CSE 206A: Lattice Algorithms and Applications Winter 2010

2: Basic Algorithms

Instructor: *Daniele Micciancio* UCSD CSE

In this lecture we describe efficient algorithms to solve some basic tasks on point lattices. We begin by analyzing the running time required to compute the Gram-Schmidt orthogonalization of a basis.

1. Gram-Schmidt orthogonalization

The number of arithmetic operations performed by the Gram-Schmidt orthogonalization procedure is $O(n^3)$. Can we conclude that it runs in polynomial time? Not yet. In order to prove polynomial time termination, we also need to show that all the numbers involved do not get too big. This will also be useful later on to prove the polynomial time termination of other algorithms. Notice that even if the input matrix $\mathbf{B}$ is integer, the orthogonalized matrix $\mathbf{B}^*$ and the coefficients $\mu_{i,j}$ will in general not be integers. However, if $\mathbf{B}$ is integer (as we will assume for the rest of this section), then the $\mu_{i,j}$ and $\mathbf{B}^*$ are rational.

The Gram-Schmidt algorithm uses rational numbers, so we need to bound both the precision required by these numbers and their magnitude. From the Gram-Schmidts orthogonalization formulas we know that

$$\mathbf{b}_i + \sum_{j<i} \nu_{i,j}\mathbf{b}_j = \mathbf{b}_i^*$$

for some reals $\nu_{i,j}$. Since $\mathbf{b}_i^*$ is orthogonal to $\mathbf{b}_t$ for all $t < i$ we have

$$\langle \mathbf{b}_t, \mathbf{b}_i \rangle + \sum_{j<i} \nu_{i,j}\langle \mathbf{b}_t, \mathbf{b}_j \rangle = 0.$$

In matrix notation, if we let $\mathbf{B}_{i-1} = [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}]$ and $(\nu_i)_j = \nu_{i,j}$ this can be written as:

$$(\mathbf{B}_{i-1}^T \cdot \mathbf{B}_{i-1}) \cdot \nu_i = -\mathbf{B}_{i-1}^T \cdot \mathbf{b}_i$$

which is an integer vector. Solving the above system of linear equations in variables ν_i using Cramer's rule we get

$$\nu_{i,j} \in \frac{\mathbb{Z}}{\det(\mathbf{B}_{i-1}^T \cdot \mathbf{B}_{i-1})} = \frac{\mathbb{Z}}{\det(\mathcal{L}(\mathbf{B}_{i-1}))^2}.$$

We use this property to bound the denominators that can occur in the coefficients $\mu_{i,j}$ and orthogonalized vectors $\mathbf{b}_i^*$. Let $D_i = \det(\mathbf{B}_{i-1})^2$ and notice that

$$D_{i-1} \cdot \mathbf{b}_i^* = D_{i-1} \cdot \mathbf{b}_i + \sum_{j<i} (D_{i-1}\nu_{i,j})\mathbf{b}_j$$

is an integer combination of integer vectors. So, all denominators that occur in vector $\mathbf{b}_i^*$ are factors of D_{i-1}. Let's now compute the coefficients:

$$\begin{aligned}\mu_{i,j} &= \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \\ &= \frac{D_{j-1}\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{D_{j-1}\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \\ &= \frac{\langle \mathbf{b}_i, D_{j-1}\mathbf{b}_j^* \rangle}{D_j} \in \frac{\mathbb{Z}}{D_j}\end{aligned}$$

and the denominators in the $\mu_{i,j}$ must divide D_j.

This proves that all numbers involved in $\mu_{i,j}$ and $\mathbf{b}_i^*$ have denominators at most $\max_k D_k \leq \prod_k \|\mathbf{b}_k\|^2$. Finally, the magnitude of the numbers is also polynomial because $\|\mathbf{b}_i^*\| \leq \|\mathbf{b}_i\|$, so all entries in $\|\mathbf{b}_i^*\|$ are at most $\|\mathbf{b}_i\|$. This proves that the Gram-Schmidt orthogonalization procedure runs in polynomial time.

Theorem 1. *There exists a polynomial time algorithm that on input a matrix* $\mathbf{B}$*, computes the Gram-Schmidt orthogonalization* $\mathbf{B}^*$

This immediately gives a polynomial time algorithm to compute the derminant of a lattice, by first computing $\mathbf{B}^*$, and then taking the product $\det(\mathbf{B}) = \prod_i \|\mathbf{b}_i^*\|$.

Corollary 1. *There exists a polynomial time algorithm to compute the determinant of a lattice.*

2. Hermite Normal Form

The Hermite Normal Form (HNF) of an integer or rational matrix $\mathbf{B}$ is a special basis for the lattice generated by $\mathbf{B}$ that is useful to solve many other computational tasks. We first define HNF for the special case of square matrices.

Definition 1. A square, non-singular matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ is in Hermite normal form (HNF) iff

- $\mathbf{B}$ is lower triangular ($b_{i,j} \neq 0$ implies $i \geq j$)
- For all $i > j$, $0 \leq b_{i,j} < b_{i,i}$, i.e., the off-diagonal elements are reduced modulo the corresponding diagonal element on the same row.

Now, we generalize this definition to (possibly) non-square matrices.

Definition 2. We say that a non-singular matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ is in Hermite normal form (HNF) iff

- There exists $1 \leq i_1 < \ldots < i_h \leq m$ such that $b_{i,j} \neq 0 \Rightarrow (j < h) \wedge (i \geq i_j)$ (strictly decreasing column height).
- For all $k > j$, $0 \leq b_{i_j,k} < b_{i_j,j}$, i.e., all elements at rows i_j are reduced modulo $b_{i_j,j}$.

The index h is the number of non-zero columns in the matrix, and index i_j is the row of the top non-zero element of column j. Because these terms are strictly increasing, each column contains rows that none of the later columns have. Thus, the non-zero columns of a matrix in HNF (as well as the rows with indices i_j) are guaranteed to be linearly independent. We also know (but will not show here) that HNF is unique, i.e., if two matrices $\mathbf{B}$ and $\mathbf{B}'$ are in

HNF and they generate the same lattice ($\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$), then $\mathbf{B} = \mathbf{B}'$ (except at most for the number of zero-columns at the end). We say that $\mathbf{H}$ is the HNF of $\mathbf{B}$ if $\mathcal{L}(\mathbf{H}) = \mathcal{L}(\mathbf{B})$, $\mathbf{H}$ is in HNF, and $\mathbf{H}$ does not contain zero columns.

It is not hard to come up with an algorith that computes the HNF of a matrix performing only a polynomial number of operations. However, a naive solution to this problem may result in superpolynomial running time because the numbers in the intermediate computations can easily get very large. In order to achieve polynomial running time, some care is required.

Notice that the problem of computing the HNF of a rational matrix $\mathbf{B} \in \mathbb{Q}^{m \times n}$ easily reduces to the problem of computing the HNF of an integer matrix as follows:

(1) let D be the least common multiple of all denominators occurring in $\mathbf{B}$,
(2) Compute the HNF $\mathbf{H}$ of integer matrix $D \cdot \mathbf{B} \in \mathbb{Z}^{m \times n}$
(3) Output $D^{-1} \cdot \mathbf{H}$

It is easy to verify that that $\mathbf{H}$ is in HNF if and only if $D \cdot \mathbf{B}$ is. Moreover, if $\mathbf{H}$ and $D \cdot \mathbf{B}$, generate the same lattice, then $D^{-1} \cdot \mathbf{H}$ and $\mathbf{B}$ also generate the same lattice. Therefore, $D^{-1} \cdot \mathbf{H}$ is the HNF of $\mathbf{B}$. This reduction is polynomial time because $\log_2 D$ is at most as big as the bitsize of the input matrix. In the rest of this section we show how to compute the HNF of an integer matrix.

So, we may assume that the input matrix has integer entries. We first give an algorithm to compute the HNF of matrices with full row rank, and then show how to adapt it to arbitrary matrices.

2.1. **Matrices with full row rank.** We give an algorithm that can find the HNF of any matrix $\mathbf{B} \in \mathbb{Z}^{m \times n}$ which has full row rank. We know that in this case the HNF of $\mathbf{B}$ is a square non-singular $m \times m$ matrix $\mathbf{H}$. The idea is to first find the HNF basis $\mathbf{H}$ of a sublattice of $\mathcal{L}(\mathbf{B})$, and then update $\mathbf{H}$ by including the columns of $\mathbf{B}$ one by one. Let's assume for now that we have a polynomial time procedure AddColumn that on input a square non-singular HNF matrix $\mathbf{H} \in \mathbb{Z}^{m \times m}$ and a vector $\mathbf{b}$outputs the HNF of matrix $[\mathbf{H}|\mathbf{b}]$. The HNF of $\mathbf{B}$ can be computed as follows:

(1) Apply the Gram-Schmidt algorithm to the *columns* of $\mathbf{B}$ to find m linearly independent columns. Let $\mathbf{B}'$ be the $m \times m$ matrix given by these columns.
(2) Compute $d = \det(\mathbf{B}')$, using the Gram-Schmidt algorithm or any other polynomial time procedure. Let $\mathbf{H}_0 = d \cdot \mathbf{I}$ be the diagonal matrix with d on the diagonal.
(3) For $i = 1, \ldots, n$ let $\mathbf{H}_i$ the result of applying AddColumn to input $\mathbf{H}_{i-1}$ and $\mathbf{b}_i$.
(4) Output $\mathbf{H}_n$.

The correctness of the algorithm is based on the invariant that for all i,$\mathbf{H}_i$ is the HNF of the lattice $\mathcal{L}([d \cdot \mathbf{I}|\mathbf{b}_1, \ldots, \mathbf{b}_i])$. The invariant is clearly satisfied for $i = 0$. Moreover, it is preserved at every iteration by definition of AddColumn. So, upon termination, the algorithm outputs the HNF of $\mathcal{L}([d \cdot \mathbf{I}|\mathbf{B}])$. Finally, since $d \cdot \mathbb{Z}^m \subseteq \mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$, we have $\mathcal{L}([d \cdot \mathbf{I}|\mathbf{B}]) = \mathcal{L}(\mathbf{B})$ and the algorithm outputs the HNF of $\mathbf{B}$.

Notice that during the entire process, all the entries of $\mathbf{H_i}$ stay bounded by d. In particular, all the numbers are polynomial in the original input $\mathbf{B}$. So, if AddColumn is polynomial time, then the entire HNF algorithm is polynomial time. It remains to give an algorithm for the AddColumn procedure. On input a square non-singular HNF matrix $\mathbf{H} \in \mathbb{Z}^{m \times m}$ and a vector $\mathbf{b} \in \mathbb{Z}^m$, AddColumn proceeds as follows. If $m = 0$, then there is nothing to

do, and we can immediately terminate with output $\mathbf{H}$. Otherwise, let $\mathbf{H}=\begin{bmatrix} a & \mathbf{0}^\top \\ \mathbf{h} & \mathbf{H}' \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} b \\ \mathbf{b}' \end{bmatrix}$ and proceed as follows:

(1) Compute $g = \gcd(a, b)$ and integers x, y such that $xa + yb = g$ using the extended gcd algorithm.

(2) Apply the unimodular transformation $\mathbf{U} = \begin{bmatrix} x & (-b/g) \\ y & (a/g) \end{bmatrix}$ to the first column of $\mathbf{H}$ and $\mathbf{b}$ to obtain

$$\begin{bmatrix} a & b \\ \mathbf{h} & \mathbf{b}' \end{bmatrix} \mathbf{U} = \begin{bmatrix} g & 0 \\ \mathbf{h}' & \mathbf{b}'' \end{bmatrix}.$$

(3) Add an appropriate vector from $\mathcal{L}(\mathbf{H}')$ to $\mathbf{b}''$ so to reduce its entries modulo the diagonal elements of $\mathbf{H}'$.

(4) Recursively invoke AddColumn on input $\mathbf{H}'$ and $\mathbf{b}''$ to obtain a matrix $\mathbf{H}''$

(5) Add an appropriate vector from $\mathcal{L}(\mathbf{H}'')$ to $\mathbf{h}'$ so to reduce its entries modulo the diagonal elements of $\mathbf{H}''$

(6) Output $\begin{bmatrix} g & \mathbf{0}^\top \\ \mathbf{h}' & \mathbf{H}'' \end{bmatrix}$

2.2. **General case.** We would like to reduce the general case to the full-dimensional case. We begin by using a projection operation Π to select $\mathbf{B}'$, a submatrix of $\mathbf{B}$ consisting only of linearly independent rows of $\mathbf{B}$. Then we use the algorithm for the full-dimensional case. Finally, we use the inverse of the projection operation to get our final result.

(1) Run the Gram-Schmidt orthogonalization process on the *rows* $\mathbf{r}_1, \ldots \mathbf{r}_m$ of $\mathbf{B}$, and let $K = \{k_1, \ldots, k_l\}$ $(k_1 < \ldots < k_l)$ be the set of indices such that $\mathbf{r}^*_{k_i} \neq \mathbf{0}$. Define the projection operation $\Pi_K : \mathbb{R}^m \to \mathbb{R}^l$ by $[\Pi_K(\mathbf{x})]_i = x_{k_i}$. Notice that the rows $\mathbf{r}_k$ $(k \in K)$ are linearly independent and any other row can be expressed as a linear combination of the previous rows $\mathbf{r}_j$ $(\{j \in K \colon j < i\})$. Therefore Π_K is one-to-one when restricted to $\mathcal{L}(\mathbf{B})$, and its inverse can be easily computed using the Gram-Schmidt coefficients $\mu_{i,j}$.

(2) Define a new matrix $\mathbf{B}' = \Pi_K(\mathbf{B})$, which is full-rank, and run the algorithm given in the previous section to find the HNF $\mathbf{B}''$ of $\mathbf{B}'$.

(3) Apply the inverse projection function, Π_K^{-1}, to the HNF determined in the previous step $(\mathbf{B}'')$, to give matrix $\mathbf{H}$. It is easy to see that $\mathcal{L}(\mathbf{H}) = \mathcal{L}(\mathbf{B})$ and $\mathbf{H}$ is in HNF. Therefore $\mathbf{H}$ is the HNF of $\mathbf{B}$.

This gives us an algorithm for determining the HNF that runs in time polynomial in n, m and $log(d)$. To complete the proof we need to show that $\log(d)$ is polynomial in the bit-size of the original matrix. Since d is the determinant of a submatrix of B, it is enough to show that for any square matrix $A \in \mathbb{Z}^{n \times n}$, size$(\det(A))$ is polynomial in size(A). Using the Hadamerd inequality $\mathrm{vol}(\mathcal{P}(A)) \leq \Pi \|a_i\|$, we can write

$$d = |\det(A)| \leq \Pi \|\mathbf{a_i}\|$$

If all a_{ij} have bit-size at most α (i.e., $\lg |b_{ij}| \leq \alpha$), we get $\lg d \leq n(\alpha + \lg \sqrt{m})$, proving that the size of the determinant is polynomial in the size of the matrix.

3. The Dual Lattice

Definition 3. The dual of a lattice Λ is the set $\hat{\Lambda}$ of all vectors $\mathbf{x} \in \text{span}(\Lambda)$ such that $\langle \mathbf{x}, \mathbf{y} \rangle$ is an integer for all $\mathbf{y} \in \Lambda$.

The dual lattice $\hat{\Lambda}$ lives in the same vector space as Λ, but usually it is not a sublattice of Λ. E.g., even if $\Lambda \subset \mathbb{Z}^n$ is an integer lattice, the dual will contain noninteger vectors. The definition of dual lattice is very natural if we compare it with the definition of dual for vector spaces. Recall that the dual of an abstract vector space V is defined as the set of linear functions $\phi\colon V \to \mathbb{R}$. When $V \subseteq \mathbb{R}^n$, it is customary to represent function ϕ as a vector $\mathbf{v} \in V$ such that $\phi(\mathbf{x}) = \langle \mathbf{v}, \mathbf{x} \rangle$.The definition of dual lattice is analogous to that for vector spaces, but with $\mathbb{R}$ replaced by $\mathbb{Z}$: the dual of a lattice Λ is the set of linear functions $\phi\colon V \to \mathbb{Z}$, represented as vectors in $\text{span}(\Lambda)$.

Theorem 2. *The dual of a lattice with basis* $\mathbf{B}$ *is a lattice with basis* $\mathbf{D} = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1}$.

Proof. First of all notice that $\text{span}(\mathbf{D}) = \text{span}(\mathbf{B})$ and

$$\mathbf{B}^\top \mathbf{D} = \mathbf{B}^\top \cdot \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1} = (\mathbf{B}^\top \mathbf{B})(\mathbf{B}^\top \mathbf{B})^{-1} = \mathbf{I}.$$

It follows that for any $\mathbf{Dy} \in \mathcal{L}(\mathbf{D})$ and $\mathbf{Bx} \in \mathcal{L}(\mathbf{B})$, we have $(\mathbf{Dy})^\top (\mathbf{Bx}) = \mathbf{y}^\top \mathbf{x} \in \mathbb{Z}$. So, $\mathcal{L}(\mathbf{D})$ is contained in the dual of $\mathcal{L}(\mathbf{B})$. Now consider an arbitrary vector $\mathbf{v}$ in the dual of $\mathcal{L}(\mathbf{B})$, satisfying $\mathbf{B}^\top \mathbf{v} \in \mathbb{Z}^k$ and $\mathbf{v} \in \text{span}(\mathbf{B})$. It follows that $\mathbf{v} = \mathbf{Bw}$ for some $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{D}(\mathbf{B}^\top \mathbf{v}) \in \mathcal{L}(\mathbf{D})$. But $\mathbf{D}\mathbf{B}^\top \mathbf{v} = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1}\mathbf{B}^\top \mathbf{Bw} = \mathbf{Bw} = \mathbf{v}$. So, $\mathbf{v} \in \mathcal{L}(\mathbf{D})$. This proves that $\mathcal{L}(\mathbf{D})$ is the dual of $\mathcal{L}(\mathbf{B})$. □

One can easily check that if $\mathbf{D} = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1}$ then $\mathbf{B} = \mathbf{D}(\mathbf{D}^\top \mathbf{D})^{-1}$. So, if $\mathbf{D}$ is dual to $\mathbf{B}$ then $\mathbf{B}$ is dual to $\mathbf{D}$ and we can talk of $\mathbf{B}, \mathbf{D}$ as a pair of dual bases. We can actually give a more symmetric definition of dual basis pair.

Definition 4. Two bases $\mathbf{B}, \mathbf{D} \in \mathbb{R}^{m \times n}$ are dual if

- $\text{span}(\mathbf{B}) = \text{span}(\mathbf{D})$ and
- $\mathbf{B}^\top \mathbf{D} = \mathbf{I}$.

It immediately follows that the dual of the dual of a lattice is just the original lattice. Let's now compute the determinant of the dual lattice.

Proposition 1. *For every lattice* Λ, $\det\left(\hat{\Lambda}\right) = \frac{1}{\det(\Lambda)}$

Proof. Let $\Lambda = \mathcal{L}(\mathbf{B})$ and $\mathbf{D} = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1}$.We know that $\det(\Lambda) = \sqrt{\det(\mathbf{B}^\top \mathbf{B})}$. Therefore

$$\det\left(\hat{\Lambda}\right) = \sqrt{\det(\mathbf{D}^\top \mathbf{D})} = \sqrt{\det((\mathbf{B}^\top \mathbf{B})^{-\top} \mathbf{B}^\top \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1})} = \sqrt{\det(\mathbf{B}^\top \mathbf{B})^{-1}} = 1/\det(\Lambda).$$

□

3.1. **The orthogonalized dual basis.** We now study how the dual lattice behaves with respect to basic column operations and the Gram-Schmidt orthogonalization procedure. This will be useful later on in the course. Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be a basis and $\mathbf{D}$ be the dual basis. First of all, consider the elementary column operations, and let's see how the dual basis should be updated to maintain the relationship $\mathbf{D}^\top \mathbf{B} = \mathbf{I}$. The following properties are easy to verify

(1) If $\mathbf{B}'$ is obtained from $\mathbf{B}$ by swapping columns i and j, then the dual basis of $\mathbf{B}'$ is also obtained from $\mathbf{D}$ by swapping columns i and j.
(2) If $\mathbf{B}'$ is obtained from $\mathbf{B}$ by multiplying columns i by -1, then the dual basis of $\mathbf{B}'$ is also obtained from $\mathbf{D}$ by multiplying column i by -1
(3) If $\mathbf{B}'$ is obtained from $\mathbf{B}$ by adding $a\mathbf{b}_i$ to $\mathbf{b}_j$, then the dual basis of $\mathbf{B}'$ is obtained from $\mathbf{D}$ by subtracting $a\mathbf{d}_j$ from $\mathbf{d}_i$.

This gives a simple way to update a basis and its dual at the same time.

Now consider the Gram-Schmidt orthogonalization process:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^* \quad \text{where} \quad \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$$

and the projection operations π_i from $\mathbb{R}^m$ onto $\sum_{j \geq i} \mathbb{R}\mathbf{b}_j^*$:

$$\pi_i(\mathbf{x}) = \sum_{j=i}^{n} \frac{\langle \mathbf{x}, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^*.$$

Consider the projected lattice

$$\pi_i(L) = \mathcal{L}\left([\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_n)]\right).$$

What is the dual of $\pi_i(L)$? We now show that the dual of $\pi_i(L)$ is the sublattice generated by $\mathbf{d}_i, \ldots, \mathbf{d}_n$.

Proposition 2. *Let $\mathbf{B}, \mathbf{D}$ a pair of dual basis. For all i, $[\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_n)]$ and $[\mathbf{d}_i, \ldots, \mathbf{d}_n]$ are also dual basis.*

Proof. We only need to prove the statement for $i = 2$. The general statement easily follows by induction on i. So, let $\mathbf{B}' = [\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_n)]$ and $\mathbf{D}' = [\mathbf{d}_2, \ldots, \mathbf{d}_n]$. We want to prove that $\mathbf{B}'$ and $\mathbf{D}'$ span the same vector space, and $(\mathbf{B}')^\top(\mathbf{D}') = \mathbf{I}$. Let's prove this second property first. We want to show that for all $i \neq j > 1$ we have $\langle \pi_2(\mathbf{b}_i), \mathbf{d}_j \rangle = \delta_{i,j}$. Using the definition of π_2 we get

$$\begin{aligned} \langle \pi_2(\mathbf{b}_i), \mathbf{d}_j \rangle &= \langle \mathbf{b}_i - \mu_{i,j}\mathbf{b}_1, \mathbf{d}_j \rangle \\ &= \langle \mathbf{b}_i, \mathbf{d}_j \rangle - \mu_{i,j}\mathbf{b}_1, \mathbf{d}_j \rangle \\ &= \delta_{i,j} - \mu_{i,j}\delta_{1,j} = \delta_{i,j} \end{aligned}$$

because $j > 1$. This proves that $(\mathbf{B}')^\top(\mathbf{D}') = \mathbf{I}$. We now show that $\mathbf{B}'$ and $\mathbf{D}'$ span the same vector space. We know that $\mathbf{B}$ and $\mathbf{D}$ span the same vector space V. The linear span of $\mathbf{B}'$ is by definition the orhogonal complement of $\mathbf{b}_1$ in V. Since the vectors $\mathbf{d}_2, \ldots, \mathbf{d}_n$ are all orthogonal to $\mathbf{b}_1$ (by definition of dual basis) and they are linearly independent, they also span the orthogonal complement of $\mathbf{b}_1$ in V. This complete the proof. □

Now define the orthogonalization of the dual basis in the usual way, but going through the basis vectors in opposite order from $\mathbf{d}_n$ to $\mathbf{d}_1$.

$$\mathbf{d}_i^\dagger = \mathbf{d}_i - \sum_{j>i} \eta_{i,j}\mathbf{d}_j^\dagger \quad \text{where } \eta_{i,j} = \frac{\langle \mathbf{d}_i, \mathbf{d}_j^\dagger \rangle}{\langle \mathbf{d}_j^\dagger, \mathbf{d}_j^\dagger \rangle}$$

and the corresponding projection operations τ_i from $\mathbb{R}^m$ onto $\sum_{j \leq i} \mathbb{R}\mathbf{d}_j^\dagger$:

$$\tau_i(\mathbf{x}) = \sum_{j=1}^{i} \frac{\langle \mathbf{x}, \mathbf{d}_j^\dagger \rangle}{\langle \mathbf{d}_j^\dagger, \mathbf{d}_j^\dagger \rangle} \mathbf{d}_j^\dagger.$$

It follows by duality that for all i, the dual of $[\mathbf{b}_1, \ldots, \mathbf{b}_i]$ is the projected basis $[\tau_i(\mathbf{d}_1), \ldots, \tau_i(\mathbf{d}_i)]$. In general we have the following.

Theorem 3. *Let* $\mathbf{D}$ *be the dual of* $\mathbf{B}$*. Then for all* $i \leq j$ *the dual of* $[\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j)]$ *is* $[\tau_j(\mathbf{d}_i), \ldots, \tau_j(\mathbf{d}_j)]$.

In particular, when $i = j$ we get the following corollary.

Corollary 2. *Let* $\mathbf{D}$ *be the dual of* $\mathbf{B}$ *and let* $\mathbf{B}^*$ *and* $\mathbf{D}^\dagger$ *the corresponding orthogonalized bases. Then for all* i *the two vectors* $\mathbf{b}_i^*$ *and* $\mathbf{d}_i^\dagger$ *satisfy*

- $\frac{\mathbf{b}_i^*}{\|\mathbf{b}_i^*\|} = \frac{\mathbf{d}_i^\dagger}{\|\mathbf{d}_i^\dagger\|}$
- $\|\mathbf{b}_i^*\| \cdot \|\mathbf{d}_i^\dagger\| = 1$.

4. Easy Lattice Problems

We use the HNF algorithm and the dual lattice to efficiently solve various basic problems on lattices.

Basis problem: Given a set of rational vectors $\mathbf{B}$, we want to compute a basis for the lattice $\mathcal{L}(\mathbf{B})$.

This problem is immediately solved (in polynomial time) by computing $HNF(\mathbf{B})$.

Equivalence problem: Given two bases $\mathbf{B}$ and $\mathbf{B}'$, we want to determine if they define the same lattice $\mathcal{L}(\mathbf{B}') = \mathcal{L}(\mathbf{B})$.

This problem can be solved in polynomial time by computing $\mathbf{H} = HNF(\mathbf{B})$ and $\mathbf{H}' = HNF(\mathbf{B}')$, and checking if $\mathbf{H} = \mathbf{H}'$.

Union of lattices: Given two bases $\mathbf{B}$ and $\mathbf{B}'$, we want to determine a basis for the smallest lattice containing both $\mathcal{L}(\mathbf{B})$ and $\mathcal{L}(\mathbf{B}')$.

It is easy to see that this lattice is generated by $[\mathbf{B} \mid \mathbf{B}']$, so a basis for the lattice can be easily computed as $HNF([\mathbf{B} \mid \mathbf{B}'])$.

Containment problem: Given two bases $\mathbf{B}$ and $\mathbf{B}'$, we want to determine if $\mathcal{L}(\mathbf{B}')$ is a sublattice of $\mathcal{L}(\mathbf{B})$, i.e., $\mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$.

This problem is easily reduced to the union and equivalence problems: $\mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$ if and only if $\mathcal{L}([\mathbf{B} \mid \mathbf{B}']) = \mathcal{L}(\mathbf{B})$. So, in order to check the inclusion we only need to compute $HNF([\mathbf{B} \mid \mathbf{B}'])$ and $HNF(\mathbf{B})$ and check for equality of the HNF bases.

Membership problem: Given a lattice $\mathbf{B}$ and a vector $\mathbf{v}$, we want to determine if $\mathbf{v} \in \mathcal{L}\mathbf{B}$.

This immediately reduces to the containment problem, by checking if $\mathcal{L}([\mathbf{v}]) \subseteq \mathcal{L}(\mathbf{B})$. If we need to check membership for many vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, then it is convenient to first compute $\mathbf{H} = HNF(\mathbf{B})$, and then for every i check if $\mathbf{H} = HNF([\mathbf{H} \mid \mathbf{v}_i])$. Notice that $HNF([\mathbf{H} \mid \mathbf{v}_i])$ can be computed much faster than a HNF matrix computation because most of the matrix is already in Hermite Normal Form.

Solving linear system: Let $\mathbf{Ax} = \mathbf{b}$ be a system of linear equations. We want to find a solution $\mathbf{x}$, or tell if no solution exists. (We know this can be done in $O(n^3)$ arithmetic operations. The problem is to show that the numbers stay small.)

We can easily tell if the system admit solution by running Gram-Schmidt on $[\mathbf{A} \mid \mathbf{b}]$, and checking that the last column equals $\mathbf{b}^* = \mathbf{0}$. Also, we can restrict our attention to solutions that only use variables for which $\mathbf{a}^* \neq \mathbf{0}$. So, assume that the columns of $\mathbf{A}$ are linearly independent, and $\mathbf{Ax} = \mathbf{b}$ for some $\mathbf{x}$. We can also eliminate redundant equations by running Gram-Schmidt on the rows of $[\mathbf{A} \mid \mathbf{b}]$ and throwing away equations for which the corresponding orthogonalized vector is $\mathbf{0}$.

So far we have reduced the problem of solving an arbitrary system of linear equations, to solving a system $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a non-singular square matrix. Then, the system can be easily solved by computing the HNF (or, equivalently Gram-Schmidt orthogonalization) of $[\mathbf{A} \mid \mathbf{b}]^T$ to yield $[\mathbf{C} \mid \mathbf{d}]^T$ where $\mathbf{C}$ is a triangular matrix, and then solve the equivalent triangular system $\mathbf{Cx} = \mathbf{d}$ by back substitution. It is easy to see that in this case all the numbers involved are guaranteed not to get too big, and the system can be solved in polynomial time. (Note: this is certainly not the fastest way to solve a system of linear equations. Much faster methods are known.)

As a special case, this shows that the inverse of a non-singular square matrix $\mathbf{A}$ can be computed in polynomial time by solving the equations $\mathbf{Ax}_i = \mathbf{e}_i$. The inverse matrix is given by $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$.

Dual Lattice: Given a lattice basis $\mathbf{B}$, compute the dual basis $\mathbf{D}$, i.e., a basis $\mathbf{D}$ such that $\mathbf{B}^\top \mathbf{D} = \mathbf{I}$ and $\text{span}(\mathbf{B}) = \text{span}(\mathbf{D})$.

This problem is easily solved by computing $\mathbf{D}$ as $\mathbf{D} = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}$. Notice that this computation involves three matrix multiplications, and one matrix inversion.

Intersection of lattices: Given two bases $\mathbf{B}$ and $\mathbf{B}'$, we want to determine a basis for the intersection $\mathcal{L}(\mathbf{B}) \cap \mathcal{L}(\mathbf{B}')$.

It is easy to show that if $\mathcal{L}(\mathbf{D})$ and $\mathcal{L}(\mathbf{D}')$ are the dual lattices of $\mathcal{L}(\mathbf{B})$ and $\mathcal{L}(\mathbf{B}')$, then the dual of $\mathcal{L}(\mathbf{B}) \cap \mathcal{L}(\mathbf{B}')$ is $\mathcal{L}([\mathbf{D} \mid \mathbf{D}'])$. So, a basis for the intersection is obtained by first computing $\mathbf{D}, \mathbf{D}'$ and $\mathbf{H} = HNF([\mathbf{D} \mid \mathbf{D}'])$, and then computing the dual of $\mathbf{H}$.

Cyclic lattice: Let $r(\mathbf{x})$ be the cyclic rotation of vector $\mathbf{x}$, i.e., $r(x_1, \ldots, x_n) = (x_n, x_1, x_2, \ldots, x_{n-1})$. Given a set of vectors $\mathbf{B}$, find the cyclic lattice generated by $\mathbf{B}$, i.e., the smallest cyclic lattice containing $\mathbf{B}$.

This problem is easily solved considering the vectors $r^i(\mathbf{b}_j)$ for all $i = 0, \ldots, n-1$ and $\mathbf{b}_j \in \mathbf{B}$, and computing the lattice generated by these vectors.

A similar problem is that of deciding if a given lattice $\mathcal{L}(\mathbf{B})$ is cyclic, i.e., if $r(\mathcal{L}(\mathbf{B})) \subseteq \mathcal{L}(\mathbf{B})$ (in which case, we also have $r(\mathcal{L}(\mathbf{B})) = \mathcal{L}(\mathbf{B})$. The cyclic lattice decision problem is easily solved computing $HNF(\mathbf{B})$ and $HNF(r(\mathbf{B}))$ and cheking these two matrices for equality.

5. Hard lattice problems

We have seen a number of problems on point lattices that can be solved efficiently, using the Gram-Schmidt and HNF algorithms and the properties of the dual lattice. For many other lattice problems no efficient solution is known, and the problem appear to be computationally hard, unless one settles for approximate solutions. Among these problems, the most famous and important ones are the following.

Problem 1. The *(approximate) Shortest Vector Problem* (SVP): given a lattice basis $\mathbf{B}$ find a nonzero lattice vector of length at most $\gamma \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$. The exact version of the problem

is obtained setting the approximation factor to $\gamma = 1$, and asking for a vector of length $\lambda(\mathcal{L}(B))$.

There is also a problem corresponding to the other successive minima.

Problem 2. The *(approximate) Shortest Independent Vectors Problem* (SIVP): given a rank n lattice basis $\mathbf{B}$ find n linearly independent lattice vectors of length at most $\gamma \cdot \lambda_n(\mathcal{L}(\mathbf{B}))$. The exact version of the problem is obtained setting the approximation factor to $\gamma = 1$.

The following is the inhomogeneous version of SVP.

Problem 3. The *(approximate) Closest Vector Problem* (CVP): given a lattice basis $\mathbf{B}$ and a target vector $\mathbf{t}$, find a lattice vector within distance $\gamma \cdot \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$. The exact version of the problem is obtained setting the approximation factor to $\gamma = 1$.

The approximation factor γ is usually a (monotonically increasing) function of the lattice rank or dimension. Typical approximation factors are $\gamma(n) = n^c$ or $\gamma(n) = 2^n$. There are several variants of all the above problems:

- The probems are usually defined with respect to the Euclidean norm, but can be considered with respect to any other norm function. Changing norm affects both the target value (e.g., $\lambda(\mathcal{L}(\mathbf{B}))$ in the case of SVP) one is trying to achieve, and the measure of quality of the solution (e.g., $\|\mathbf{x}\|$). Some non-euclidean norms that often arise in applications are the ℓ_1 norm $\|\mathbf{v}\|_1 = \sum_i |v_i|$ and the ℓ_∞ norm $\|\mathbf{v}\|_\infty$.
- All problems can be relaxed by asking for just the (approximate) value achieved by the optimal solution. E.g., for SVP, one can simply ask to approximate the value $\lambda(\mathcal{L}(\mathbf{B}))$, without necessarily finding a vector of that length. Similarly for CVP and SIVP. These relaxed variants of lattice problems are usually formulated as decision problems, and denoted GapSVP, GapCVP and GapSIVP. For example, in GapSVP, one is given a lattice basis and a value d, and needs to determine if $\lambda \leq d$ or $\lambda > \gamma d$. (When $d < \lambda \leq \gamma d$ any answer is acceptable, capturing the slackness of the approximation.)

Exercise 1. Consider the following two variants of SVP and prove that they are equivalent, in the sense that each of them can be reduced to the other in polynomial time:

- OptSVP: given a lattice basis $\mathbf{B}$, output a value in the range $[\lambda(\mathcal{L}(\mathbf{B})), \gamma \cdot \lambda(\mathcal{L}(\mathbf{B}))]$
- GapSVP: given a lattice basis $\mathbf{B}$ and a scalar d such that either $\lambda \leq d$ or $\lambda > \gamma d$, determine which case holds true.