m)$ candidates for λ_m. Arguing in this vein, the total number of candidates is at most $3^m n^{n/2} \prod_{j=1}^{m}(b_1(1)/b_j(j))$ which is at most $(3n)^n$ by Minkowski's theorem. This will give an algorithm for finding the L_1 shortest vector in $O(3^n n^n s)$ arithmetic operations. Similar ideas work for the L_∞ or for any other L_p norms. van Emde Boas (1981) has shown that the shortest vector problem for the L_1 and L_∞ norms is NP-complete.

3. Size of the numbers involved in the algorithm.

We assume that the original input consists of integers. It is easy to see then that all the numbers produced by the algorithm are rational numbers. In what follows, I will derive bounds on the size of the numerators and denominators of all these numbers. The numerator of a rational is of course bounded in absolute value by its magnitude, so really the bounds will be on the magnitude and the denominator of each rational.

First, we will observe that even though the algorithm works on various projected lattices, there is always an implicit "current basis" of the original input n-dimensional lattice. This is true of step 2 (of *SHORTEST*) from the Lenstra, Lenstra and Lovász algorithm. In step 4, we work on the projected lattice $L_2(b_1, b_2, \ldots, b_n)$, but since there is a natural way to "lift" any element of $L_2(b_1, b_2, \ldots, b_n)$ to $L(b_1, b_2, \ldots, b_n)$ (in §1), we can assume that implicitly we have a basis of the whole lattice $L(b_1, b_2, \ldots, b_n)$ provided we can assume that during step 4, while the algorithm is working on $L_2(b_1, b_2, \ldots, b_n)$, it has a basis of $L_2(b_1, b_2, \ldots, b_n)$. By induction, we may indeed assume this and thus there is always an implicit basis of the whole lattice during step 4. Step 5 explicitly computes this implicit basis. By the definition of lifting, note that the basis constructed in step 5 satisfies (2.8)—we will refer to any such basis as "proper". The LLL algorithm always explicitly maintains a basis of the input lattice. Unfortunately, however this basis is not proper at all times. However, when the LLL algorithm terminates, the basis will be proper. It is thus easy to see the following.

PROPOSITION 3.1. *There is an (implicit) "current basis" of the whole lattice at all times during the execution of the algorithm SHORTEST. This basis is proper (satisfies* (2.8)) *except possibly in the middle of the execution of the LLL algorithm.*

In what follows I talk about certain properties of the "current basis" which I will refer to as $b_1, b_2, \ldots, b_n$. With this we can associate the quantities $b_i(j)$ as defined in (1.5).

PROPOSITION 3.2. $\mathrm{Max}_{i=1}^{n} b_i(i)$ *never increases during the execution of SHORTEST.*

PROOF. We consider the algorithm step by step. The proof is by induction on n. For $n = 1$, the proof is trivial. So assume $n \geqslant 2$. For step 2, the *LLL* algorithm never increases the quantity as seen from their proof of their Proposition 1.26. For step 4, the inductive hypothesis suffices. In step 6, $b_1(1)$ strictly decreases, the new $b_2(2)$ is at most the old $|b_1|$ and $b_3(3), \ldots, b_n(n)$ remain the same. For the enumeration and basis selection processes, the proof is a little harder and is dealt with in Proposition 3.3. For step 11 again, we invoke the inductive hypothesis, completing the proof of this proposition. ∎

PROPOSITION 3.3. *Steps* 9 *and* 10 *of the algorithm SHORTEST*$(n; b_1, \ldots, b_n)$ *do not increase* $\max_i b_i(i)$.

PROOF. Suppose $b_1, b_2, \ldots, b_n$ is the basis of the lattice at the beginning of step 8. Let $b_i(j)$, $1 \leqslant j \leqslant i \leqslant n$ be defined as in (1.5). Suppose v_1 is found to be shortest nonzero vector of $L(b_1, b_2, \ldots, b_n)$ by enumeration. Define $u_1 = v_1$, $u_2 = b_1$, $u_3 = b_2, \ldots, u_{n+1} = b_n$. Let $u_i(j)$, $1 \leqslant j \leqslant i \leqslant n+1$ be defined again as in (1.5), i.e., by performing Gram-Schmidt on $u_1, u_2, \ldots, u_{n+1}$. Clearly precisely one of the $u_i(i)$'s is zero. Let this be $u_j(j)$. Let $v_1, v_2, \ldots, v_n$ be the basis returned by SELECT–BASIS in step 10. Again define $v_i(j)$ by (1.7). Then, by the way *SELECT–BASIS* works, it is clear that

$$v_1(l) \leqslant u_l(l) \quad \text{for } l = 2, 3, \ldots, j-1.$$

Also, it is obvious that $u_l(l) \leqslant b_{l-1}(l-1)$ for $l = 2, 3, \ldots, j-1$. Further, $v_l(l) = u_{l+1}(l+1) \leqslant b_l(l)$ for $l = j, j+1, \ldots, n$.

These inequalities together establish the proposition, since obviously $v_1(1) \leqslant b_1(1)$.∎

We now define, for any basis $b_1, \ldots, b_n$ of the n-dimensional lattice $d_i = d(L(b_1, b_2, \ldots, b_i))^2$.

It is not difficult to see that d_i is the determinant of the $i \times i$ matrix with entries (b_j, b_l) for $1 \leqslant j$, $l \leqslant i$. Since our original basis vectors had integer coordinates, this is also true of any other basis. Thus the d_i are all integers. Clearly,

$$d_i = \prod_{j=1}^{i} |b_j(j)|^2. \tag{3.4}$$

The following proposition resembles a similar one in the *LLL* paper.

PROPOSITION 3.5. *All numbers produced by the algorithm are rationals of the form* p/q, p, q *in* $\mathscr{Z}$ *where* q *is one of the* d_i*'s corresponding to the current basis.*

PROOF. Let $b(j, i)$ be the projection of b_j orthogonal to $b_1, \ldots, b_{i-1}$ (for $j \geqslant i \geqslant 2$). (See (1.5)′.) Then $b(j, i) = b_j - \sum_{k=1}^{i-1} \delta_{jk} b_k$ where δ_{jk} are some real numbers. Taking a dot product with b_l $(1 \leqslant l \leqslant i-1)$ and noting that $(b_l, b(j, i)) = 0$, we have

$$(b_j, b_l) = \sum_{k=1}^{i-1} \delta_{jk}(b_k, b_l) \quad \text{for } l = 1, 2, \ldots, i-1.$$

These are $(i-1)$ independent equations in the $(i-1)$ variables δ_{jk} with a coefficient matrix whose determinant is d_{i-1}. Thus $d_{i-1}\delta_{jk}$ are all integers. Hence $d_{i-1} b(j, i)$

is an integral vector. Now, the algorithm *SHORTEST* keeps these vectors $b(j, i)$ for some i as it works on projected lattices. In addition, it has to keep some auxiliary quantities at various times—the μ_{ij}'s during LLL, certain other quantities during the execution of the enumeration and select-basis steps. The proof of the proposition for other quantities is similar and I omit it. ■

LEMMA 3.6. *Except while executing the Lenstra, Lenstra and Lovász algorithm, the current basis contains vectors of length at most $(nB)^{1/2}$ if the original input $b_1, \dots, b_n$ consisted of integral vectors each of length at most $\sqrt{B}$.*

PROOF. By Proposition 3.1, the current basis is always proper in these situations which implies of course that if we did Gram-Schmidt on the current basis, the μ_{kl} of (1.4) are all at most 1/2 in magnitude. Further, the initial $b_i(i)$ is at most $\sqrt{B}$ by hypothesis and so by Proposition 3.2, they are all always bounded by this quantity. Thus using (1.5), and properness, we observe that the length of b_i in the current basis is at most $(|b_i^*|^2 + (1/4)\Sigma_{j=1}^{i-1}|b_j^*|^2)^{1/2}$ which is at most $(nB)^{1/2}$. ■

LEMMA 3.7. *In $SHORTEST(n; b_1, \dots, b_n)$ during every execution of LLL algorithm all numbers produced are bounded in magnitude by $(\sqrt{n}\,B)^{cn^2}$ for some fixed constant c.*

PROOF. Whenever the LLL algorithm is called, all the input vectors to it—say—$a_1, a_2, \dots, a_i$ have rational components with common denominator d where by Proposition 3.5, d is one of the d_i and hence by (3.4) and by Proposition 3.2, is bounded by B^i. Also, by the previous lemma, the lengths of the vectors are all bounded by $(\sqrt{nB})$. Further, it is easily seen that the LLL algorithm behaves identically on input $(da_1, da_2, \dots, da_i)$ as it does on input $a_1, a_2, \dots, a_i$ except that in the second case all vectors are divided by d. $(da_1, \dots, da_i)$ are integral vectors and thus the bounds proved in the LLL paper apply to them. For these input, we have (from their Proposition 1.26) that all numbers produced by their algorithm are bounded in magnitude by $((\max|da_i|)_{i=1}^{n})^{cn}$ which is at most $(B^{n-1}\sqrt{nB})^{cn} \leqslant (\sqrt{n}\,B)^{cn^2}$. ■

PROPOSITION 3.8. *The total number of arithmetic operations performed by SHORTEST while executing calls to the LLL algorithm is $O(n^n \log B)$.*

PROOF. First, let us bound the total number of times LLL is called. By Proposition 2.12, this is at most $(5/2)^n n!$. Using the argument in Lemma 3.7, each call to LLL performs at most as many arithmetic operations as a call to LLL with integer input vectors each of length at most $B^{n-1}\sqrt{nB}$ which is at most $\sqrt{n}\,B^n$. Using their Proposition 1.26 then, we have that each call to LLL performs at most $O(n^4 \log(\sqrt{n}\,B^n)) = O(n^5 \log B + n^4 \log n)$ arithmetic operations. So the total number of operations performed by all calls to LLL is $(5/2)^n n!(n^5 \log B + n^4 \log n)$. Using Stirling's approximation and the fact that 5/2 is strictly less than e, the base of the natural logarithm, we see that this is asymptotically $O(n^n \log B)$. ■

THEOREM 3.9. *On input $b_1, \dots, b_n$ which are independent vectors with integer components of length at most $\sqrt{B}$, all numbers produced by the algorithm $SHORTEST(n; b_1, \dots, b_n)$ can be represented in $O(n^2(\log n + \log B))$ bits.*

PROOF. The proof will be based on Lemma 3.6. It is not by induction on n—I will actually consider the execution of the recursive calls in detail. Let us consider any call to the procedure $SHORTEST(i; u_1, u_2, \dots, u_i)$ (where i is less than n) occurring inside the main call to $SHORTEST(n; \dots)$. For each such call, I will consider the execution of steps 1 through 3 and steps 5 through 10. (In other words, I do not consider the steps invoking the recursive calls since I have in the first place picked any arbitrary call to the procedure inside of the main program.) Step 1 is trivial, step 2 is covered by Lemma 3.7. In step 3 we have to project vectors—$u_2, u_3, \dots, u_i$ perpendicular to a vector u_1 where, of course, these u_j form the basis of some projected lattice. Arguing as in Lemma 3.6, we see that the u_j are all bounded in length by $\sqrt{nB}$. By (3.5), their denominators are bounded by B^{n-1}. Since projecting perpendicular to a vector involves taking certain dot products and simple arithmetic operations, it is easy to see that step 3 never involves more than $O(n(\log n + \log B))$ bit integers. Step 5 is a little harder to analyze partly because I have not specified exactly how the lifting is done. I will do so presently. Suppose $u_1, u_2, \dots, u_i$ is the basis of the lattice in step 3 and suppose $\bar{u}_j'$, $j = 2, 3, \dots, i$ are the projections perpendicular to u_1 in step 3, let $\bar{U}'$ be a matrix with these $i - 1$ vectors as its $i - 1$ rows. Further, let $\bar{u}_2, \bar{u}_3, \dots, \bar{u}_i$ be the basis returned in step 4 after the call to *SHORTEST*$(i - 1; \dots)$ and let $\bar{U}$ be the matrix with these $i - 1$ vectors as its $i - 1$ rows. To lift these vectors, we do the following: We solve a linear system of equations in $(i - 1)^2$ variables to find a $(i - 1) \times (i - 1)$ matrix T so that $\bar{U} = T\bar{U}'$.

Clearly, T so found will have integer entries and the determinant of it will be 1 in absolute value. Now let V equal TU where U is the matrix with $u_2, u_3, \dots, u_i$ as its $i - 1$ rows. Then the rows of V are nearly what we want. We need to ensure that for each row of V, the projection of the row into u_1 is at most $(1/2)|u_1|$ in length. This is done without much difficulty. The solution of the simultaneous equations with a coefficient matrix with entries of $O(n(\log n + \log B))$ bits does not produce any numbers larger than $O(n^2(\log n + \log B))$ bits (Edmonds 1967).

All other steps are easily handled. In fact the only other step in which the size of numbers exceeds $O(n(\log n + \log B))$ bits is in the *SELECT-BASIS* step when we have to solve equations with the coefficient matrix entries with $O(n \log B)$ bits—in this case the number of bits still remains $O(n^2 \log B)$. ∎

4. Finding the closest vector.

In this section, I consider the following *closest vector problem*:

(4.1) Given $b_1, b_2, \dots, b_n$ independent vectors in $\mathcal{Q}^n$ and b_0 in $\mathcal{Q}^n$, find b in $L(b_1, b_2, \dots, b_n)$ such that $|b_0 - b|$ is as small as possible.

This is called the inhomogeneous problem (corresponding to the homogeneous problem called the Shortest Vector Problem earlier). The reason for this terminology is that in the SVP we had to find the closest lattice point to 0, excluding itself whereas here we have to find the closest lattice point to an arbitrary b_0. Note however that here if b_0 itself belongs to the lattice, then the answer to be returned is b_0—in other words, here we do not exclude b_0 as an answer. We can test in polynomial time whether b_0 in fact belongs to the lattice by using the algorithm of von zur Gathen and Sieveking (1976) or Kannan and Bachem (1979) to solve simultaneous diophantine equations, so I assume this is done at the outset and in what follows b_0 does not belong to the lattice.

The CVP algorithm functions as follows: It first uses the procedure *SHORTEST* to make the basis $b_1, b_2, \dots, b_n$ reduced—i.e., the basis then satisfies (2.7) and (2.8). Next, we use an upper bound $\frac{1}{2}(\sum_{j=1}^{n}(b_j(j))^2)^{1/2} = M$ (say) on the distance between any b_0 and its closest lattice point. (This bound will be proved in Proposition 4.2.) Because of this I can argue that there are not too many values of $(\alpha_1, \alpha_2, \dots, \alpha_n)$ integers such that $|\sum_{j=1}^{n}\alpha_j b_j - b_0|$ is within the upper bound. Arguing as in the case of shortest vector problem, (Lemma 2.14), this gives us a bound of $M^n/d(L)$ on the number of possible n-tuples $(\alpha_1, \alpha_2, \dots, \alpha_n)$ to enumerate. Unfortunately, this will not in general be bounded by a function of n alone. So we have to use another idea: If $b_i(i)$ is the largest among all the $b_j(j)$, then I will show that not too many values of $(\alpha_i, \alpha_{i+1}, \dots, \alpha_n)$ are candidates to be tried. The bound on the number of candidates will be $(n + \sqrt{n})^{(n-i+1)}$. For each such candidate, we project to a $(i - 1)$ dimensional problem and solve these recursively. The details are explained after the algorithm. At the outset, we apply *SHORTEST* to get a basis satisfying (2.7) and (2.8). We do not need to repeat this in any recursive call.

```
  Procedure CVP($n$; $b_0, b_1, b_2, \ldots, b_n$).
  Comment. Assumes the basis $b_1, b_2, \ldots, b_n$ is reduced in the sense of (2.7) and (2.8).
  Finds the point of $L(b_1, b_2, \ldots, b_n)$ closest to $b_0$.
  If $n = 1$ then return the easily computed closest lattice point.
  Find $i$ such that $b_i(i) = \max_{j=1}^n b_j(j)$.

                    CANDIDATES $\leftarrow \varnothing$.

For each "possible" $\lambda_i, \lambda_{i+1}, \ldots, \lambda_n$ integers do:
  Comment. This is the enumeration step. I will later explain what the word "possible"
  here means.
  If $i = 1$ then CANDIDATES $\leftarrow$ CANDIDATES $\cup \{\sum_{j=1}^n \lambda_j b_j\}$
    else do
        $v \leftarrow \sum_{j=i}^n \lambda_j b_j$,
        $v' \leftarrow$ CVP($i - 1$; $b_0 - v, b_1, b_2, \ldots, b_{i-1}$).
        CANDIDATES $\leftarrow$ CANDIDATES $\cup \{v + v'\}$.
    end
  end
Return the element of CANDIDATES that is closest to $b_0$.
end CVP
```

The following proposition is used to show that the number of "possible" $\lambda_i, \lambda_{i+1}, \ldots, \lambda_n$ is small.

PROPOSITION 4.2. *Suppose $L = L(b_1, b_2, \ldots, b_n)$ is a lattice in $\mathscr{R}^k$, $k \geqslant n$ with $b_1, b_2, \ldots, b_n$ independent and suppose b_0 is any point in $\mathscr{R}^k$. Let $\bar{b}_0$ be the projection of b_0 into the span of $\{b_1, b_2, \ldots, b_n\}$. Then there exists a point b in L such that*

$$|b - \bar{b}_0| \leqslant \frac{1}{2}\left(\sum_{j=1}^{n}\left(b_j(j)\right)^2\right)^{1/2}.$$

Further if i is such that $b_i(i) = \max b_j(j)$, then clearly, $|b_0 - b| \leqslant (\sqrt{n}/2) b_i(i)$.

PROOF. It is not difficult to see that we can successively choose integers $\alpha_n, \alpha_{n-1}, \ldots, \alpha_1$ (in that order) such that

$$\left|\left(\left(\sum_{l=j}^{n}\alpha_l b_l - \bar{b}_0\right), b(j,j)\right)\right| \leqslant \left(b_j(j)\right)^2/2$$

for all j. This is so because the choice of α_j does not affect the inequalities that were earlier ensured. Since $b(1,1), b(2,2), \ldots, b(n,n)$ form an orthogonal basis for the vector space they span, and $\bar{b}_0$ by definition lies in that space, the proposition follows. ∎

PROPOSITION 4.3. *With the notation set up in the last proposition, there exists an easily determined set $T \subset \mathscr{Z}^{n-i+1}$ with $|T| \leqslant (n + \sqrt{n})^{n-i+1}$ such that if $\sum_{j=1}^n \lambda_j b_j$ is the closest point to b_0 in the lattice then $(\lambda_i, \lambda_{i+1}, \ldots, \lambda_n)$ belongs to T.*

PROOF. Suppose $\sum_{j=1}^n \lambda_j b_j = v$ is a closest point in L to b_0. Then clearly, v must be the closest point in L to $\bar{b}_0$. By the last proposition we must have

$|v - \bar{b}_0| \leqslant (\sqrt{n}/2)b_i(i)$. But $|v - \bar{b}_0| \geqslant |((v - \bar{b}_0), b(n, n))|/b_n(n) = |\lambda_n b_n(n) - ((\bar{b}_0, b(n, n))/b_n(n))| = |\lambda_n - t|b_n(n)$ for some fixed real number t. Thus there are at most $1 + \sqrt{n}\, b_i(i)/b_n(n)$ candidates for λ_n. Now, one can show a similar bound for $\lambda_i, \lambda_{i+1}, \ldots, \lambda_n$ using an argument similar to Proposition 2.13. So suppose $\lambda_{j+1}, \ldots, \lambda_n$ are fixed integers, for some $j \geqslant i + 1$. Then arguing as in that proposition there are at most $1 + \sqrt{n}\, b_i(i)/b_j(j) \leqslant (1 + \sqrt{n})b_i(i)/b_j(j)$ possible values of λ_j such that the length of $v - \bar{b}_0$ in the direction of $b_j(j)$ remains bounded by $(\sqrt{n}/2)b_i(i)$. Note that I have used the fact that $b_i(i) \geqslant b_j(j)$. Thus we have to consider a set T of candidates $\lambda_i, \lambda_{i+1} \ldots \lambda_n$ where

$$|T| \leqslant \prod_{j=i}^{n} (1 + \sqrt{n})b_i(i)/b_j(j). \tag{4.4}$$

Since the basis was reduced in the sense of (2.7) and (2.8), $b_i(i)$ is the length of the shortest vector in the lattice $L_i(b_1, b_2, \ldots, b_n)$. Further, the denominator of the expression in (4.4) is obviously the determinant of the lattice $L_i(b_1, b_2, \ldots, b_n)$. Thus by Minkowski's theorem, $|T| \leqslant (1 + \sqrt{n})^{n-i+1}(n - i + 1)^{(n-i+1)/2} \leqslant (n + \sqrt{n})^{(n-i+1)}$. ■

THEOREM 4.5. *The algorithm $CVP(n; \ldots)$ solves the closest vector problem in $O(n^n s)$ arithmetic operations where s is the length of the input. Further all numbers produced by the algorithm are rationals with numerator and denominator expressible in $O(n^2(s + \log n))$ bits each.*

PROOF. Let $T(n)$ be the number of arithmetic operations performed by $CVP(n; \ldots)$. Then,

$$T(n) \leqslant (n + \sqrt{n})^{(n-i+1)}T(i - 1) + q(n)$$

where $q(n)$ is a polynomial. (Note that this does not depend upon s.) Using the fact that the maximum of $((i - 1)/(n + \sqrt{n}))^{(i-1)}$ for $1 \leqslant i \leqslant n$ is attained at $i = n$ and that the limit of $((n - 1)/n)^{(n-1)}$ is $1/e$, we can establish by induction on n that $T(n)$ is $O(n^n)$. The proof is similar to that of Theorem 2.16 and I omit the details. So, the number of arithmetic operations performed by $CVP(n; \ldots)$ is $O(n^n)$ plus the number performed by *SHORTEST*. Applying Theorem 2.16, we get the current theorem. The bound on the number of bits of all numbers is similar to the proof in §3. ■

One can also find the L_1 closest and the L_∞ closest vectors. See Remark 2.17. The number of candidates will have to be suitably adjusted.

5. Integer programming.

Integer programming again is the following problem:

(5.1) Given $m \times n$ and $m \times 1$ matrices A and b of integers, determine whether there is a x in $\mathbf{Z}^n$ such that $Ax \leqslant b$.

We will do some "preprocessing" on the problem. First, we will modify the problem so that the set $\{x\colon Ax \leqslant b\}$ is bounded, i.e., is a polytope. Second, we ensure that the polytope has positive volume by projecting down to some lower dimensional set if necessary. Then, we will apply an invertible linear transformation to both the polytope and the lattice simultaneously so that the polytope becomes "well-rounded". I will define "well-rounded" more rigorously in (5.2) below. Intuitively, it means that there are two concentric spheres with the smaller one contained in the polytope and the larger one containing the polytope so that the ratio of their radii is bounded above by a function of the dimension alone. Lovász has devised an ingenious polynomial time algorithm to make the polytope "well-rounded". This and the rest of the preprocessing are also part of Lenstra's algorithm. He gives a complete description of this in his paper, so I will say nothing more here except to state precisely the problem at the end of the preprocessing:

(5.2) Given independent vectors $b_1, b_2, \ldots, b_n$ in $\mathbf{Z}^n$, an $m \times n$ integer matrix A and an $m \times 1$ integer matrix b, determine whether there is an x in $L(b_1, b_2, \ldots, b_n)$ such that $Ax \leqslant b$, where the following additional conditions are satisfied by the input: $\exists p \in \mathscr{R}^n$, r and R reals such that

$$R/r \leqslant 2n^{3/2} \tag{5.2a}$$

$$B(p, r) \subset \{x \in \mathscr{R}^n, Ax \leqslant b\} \subset B(p, R) \tag{5.2b}$$

where ($B(q, s)$ is the ball of radius s with q as center).

We proceed as follows: We apply *SHORTEST* to $b_1, \ldots, b_n$. Let now $|b_i(i)| = \max_j |b_j(j)|$. Then there is clearly a point b of $L(b_1, \ldots, b_n)$ (by Proposition 4.2) such that $|b - p| \leqslant \sqrt{n}\, b_i(i)/2$. We consider two cases: (as in Lenstra)

Case 1. $r \geqslant \sqrt{n}\, b_i(i)/2$. Then the answer to question (5.2) is Yes since the inner sphere itself contains a lattice point. So we can return Yes and stop the algorithm. It is easy to see that in this case, we can in fact find the lattice point.

Case 2. $r < \sqrt{n}\,|b_i(i)|/2$ whence $R < n^2|b_i(i)|$. In this case, we argue as in the last section that there are not too many integer values of $\lambda_i, \lambda_{i+1}, \ldots, \lambda_n$ for which there exist integers $\lambda_1, \lambda_2, \ldots, \lambda_{i-1}$ so that $\sum_{j=1}^n \lambda_j b_j$ belongs to $B(p, R)$. We then enumerate all these values of $\lambda_i, \ldots, \lambda_n$ and for each, solve a $(i-1)$ dimensional problem. So the algorithm is going to be a recursive procedure.

Procedure ILP $(n; A, b)$.

Comment. See description of problem (5.1) above. A is an $m \times n$ matrix of integers and b an $m \times 1$ matrix of integers. The procedure returns Yes or No to the question (5.1)

1. Ensure boundedness of the feasible set in $\mathscr{R}^n$. Then ensure positive volume. Use Lovász's algorithm which applies a suitable linear transformation on the space and ensures conditions (5.2a) and (5.2b). Apply the same linear transformation to the lattice. So now we have independent vectors $b_1, b_2, \ldots, b_n$, an $m \times n$ matrix A and an $m \times 1$ matrix b satisfying the conditions of problem (5.2) and we must solve this problem. (Of course n, m may not be the same as in the original input.)

2. $\{b_1, b_2, \ldots, b_n\} \leftarrow SHORTEST\{b_1, b_2, \ldots, b_n\}$.

3. Let $b_i(i) = \max_{j=1}^n b_j(j)$.

4. if $r \geqslant \sqrt{n}\,|b_i(i)|/2$ *then return Yes*

Comment. We may now assume that $r < \sqrt{n}\, b_i(i)/2$ and $R < n^2 b_i(i)$.

5. for each candidate $\{\lambda_i, \lambda_{i+1}, \ldots, \lambda_n\} \in \mathscr{Z}^{n-i+1}$ *do*:

Comment We explain later what the candidates are.

6. $b_0 \leftarrow \sum_{j=i}^n \lambda_j b_j$.

Now, a candidate $\{\lambda_i, \lambda_{i+1}, \ldots, \lambda_n\} \in \mathscr{Z}^{n-i+1}$ is fixed. We want to determine whether there is a point z in $L(b_1, b_2, \ldots, b_{i-1})$ such that $z + b_0$ satisfies $Ax \leqslant b$, equivalently z satisfies $Az \leqslant b - Ab_0$. It is clear how to handle the case $i = 1$, so assume that $i > 1$. Letting $z = \sum_{j=1}^{i-1} \alpha_j b_j$, and B to be the $n \times (i-1)$ matrix with $b_1, b_2, \ldots, b_{i-1}$ as its columns, we want $AB\alpha \leqslant (b - Ab_0)$ where α is required to be a $i - 1$ vector of integers.

```
    7. if $ILP(i-1; AB, b-Ab_0)$ returns yes then return yes.
  end
  Return No
    end ILP
```

We first explain the enumeration process. At the beginning of step 5, we may assume that R is less than $b_i(i)n^2$. Thus any vector a in $L(b_1, \ldots, b_n)$ which could belong to the polytope $\{x\colon Ax \leqslant b\}$ must have the property that $|a-p| \leqslant b_i(i)n^2$. Hence the projection of $a-p$ in the direction of $b(n,n)$ must be less than $n^2 b_i(i)$. Thus, we need to try at most

$$1 + 2n^2\frac{b_i(i)}{b_n(n)} \text{ values of } \lambda_n.$$

Arguing in a similar vein to Proposition 2.13 and Proposition 4.3, the number of candidates for $\lambda_i, \lambda_{i+1}, \ldots, \lambda_n$ is at most

$$2^{n-i+1}n^{2(n-i+1)}\prod_{j=i}^{n}\left(1+\frac{1}{n^2}\right)b_i(i)/b_j(j). \tag{5.3}$$

$b_i(i)$ equals $\Lambda_1(L_i(b_1, b_2, \ldots, b_n))$ since we applied *SHORTEST*. The denominator of (5.3) is, of course, $d(L_i(b_1, b_2, \ldots, b_n))$. Thus using Minkowski's theorem, the whole expression is bounded by $(2n)^{5(n-i+1)/2}$.

The lengthy comments already argue the correctness of the algorithm. In what follows, I prove bounds on the number of arithmetic operations and the bits needed to represent intermediate numbers. The latter bound is not a polynomial, which is undesirable. Recently, Frank and Tardos (1985) have used an ingenious method of approximating linear equations to make the number of bits in this algorithm polynomially bounded. The interested reader is referred to their paper.

THEOREM 5.4. *The algorithm $ILP(n; \ldots)$ on an input of length s, correctly solves the n-variable integer programming problem in $O(n^{9n/2}s)$ arithmetic operations. Each integer produced by the algorithm is $O(n^{2n}s)$ bits in size.*

PROOF. The second part is proved first. There are two steps that dominate the production of large numbers—the "rounding out" step (step 1) and the execution of *SHORTEST*. In what follows, I will restrict attention to these steps, leaving it to the reader to check the other ones. The *ILP* algorithm takes various sections of the polytope and works on each of these sections. Since "taking a section" reduces the dimension by 1, there can be at most n nestings of the "rounding out" and *SHORTEST* steps. I will argue that one pair of executions of these two steps does not increase the size of numbers by more than a factor of $O(n^2)$, thus yielding an overall factor of at most $O(n^{2n})$.

By going through the construction to "round out" a polytope due to Lovász, one finds that this increases the number of bits by at most a factor of n^2. This is because the algorithm obtains the affine transformation that rounds out the polytope $\{x\colon Ax \leqslant b\}$ by mapping $(n+1)$ of its vertices (in n dimensions) to $(0,0,0,\ldots,0), (1,0,\ldots,0), (0,1,\ldots,0), \ldots, (0,0,\ldots,0,1)$. Let S be the $n \times n$ matrix whose ith row equals the $i+1$st of these $n+1$ vertices minus the first. Then the linear transformation corresponding to the affine transformation has as its matrix S^{-1}. The number of bits of S^{-1} is at most $O(n^2)$ times the number of bits needed to define the polytope; we lose at most a factor of $O(n)$ to get S—see for example Gács and Lovász (1979) and a factor of $O(n)$ for the inverse. Thus the transformation does not increase the number of bits by more than a factor of $O(n^2)$. The algorithm

SHORTEST then increases the sizes by at most a factor of $O(n^2)$ by Theorem 3.9. (There is an *additive* $n^2 \log n$ term in the theorem, but this is subsumed by the other terms.) But when the algorithm *SHORTEST* is finished, the sizes are much smaller by Propositions 3.1 and 3.2, in fact, they are at most $O(\log n)$ *plus* what it used to be at the start of *SHORTEST*. Thus I have proved what I promised at the end of the last paragraph.

For the number of arithmetic operations, we use the recursion

$$T(n, s) \leqslant (2n)^{5(n-i+1)/2} T(i-1, n^2 s) + cn^n s.$$

By induction, we can now show that $T(n, s)$ is $O(n^{9n/2} s)$. ∎

If we use the algorithm of Frank and Tardos (1985) and keep numbers polynomially bounded in size, the number of arithmetic operations will be reduced to $O(n^{5n/2} s)$.

I will now briefly discuss the structural result underlying the algorithm. Consider the subspace V spanned by the vectors $b_1, b_2, \ldots, b_{i-1}$ where i is defined in step 3 of the procedure *ILP*. Every lattice point of $L(b_1, b_2, \ldots, b_n)$ belongs to a translate of V of the form $V + z$ where z is in $L(b_i, b_{i+1}, \ldots, b_n)$. What I have shown is that at most $(2n)^{5(n-i+1)/2}$ such translates intersect the polytope if the polytope contains no lattice points. The proof applies to only "well-rounded" polytopes. But it can be easily extended to all polytopes with positive volume: Suppose P is any polytope with positive volume (i.e., is full-dimensional). Then Lovász's algorithm finds a linear transformation τ so that τP is well-rounded. Then clearly, the number of translates of V intersecting τP equals the number of translates of $(\tau^{-1})V$ intersecting P.

If we only want an existential result and are not interested in finding the subspace V, we can do better than $\frac{5}{2}$ in the exponent. The argument is as follows: A result of John (1948) says that for any convex body K in $\mathscr{R}^n$ (the word "body" is used to denote a set of positive volume) there are two similar ellipsoids E_1, E_2 such that $E_1 \subseteq K \subseteq E_2$ and E_2 is obtained by dilating E_1 about its center by a factor of n. Suppose τ is the invertible linear transformation that sends E_1 into a sphere of radius 1 (and hence E_2 into a sphere of radius n). Suppose also that $K \cap \mathscr{Z}^n$ is empty. Let $b_1, b_2, \ldots, b_n$ be a reduced basis of the lattice $L = \tau\mathscr{Z}^n$ (in the sense of (2.6)). Let $b_i(i)$ be the maximum of the $b_j(j)$'s and let V be the space spanned by $b_1, b_2, \ldots, b_{i-1}$. Since $E_1 \cap \mathscr{Z}^n$ is empty, we must have that $\tau E_1 \cap L$ is empty and hence $\sqrt{n}\, b_i(i)/2 > 1$ by Proposition 4.2. This and the fact that τE_2 has radius n can be used to show that the number of $\lambda_i, \lambda_{i+1} \ldots \lambda_n$ for which there exists $\lambda_1, \lambda_2, \ldots, \lambda_{i-1}$ (all integers) so that $\sum_{j=1}^n \lambda_j b_j$ belongs to τE_2 is at most

$$n^{3(n-i+1)/2} \prod_{j=i}^{n} \left(1 + \frac{1}{n^{3/2}}\right) \frac{b_i(i)}{b_j(j)}$$

and by using Minkowski's Theorem 1.9 we get that this quantity is at most $O(n^{2(n-i+1)})$. This bounds the number of translates of V intersecting the τK and hence the number of translates of $\tau^{-1}V$ intersecting K. The case when i was equal to 1 was the "best" case for the algorithm, because then the problem is solved by simple enumeration, no recursive calls were needed in this case. However, for the existential result, it is not intersecting because then $V = \{0\}$ and the relevant translates of V are just the singleton sets consisting of lattice points. Since $K \cap \mathscr{Z}^n = \varnothing$, we already know that *none* of these translates intersects K. But going back to the enumeration argument, since $b_i(i)/b_i(i) = 1$, we could argue exactly the same bound on the number of candidates of $\lambda_{i+1}, \lambda_{i+2} \ldots \lambda_n$. This ensures that the subspace V is always of dimension at least 1. I have proved (albeit sketchily) the following theorem.

THEOREM 5.5. *Suppose K is any bounded convex body in $\mathscr{R}^n$ with $K \cap \mathscr{Z}^n = \varnothing$. Then there is an i, $1 \leqslant i \leqslant n-1$ and an i-dimensional space V which has a basis of integer vectors such that the number of translates of V containing lattice points that intersect K is at most $O(n^{2(n-i+1)})$.*

The result of H. W. Lenstra's mentioned in the introduction can be restated as: If K is any bounded convex body in $\mathscr{R}^n$ with $K \cap \mathscr{Z}^n = \varnothing$, then there is an $n-1$ dimensional subspace V, spanned by integer vectors such that the number of translates of V containing integer points that intersect K is at most c^{n^2}. The bound was improved to c^n by Babai (1985). Based on the results of Lenstra and Schnorr (1984), Hastad (1985) improved it to a polynomial—$O(n^{5/2})$. Grötschel, Lovász and Schrijver (1982) have extended this to unbounded convex bodies. Cook, Collurd and Turan (1985) use this to derive bounds on the number of cutting planes needed to prove the infeasibility of integer programs. By not restricting only to $n-1$ dimensional subspaces, Theorem 5.5 is able to get a 2 in the exponent. It is likely that both Hastad's result and Theorem 5.5 can be improved giving us further improvement in the running time of the integer programming algorithm.

General convex bodies and mixed integer programs. It is easy to see that the methods of this section apply to the problem of determining whether $P \cap \mathscr{Z}^n = \varnothing$ for any convex set defined by a separation oracle (Grötschel, Lovász and Schrijver 1982). This is because the actual description of the convex set (so far assumed to be a polytope) is only used in the Lovász rounding algorithm to approximately optimize a linear function over the set and for this, a separation oracle suffices if we use the ellipsoid algorithm of Khaciyan (1979). Following the argument of Lenstra (1983, §5), we can show that a mixed integer program with n integer variables can be solved in time $O(n^{O(n)}q(s))$ where s is the length of the input and $q(\cdot)$ is a polynomial. It is possible that more efficient algorithms can be developed for special convex sets.

6. Complexity issues.

It was conjectured in Lenstra (1979) that the problem of finding a shortest vector in a lattice $L = L(b_1, \ldots, b_n)$ given $b_1, \ldots, b_n$ is NP-hard.

The conjecture is still open. Van Emde Boas has proved the language L_2-*CLOSEST* defined below (which is the natural language corresponding to the Closest Vector Problem) to be NP-complete. Van Emde Boas's proof is complicated and technical. It is also not published. So I will give here a more natural NP-completeness proof of this language. The reduction will be from 3-dimensional matching (3DM) described below which is known to be NP-complete (Karp 1972).

(6.1) Given a set $T \subseteq \{1, 2, \ldots, n\}^3$, determine whether there is a subset M of T such that for each $i \in \{1, 2, \ldots, n\}$, M has precisely one 3-tuple containing i in the first coordinate, precisely one 3-tuple containing i in the second coordinate and precisely one 3-tuple containing it in the third coordinate. (These 3-tuples do not have to be distinct.)

THEOREM 6.2. *The language L_2-CLOSEST $= \{(b_0, b_1, \ldots, b_n; K) | \exists b \in L(b_1, \ldots, b_n)$ such that $|b - b_0| \leqslant K\}$ is NP-complete.*

PROOF. We can easily reduce the 3DM problem to an integer program as follows: We set up one variable x_t for each 3-tuple t in T. This variable will be forced to take on only the values 0 or 1. The interpretation is that $x_t = 1$ iff t is included in M. Then the 3DM problem is equivalent to the following problem. I leave the proof of this to the reader.

(6.3) Does there exist a feasible solution to the following integer program:

$$\sum_{\{(j,k):\,(i,j,k)\in T\}} x_{(i,j,k)} = 1 \quad \text{for } i = 1,2,\ldots,n, \tag{6.3a}$$

$$\sum_{\{(j,k):\,(j,i,k)\in T\}} x_{(j,i,k)} = 1 \quad \text{for } i = 1,2,\ldots,n, \tag{6.3b}$$

$$\sum_{\{(j,k):\,(j,k,i)\in T\}} x_{(j,k,i)} = 1 \quad \text{for } i = 1,2,\ldots,n, \tag{6.3c}$$

$$x_t \in \{0,1\} \quad \text{for all } t \in T. \tag{6.3d}$$

PROPOSITION 6.4. *In the above integer program,* (6.3d) *can be replaced by the following conditions*:

$$\sum_{t\in T} x_t^2 \leqslant n; \tag{6.5a}$$

$$x_t \in \mathscr{Z} \quad \forall t \in T. \tag{6.5b}$$

PROOF. Suppose x (considered as a vector with $|T|$ components) is a solution to (6.3a), (6.3b), (6.3c) and (6.3d). If x has less than n nonzero components (which are each of course one), then one of the equations (6.3a) will be violated because (6.3a) is comprised of n different equations in disjoint sets of variables; also if x has more than n components with value 1, one of the left-hand sides in (6.3a) will be at least 2. Thus x must have precisely n 1's and so it satisfies (6.5). Conversely, suppose x satisfies (6.3a), (6.3b), (6.3c) and (6.5). To satisfy (6.3a) for example, x must have at least n nonzero components. Each of the nonzero components is of course an integer, so to satisfy the inequality in (6.5), there must be precisely n nonzero components in x and each of these must be ± 1. But if even one of them is -1, there is no way to satisfy (6.3a) say. So they must all be $+1$ and we have proved the proposition.

With the proposition, I have shown the following problem to be NP-complete: (by reducing 3DM to it)

(6.6) Given $m \times n$ and $m \times 1$ matrices of integers A and b respectively and an integer K, determine whether there is a n-vector x satisfying:

$$Ax = b, \qquad x \in \mathscr{Z}^n, \quad |x| \leqslant \sqrt{K}, \tag{6.7}$$

where $|x|$ is the Euclidean length. I will now show that this problem is polynomial time many-one reducible to the Closest Vector Problem (CVP). By using the Hermite Normal form algorithm of Kannan and Bachem (1979), one can find the general integer solution of a system of linear equations in polynomial time. We use this to obtain $b_0, b_1, \ldots, b_\tau$ belonging to $\mathscr{Z}^n$ so that $Ab_0 = b$ and $L(b_1, b_2, \ldots, b_\tau) = \{x\colon\ x \in \mathscr{Z}^n;\ Ax = 0\}$ whence we have $\{x\colon\ x \in \mathscr{Z}^n,\ Ax = b\} = b_0 + L(b_1, b_2, \ldots, b_\tau)$. Thus to solve the problem (6.7), it suffices to find whether there is an element of $L(b_1, b_2, \ldots, b_\tau)$ within distance $\sqrt{K}$ of $-b_0$. This then completes the proof of Theorem 6.2. ∎

The inequality (6.5a) may be replaced by $\sum |x_t| \leqslant n$. This proves that the corresponding language for the L_1 norm is also NP-complete. A similar proof works for the L_∞ norm too.

Now, let us turn our attention to the Shortest Vector Problem (SVP). First, it is convenient to define a language corresponding to the SVP. I will call this language

L_2-SHORTEST:

$$L_2\text{-}SHORTEST = \{(b_1,\dots,b_n;K) \mid \exists b \in L(b_1,\dots,b_n) \quad b \neq 0 \text{ and } |b| \leqslant K\}.$$

THEOREM 6.8. *Given $b_0, b_1, \dots, b_n$ in $\mathscr{R}^n$, $n \geqslant 2$, with polynomially many calls to a subroutine for deciding membership in L_2-SHORTEST and polynomial additional time we can find a vector y in $L(b_1,\dots,b_n)$ such that for all y' in $L(b_1,\dots,b_n)$,*

$$|y - b_0| \leqslant \sqrt{n/2} \cdot |y' - b_0|.$$

REMARK. The theorem asserts that the problem of finding an approximate closest vector to within a factor of $\sqrt{n/2}$ is polynomial-time Turing (Cook) reducible to *L_2-SHORTEST*. The reduction given is essentially a Cook reduction—it invokes more than one call to the subroutine.

We show first that given a subroutine that accepts L_2-shortest, we can actually find a shortest vector in a lattice. Suppose $L = L(b_1,\dots,b_n)$, $b_i \in \mathscr{Z}^n$ independent is the lattice in which we want to find a shortest nonzero vector. Define

$$l = \left(\sqrt{n}\,(d(L))^{1/n}\right)^4. \tag{6.9}$$

Let τ be the linear transformation given by the $n \times n$ diagonal matrix containing entries $l^{3n} + l^{n+1-i}$ in the (i,i)th position for $i = 1,2,\dots,n$. (6.10)

τ multiples the ith coordinate by $(l^{n+1-i} + l^{3n})$. The following lemma is not difficult to prove.

LEMMA 6.11. *Suppose $L = L(b_1,\dots,b_n)$ where $b_i \in \mathscr{Z}^n$ and are independent and define l and τ as in (6.9) and (6.10). Then for $L^* = \tau L$, any shortest nonzero vector of L^* must be of the form*

$$Y = \left((l^{3n} + l^n)y_1, (l^{3n} + l^{(n-1)})y_2, \dots, (l^{3n} + l)y_n\right) \tag{6.12}$$

where $(y_1, y_2, \dots, y_n)$ is a shortest vector L and for any other shortest vector $(y_1', y_2', \dots, y_n')$ of L, we have

$$|y_{i_0}| < |y_{i_0}'| \quad \text{for } i_0 = \min_{i=1}^{n}\{i\colon |y_i| \neq |y_i'|\}. \tag{6.13}$$

(In other words $(|y_1|,\dots,|y_n|)$ is the lexicographically least among the shortest vectors of L.) ∎

From (6.12) and the fact that a shortest nonzero vector $y = (y_1,\dots,y_n)$ of L must satisfy $|y_j| \leqslant l^{1/4}$ for all j, we see easily that if $|Y|^2$ is given, then $(|y_1|, |y_2|, \dots, |y_n|)$ can be determined: Expand the integer $|Y|^2$ to the base l to write

$$|Y|^2 = \sum_{j=0}^{6n} \alpha_j l^j;$$

then $y_1^2 = \alpha_{2n}$, $y_2^2 = \alpha_{2(n-1)}, \dots, y_n^2 = \alpha_2$. Now further, given a subroutine for L_2-*SHORTEST*, we can find $|Y|^2$ using binary search in polynomial time. Thus we can find $(|y_1|, |y_2|, \dots, |y_n|)$ using the subroutine and polynomial additional time. Using this, of course, we could also find $|Y_1|, \dots, |Y_n|$.

We still need the signs of the components of y. Towards this end, first note that L^* has the property that if Y is a shortest vector of L^*, then for any other shortest vector

Y' of L^*, $|Y_i| = |Y_i'|$ (by (6.13)). Let $(|Y_1|, |Y_2|, \ldots, |Y_n|)$ be the magnitudes of the coordinates of a shortest vector Y of L^* already found as described above. Consider the $(n-1)$ dimensional lattice:

$$L' = L^* \cap \{x\colon\ x_1|Y_2| - x_2|Y_1| = 0\}.$$

Clearly, $\Lambda_1(L') = \Lambda_1(L^*)$ iff there is a shortest vector of L^* with the first two coordinates positive. Let $L'' = L^* \cap \{x\colon\ x_1|Y_2| + x_2|Y_1| = 0\}$. Then $\Lambda_1(L'') = \Lambda_1(L^*)$ iff there is a shortest vector of L^* with the first two coordinates of opposite signs. So, we do the following: using our subroutine for L_2–Shortest, we check if $\Lambda_1(L') = \Lambda_1(L^*)$. If so we find (recursively) a shortest vector in L' and hence figure out a shortest vector of L^*, then of L. If not, we find (recursively) a shortest vector of L'' and do like-wise. Note that to solve the problem of finding a shortest vector in n-dimensions, we solve one instance of the corresponding $(n-1)$ dimensional problem plus polynomially many calls to L_2-shortest.

LEMMA 6.14. *With polynomially many calls to a subroutine accepting the language L_2-SHORTEST and polynomial additional time, we can find a shortest nonzero vector in a lattice.*

REMARK. A lemma similar to the one above holds for most known NP-complete languages and several other ones-like linear programming. For example, it is easy to see by using self-reducibility that given an algorithm to test whether a given Boolean formula is satisfiable, we may use it to find a satisfying assignment. This speaks for the versatility of the language SAT (the set of satisfiable Boolean formulas). It is interesting that the language *L_2-SHORTEST* not yet known to be NP-complete has this versatility.

We now study the relationship between the problem of finding a closest vector of a lattice in $\mathscr{R}^n$, to a given point in $\mathscr{R}^n$ (called the "inhomogeneous problem") to that of finding a shortest nonzero vector of a lattice (called the "homogeneous problem"). The device we use to relate these two may be called the process of "homogenization". The technique is used in polyhedral theory. The idea is to relate the inhomogeneous problem for a lattice L in n dimensions to a homogeneous problem for a lattice L' constructed from L in $(n+1)$ dimensions.

Suppose we are given $b_1, b_2, \ldots, b_n, b_0$ in $\mathscr{Z}^n$ and are asked to find a point b of $L = L(b_1, \ldots, b_n)$ which is *approximately* (to be defined later) closest (in Euclidean distance) point of L to b_0. We first check whether b_0 is in L by using a polynomial-time algorithm to solve linear diophantine equations. If so, we may stop. Otherwise we find (using the subroutines for the homogeneous problem) $\Lambda_1(L)$. We then consider the lattice L' in $\mathscr{Q}^{n+1}$ generated by $b_i' = (b_i, 0)$ for $i = 1, 2, \ldots, n$ and $b_{n+1}' = (b_0, (0.51)\Lambda_1(L))$. We find a shortest nonzero vector $v = (v_1, \ldots, v_{n+1})$ of L' (Lemma 6.14). This gives us information about the vector closest to b_0 in L as summarized by the following lemma:

LEMMA 6.15. *Suppose $L = L(b_1, \ldots, b_n)$ is a lattice in $\mathscr{Z}^n$ and b_0 in $\mathbf{Z}^n$ is not in L. Let L' be as defined in the last paragraph and let $v = (v_1, \ldots, v_{n+1})$ be a shortest nonzero vector of L' with $v_{n+1} \leqslant 0$. If $v_{n+1} = 0$, then $|b_0 - b| \geqslant 0.8\Lambda_1(L)$ for all b in L. If $v_{n+1} \neq 0$ then $v_{n+1} = -(0.51)\Lambda_1(L)$ and $(v_1, v_2, \ldots, v_n) + b_0$ is the closest vector in L to b_0.*

PROOF. The shortest vector $v = (v_1, v_2, \ldots, v_{n+1})$ of L' must clearly satisfy $|v_{n+1}| \leqslant \Lambda_1(L)$ because there is a vector of length $\Lambda_1(L)$ in L and hence in L'. Thus $v_{n+1} = 0$ or $\pm 0.51\Lambda_1(L)$. Without loss of generality, we assumed $v_{n+1} \leqslant 0$ and hence is 0 or $-(0.51)\Lambda_1(L)$. Let b be a closest point of L to b_0 and $b = \sum_{j=1}^{n} \alpha_j b_j$. If

$|b - b_0| < 0.8\Lambda_1(L)$, then

$$\left|\sum_{j=1}^{n} \alpha_j b_j' - b_{n+1}'\right| \leqslant \Lambda_1(L)\big((0.8)^2 + (0.51)^2\big)^{1/2} < \Lambda_1(L)$$

and hence the shortest vector v of L' must have $v_{n+1} = -(0.51)\Lambda_1(L)$. This proves the first statement.

To prove the second statement, assume that $v_{n+1} = -0.51\Lambda_1(L)$. Then v equals $(-b_{n+1}' + \sum_{j=1}^{n}\beta_j b_j')$ for some integers β_j and since the last component of v is fixed at absolute value $0.51\Lambda_1(L)$, v will be shortest when $\sum_1^n \beta_j b_j$ is closest to b_0. This proves the lemma. ∎

The lemma leads to the following recursive algorithm for approximating the closest vector. The recursion will be on the dimension of the lattice. The factor of approximation will be $\sqrt{n/2}$ as asserted in Theorem 6.8. For $n = 2$, the algorithm is obvious. So assume we are given a lattice L of dimension $n > 2$ and a point b_0. First, we find the shortest vector v in the lattice L' used in the lemma. If $v_{n+1} \neq 0$, then we have already found the closest vector and we may stop. In the other case, the distance of b_0 to L (henceforth denoted $d(b_0, L)$) is at least $0.8\Lambda_1(L)$. We obtain a basis $b_1, b_2 \ldots b_n$ of L with b_1 as a shortest vector using the subroutine for L_2-*SHORTEST* (cf. Lemma 6.14 and the procedure *SELECT-BASIS* of §2). In the rest of this proof, we let the superscript ^ denote the projection perpendicular to b_1. Recursively we find an element $\hat{b} \in \hat{L}$ so that

$$|\hat{b} - \hat{b}_0| \leqslant \sqrt{\frac{n-1}{2}}\, d(\hat{b}_0, \hat{L}).$$

Now, find b in L so that b projects to $\hat{b}$ and $b - b_0$ has a projection along the direction of b_1 of length at most $|b_1|/2$. Then

$$|b - b_0|^2 \leqslant |b_1|^2/4 + \frac{(n-1)}{2}\big(d(\hat{b}_0, \hat{L})\big)^2. \tag{6.16}$$

We know that $|b_1|^2/4 \leqslant (4 \times 0.64)^{-1} d(b_0, L)^2 \leqslant \frac{1}{2}(d(b_0, L))^2$. Further, we must of course have $d(\hat{b}_0, \hat{L}) \leqslant d(b_0, L)$. Using these two inequalities in (6.16), we get $|b - b_0|^2 \leqslant (n/2)(d(b_0, L))^2$ proving Theorem 6.8.

Remarks. The most important open problem in the area is of course the complexity of the shortest vector problem which has been discussed in the body of the paper. It is conjectured that this problem is NP-hard at least under Cook (Turing) reductions. One approach to proving this is to prove that the approximate version of the closest vector problem is NP-hard. Approximate versions of Integer Programming, Traveling Salesman problem etc. are known to be NP-hard. The difficulty with the CVP is that it is asking for an integer point within a sphere—a very special object. This also raises another interesting question—in proving NP-completeness of the CVP in §4, I reduced the 3-dimensional matching problem to it. Suppose now, we wish to reduce Integer Programming to the CVP. If the IP has n variables and has a total description of length s (the number of bits), then the reduction to the 3DM in general will lead to a problem where the number of variables will depend on *n as well as s* polynomially. The question is: Can we reduce integer programming in polynomial time to CVP so that the number of dimensions of the CVP is a small function of n, the number of variables of the IP? Geometrically, can the question of whether a polytope in $\mathscr{R}^n$ has an integer point be reduced in polynomial time to questions of whether certain spheres in $\mathscr{R}^m$

have integer points for some m close to n. It seems possible that we can achieve a polynomial bound on m in terms of n alone. For the reasons stated earlier in the paragraph, the answer to this question should shed some light on the NP-hardness of the SVP. Another interesting open problem is to devise polynomial time algorithms that come within a subexponential factor of the shortest vector.

One of the essential ideas for all the three algorithms in this paper is the argument bounding the number of candidates for the enumeration. It seems possible that this argument will be of more general use. There is a context other than those in this paper where it has been shown to be useful (Furst and Kannan 1985). I mention this briefly: Suppose we are given a basis $b_1, b_2, \ldots, b_n$ of a lattice with $\mathrm{Min}_{i=1}^{n} b_i(i) = t$. Then for any vector v, we can determine in polynomial time whether there is a point u in the lattice such that $|u - v| < t/2$. To see this, let $u = \Sigma\lambda_i b_i$ satisfy $|u - v| < t/2$. It is not difficult to see that there is at most one candidate for λ_n, since $b_n(n) \geqslant t$. Similarly, if $\lambda_n, \lambda_{n-1}, \ldots, \lambda_{i+1}$ are fixed, there is at most one candidate for λ_i. This helps us determine quickly whether or not there is such a u. This is one of the ideas used by Furst and me to develop a proof system that yields polynomial length proofs of the *infeasibility* of subset sum problems in almost all instances.

Acknowledgement. I wish to thank Alan Frieze, Merrick Furst, Bettina Helfrich, Gary Miller, and Claus Schnorr for helpful discussions and pointing out errors in earlier drafts. I thank the referees for a very careful and thorough reading.

References

Babai, L. (1985). On Lovász's Lattice Reduction and the Nearest Lattice Point Problem. *Lecture Notes in Computer Science* **182**, K. Melhorn (Ed.), 2nd Annual Symp. Theoretical Aspect of Comp. Sci., Springer, Berlin, 13–20.

Blichfeldt, H. F. (1929). The Minimum Value of Quadratic Forms and the Closest Packing of Spheres. *Math. Ann.* **101** 605–608.

Cassels, J. W. S. (1971). *An introduction to the geometry of numbers*. Springer Verlag, Berlin.

Cook, S. A. (1971). The Complexity of Theorem Proving Procedures. Proc. 3rd Ann. ACM Sympos. Theory of Computing, Assoc. Comput. Mach., New York, 151–158.

Cook, W., Cullard, C. and Turan, Gy. (1985). Complexity of Cutting Plane Technical report, Institut für Operations Research, University of Bonn.

Edmonds, J. (1967). Systems of Distinct Representatives and Linear Algebra. *J. Res. Nat. Bur. Standards, Sect. B* **71B** 241–245.

Frank, A. and Tardos, E. (1985). An Application of Simultaneous Approximation in Combinatorial Optimization. Report Institut für Ökonometrie und Operations Research, Univ. Bonn. To appear in *Combinatorica*.

Furst, M. L. and Kannan, R. (1985). Proofs of Infeasibility for Almost all Subset Sum Problems. In preparation.

Gács, P. and Lovász, L. (1979). Khachian's Algorithm for Linear Programming. *Math. Programming Studies*. **14** 61–68.

Grötschel, Lovász, L. and Schrijver, A. (1984). Geometric Methods in Combinatorial Optimization. in *Progress in Combinatorial Optimization*. W. R. Pulleyblank (Ed.), Proc. Silver Jubilee Conference on Comb. Opt., Univ. of Waterloo. Vol. 1. 1982, Academic Press, New York.

Hasted, J. (1985). Private Communication.

Helfrich, B. (1985). Algorithms to Construct Minkowski Reduced Hermite Reduced Lattice Bases. Uni. Frankfurt Technical report, to appear in *Theoretical Computer Sci.*

Hirschberg, D. and Wong, C. K. (1976). A Polynomial-Time Algorithm for the Knapsack Problem with Two Variables. *J. Assoc. Comput. Mach.* **23** 147–154.

John, F. (1948). Extremum Problems with Inequalities as Subsidiary Conditions. Studies and Essays presented to R. Courant.

Kannan, R. (1980). A Polynomial Algorithm for the Two Variable Integer Programming Problem. *J. Assoc. Comput. Mach.* **27** 118–122.

——. (1987). Lattices: Using Mathematics for Computation. (To appear in *Ann. Rev. Computer Sci.* 2. Annual Reviews Inc. Publish.

——. (1983). Improved Algorithms for Integer Programming and Related Lattice Problems. *15th Annual ACM Sympos. Theory of Computing*. 193–206.

Kannan, R. and Bachem, A. (1979). Polynomial Time Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix. *SIAM J. Comput.* **8** 499–507.

Karp, R. M. (1972). Reducibility among Combinatorial Problems. in R. E. Miller and J. W. Thatcher (Eds.), *Complexity of Computer Computations*. Plenum Press, New York, 85–103.

Khaciyan, L. G. (1979). A Polynomial Algorithm in Linear Programming. *Dokl. Akad. Nauk SSSR* 244 1093–1096. (English Translation: *Soviet Math. Dokl.* **20** 191–194.)

Korkine, A. and Zolotareff, G. (1873). Sur les formes quadratiques. *Math. Ann.* **6** 366–389.

Lagarias, J., Lenstra, H. W. and Schnorr, C. P. (1986). Korkine-Zolotarev Bases and Successive Minima of a Lattice and Its Reciprocal Lattice. Manuscript.

Lekkerkerker, C. G. (1969). *Geometry of Numbers*. North Holland, Amsterdam.

Lenstra, A. K. (1981). Lattices and Factorization of Polynomials. Report IW 190/81, Mathematisch Centrum, Amsterdam.

Lenstra, A. K., Lenstra, H. W. and Lovász, L. (1982). Factoring Polynomials with Rational Coefficients. *Math. Ann.* **261** 513–534.

Lenstra, H. W. (1983). Integer Programming with a Fixed Number of Variables. First announcement (1979). *Math. Oper. Res.* **8, 4** 538–548.

Minkowski, H. (1910). *Geometrie der Zahlen*. Teubner Leipzig.

Scarf, H. E. (1981). Production Sets with Individisiblities. I. Generalities. *Econometrica* **49** 1–32; II. The Case of Two Activities. *ibid*. 395–423.

Schnorr, C. P. (1984). A Hierarchy of Polynomial Time Basis Reduction Algorithms. in *Colloq. Math. Soc. Janos Bolyai*. **44**. Theory of Algorithms Pécs (Hungary).

Strang, G. (1980). *Linear Algebra and its Applications*. Academic Press, New York.

van Emde Boas, P. (1981). Another NP-Complete Problem and the Complexity of Computing Short Vectors in a Lattice. Report 81-04, Mathematische Instituut, Univ. Amsterdam.

von zur Gathen, J. and Sieveking, M. (1976). Weitere zum Erfullungsprobleme polynomial equivalente kombinatorische Aufgaben. in *Lecture Notes in Computer Sci.* **43**. Springer, Berlin.

DEPARTMENT OF COMPUTER SCIENCE, CARNEGIE-MELLON UNIVERSITY, PITTSBURGH, PENNSYLVANIA 15213