Computational Number Theory and Algebra June 11, 2012

Lecture 15

Lecturers: Markus Bläser, Chandan Saha *Scribe: Chandan Saha*

Hermann Minkowski introduced the theory of *geometry of numbers* [Min10], which studies certain mathematical objects known as *lattices*. In the next few lectures, we will show how these objects are elegantly used to design a deterministic polynomial time algorithm for factoring polynomials over rationals. (Recall that there is no known deterministic polynomial time factoring algorithm for polynomials over finite fields.) As another application of lattices, we will discuss the *low exponent attack* on the RSA cryptosystems. The topics of discussion for today's class are:

- Short vectors in lattices.
- The LLL basis reduction algorithm.

1 Short vectors in lattices

A lattice $\mathcal{L} \subseteq \mathbb{R}^m$ is generated by integer linear combinations of vectors in $\mathbb{R}^m$, where $\mathbb{R}$ is the set of real numbers. Formally,

Definition 1 (Lattice) *Let $v_1, \ldots, v_n$ be linearly independent vectors of $\mathbb{R}^m$. A lattice $\mathcal{L}$, generated by $v_1, \ldots, v_n$, is defined as $\mathcal{L} = \{\sum_{i=1}^{n} a_i v_i \mid a_1, \ldots, a_n \in \mathbb{Z}\}$.*

The set $\{v_1, \ldots, v_n\}$ is a *basis* of the lattice $\mathcal{L}$ and n is the dimension of the lattice. Note that dimension of a lattice is well-defined as two different bases must contain the same number of elements. Let V be the $n \times m$ matrix with v_i as the i^{th} row. When $m = n$, volume of $\mathcal{L}$ is given by $\mathsf{vol}(\mathcal{L}) = |\det(V)|$, which is independent of the choice of the basis vectors. This can be argued as follows: Let $\{u_1, \ldots, u_n\}$ be some other basis of $\mathcal{L}$. Denote by U the $n \times m$ matrix with u_i as the i^{th} row. Then there are two $n \times n$ integer matrices A and B such that $V = A \cdot U$ and $U = B \cdot V$, implying that A is an invertible integer matrix. Therefore, $\det(A) = \pm 1$ and if $m = n$ then $|\det(V)| = |\det(U)|$.

We denote the 2-norm of a vector v by $\|v\|$. The following theorem gives an upper bound on the length of a shortest vector of a lattice (when $m = n$) with respect to the 2-norm.

Theorem 2 (Minkowski, 1896) *The length of a shortest vector of a lattice $\mathcal{L} \subseteq \mathbb{R}^n$ of dimension n, is at most $\sqrt{n} \cdot \mathsf{vol}(\mathcal{L})^{\frac{1}{n}}$.*

Proof (Sketch) Let λ be the length of any shortest vector of $\mathcal{L}$. Consider the parallelepiped defined by the vectors $v_1, \ldots, v_n$. The volume of this parallelepiped is $\det(V) = \mathsf{vol}(\mathcal{L})$. Place spheres of radii $\frac{\lambda}{2}$ centered at the corners of this parallelepiped. Each sphere intersects with a part of this parallelepiped. The sum of the volumes of these intersecting spaces for all the spheres must equal the volume of a single sphere in $\mathbb{R}^n$ of radius $\frac{\lambda}{2}$. Since the spheres do not intersect among themselves, this volume must be less than the volume of the parallelepiped. Volume of a sphere in $\mathbb{R}^n$ of radius r is $\frac{\pi^{n/2} r^n}{\Gamma(n/2+1)}$, where Γ is the Gamma function. Therefore, $\frac{\pi^{n/2}(\lambda/2)^n}{\Gamma(n/2+1)} \leq \mathsf{vol}(\mathcal{L})$. Simplifying using Stirling's theorem, $\lambda \leq \sqrt{n} \cdot \mathsf{vol}(\mathcal{L})^{\frac{1}{n}}$. ■

The task of computing a short vector in a given lattice is a key step in many algorithms. However, it is known that finding a vector of length very close to the shortest vector length is a computationally hard problem.

A brief history of short vector computation - In a series of important developments, Ajtai [Ajt98] showed that it is NP-hard to compute a shortest vector in a lattice. (In fact, Ajtai showed that the problem

is as hard on the average as in the worst case.) Following this, Micciancio [Mic00] showed that finding a vector within $\sqrt{2}$ factor of the shortest vector length is also NP-hard. Haviv and Regev [HR07], building on an earlier work by Khot [Kho05], showed that under a reasonable complexity theory assumption ($\mathsf{NP} \not\subseteq \mathsf{RTIME}(2^{\mathsf{poly}(\log n)})$) there is no polynomial time algorithm that can find a vector to within $2^{(\log n)^{1-\epsilon}}$ factor of the shortest vector length, for any arbitrarily small $\epsilon > 0$.

Nevertheless, it is also known that approximating the shortest vector to within a polynomial factor is probably not a NP-hard problem. Goldreich and Goldwasser [GG98] showed that obtaining a $\sqrt{\frac{n}{\log n}}$ factor approximation is in the complexity class AM[2] and hence unlikely to be NP-hard. Furthermore, it was shown by Aharonov and Regev [AR04] that finding a $\sqrt{n}$ factor approximate vector is in the class NP $\cap$ coNP. For a survey on the theory of lattices, refer to the article by Hendrik W. Lenstra Jr. [Jr.08] in [BS08].

Exponential factor approximation of short vectors - Fortunately, for many algorithms finding a short vector even within an exponential factor of the shortest vector length is useful enough. In a major breakthrough, Lenstra, Lenstra and Lovász [LJL82] gave a polynomial time algorithm (LLL algorithm, for short) to find a vector of length no more than $2^{\frac{n-1}{2}}$ times the length of a shortest vector. All the hardness results, mentioned above, followed after this work in an attempt to bridge the gap between the approximation factors for which either an efficient algorithm or a hardness result is known.

In order to understand the LLL algorithm, let us briefly refresh the Gram-Schmidt orthogonalization process.

2 Gram-Schmidt orthogonalization

Let $v_1, \ldots, v_n$ be linearly independent vectors in $\mathbb{R}^m$ and $\mathcal{V}$ be the space spanned by them. Gram-Schmidt orthogonalization is a technique to find orthogonal vectors $v_1^*, \ldots, v_n^*$ such that the space spanned by them is $\mathcal{V}$. We denote the dot product of two vectors u and w by $u \bullet w$ and $\|u\| = \sqrt{u \bullet u}$ is the 2-norm of u. The construction of the orthogonal vectors proceeds as follows,

$$\begin{aligned} v_1^* &= v_1 \quad \text{and} \\ v_i^* &= v_i - \sum_{j<i} \mu_{ij} v_j^* \quad \text{for } 2 \le i \le n \text{ where, } \mu_{ij} = \frac{v_i \bullet v_j^*}{v_j^* \bullet v_j^*} \text{ for } 1 \le j < i. \end{aligned}$$

Define the projection matrix as $M = (\mu_{ij})_{1 \le i,j \le n}$ where $\mu_{ii} = 1$ for all i, $\mu_{ij} = 0$ for $j > i$ and $\mu_{ij} = \frac{v_i \bullet v_j^*}{v_j^* \bullet v_j^*}$ for $j < i$. Let V be the $n \times m$ matrix with $v_1, \ldots, v_n$ as the rows and V^* be the matrix with $v_1^*, \ldots, v_n^*$ as the rows. The following facts are easy to verify and are left as exercise.

Lemma 3 *1. The vectors $v_1^*, \ldots, v_n^*$ are mutually orthogonal and the space spanned by them is $\mathcal{V}$.*

2. v_i^* is the projection of v_i on the orthogonal complement of $\mathcal{V}_{i-1}$, the space spanned by $v_1, \ldots, v_{i-1}$ which is also the space spanned by $v_1^*, \ldots, v_{i-1}^*$. Hence $\|v_i^*\| \le \|v_i\|$ for all i.

3. $V = M \cdot V^*$.

4. $\det(M) = 1$ and so if $m = n$ then $\det(V) = \det(V^*)$.

3 Reduced basis

The main idea behind the LLL algorithm is to compute a *reduced* basis, from a given basis of the lattice, by closely following the Gram-Schmidt orthogonal basis computation. This reduced basis is guaranteed to contain a 'short' vector. The following lemma explains why Gram-Schmidt orthogonalization (GSO) plays a

role in short vector computation. Given a basis $\{v_1, \ldots, v_n\}$ of $\mathcal{L}$, let $\{v_1^*, \ldots, v_n^*\}$ be the orthogonal basis computed by the GSO. Then,

Lemma 4 *For any non-zero $v \in \mathcal{L}$, $\|v\| \geq \min\{\|v_1^*\|, \ldots, \|v_n^*\|\}$.*

Proof Let $v = \sum_{i=1}^{n} a_i v_i$, where each $a_i \in \mathbb{Z}$, and k be the largest index for which $a_k \neq 0$. The GSO computes each v_i^* as $v_i^* = v_i - \sum_{j<i} \mu_{ij} v_j^*$, where $\mu_{ij} \in \mathbb{R}$. Therefore, $v = a_k v_k^* + \sum_{j<k} \mu_j' v_j^*$, for some $\mu_j' \in \mathbb{R}$. Now, $\|v\|^2 = a_k^2 \|v_k^*\|^2 + \sum_{j<k} {\mu_j'}^2 \|v_j^*\|^2 \geq \|v_k^*\|^2$, since a_k is an integer. Hence, $\|v\| \geq \|v_k^*\| \geq \min\{\|v_1^*\|, \ldots, \|v_n^*\|\}$. ■

Thus, if the vectors $v_1^*, \ldots, v_n^*$ belong to the lattice $\mathcal{L}$ then a v_i^* with the minimum norm is a shortest vector of $\mathcal{L}$. But, the basis vectors computed by the GSO need not always belong to the lattice. This gives rise to the notion of a *reduced basis*.

Definition 5 (Reduced basis) *A basis $\{u_1, \ldots, u_n\}$ of $\mathcal{L}$ is called a reduced basis, if the orthogonal basis vectors $\{u_1^*, \ldots, u_n^*\}$, computed by the GSO, satisfy the property $\|u_i^*\|^2 \leq 2\|u_{i+1}^*\|^2$ for all $1 \leq i < n$.*

Suppose we succeed in efficiently computing a reduced basis $\{u_1, \ldots, u_n\}$ from a given basis of the lattice $\mathcal{L}$. From Lemma 4, any vector $u \in \mathcal{L}$ satisfies $\|u\| \geq \min\{\|u_1^*\|, \ldots, \|u_n^*\|\}$, and by the above definition any $\|u_i^*\| \geq 2^{-\frac{n-1}{2}} \|u_1^*\|$. Therefore, $\|u\| \geq 2^{-\frac{n-1}{2}} \|u_1\|$, as $\|u_1^*\| = \|u_1\|$. This means, the length of the vector u_1 is at most $2^{\frac{n-1}{2}}$ times the length of a shortest vector in $\mathcal{L}$.

4 The LLL basis reduction algorithm

The LLL algorithm computes a reduced basis $\{u_1, \ldots, u_n\}$ from a given basis $\{v_1, \ldots, v_n\}$, where $v_i \in \mathbb{Z}^m$ and $\|v_i\| \leq A \in \mathbb{Z}^+$ for every i, in time polynomial in m and $\log A$. This implies the following theorem.

Theorem 6 (Lenstra-Lenstra-Lovász, 1982) *Given n vectors $v_1, \ldots, v_n \in \mathbb{Z}^m$ that are linearly independent over $\mathbb{Q}$ and $\|v_i\| \leq A \in \mathbb{Z}^+$ for every i, a vector $v \in \mathcal{L} = \sum_{i=1}^{n} \mathbb{Z} v_i$ can be computed in time $\mathsf{poly}(m, \log A)$ such that $\|v\|$ is at most $2^{\frac{n-1}{2}}$ times the length of a shortest vector in the lattice $\mathcal{L}$.*

Remark - The condition that $v_1, \ldots, v_n$ are linearly independent is a bit superfluous because one can easily compute a set of linearly independent vectors $v_1', \ldots, v_{n'}' \in \mathbb{Z}^m$ from $v_1, \ldots, v_n$ such that $\mathcal{L} = \sum_{i \leq n} \mathbb{Z} v_i = \sum_{j \leq n'} \mathbb{Z} v_j'$. See exercise 1.

Notations - Before we describe the LLL algorithm, let us fix a few notations and conventions. Let V be an $n \times m$ matrix with rows $\{v_1, \ldots, v_n\}$ and U the matrix with rows $\{u_1, \ldots, u_n\}$. V^* and U^* are $n \times m$ matrices with v_i^* and u_i^* as the i^{th} rows, respectively, where $\{v_1^*, \ldots, v_n^*\}$ is the orthogonal basis computed by GSO from $\{v_1, \ldots, v_n\}$, and similarly $\{u_1^*, \ldots, u_n^*\}$ is the orthogonal basis computed by GSO from $\{u_1, \ldots, u_n\}$. As defined in section 2, the projection matrix M, in a GSO computation from U to U^*, is given by $M = (\mu_{ij})_{1 \leq i,j \leq n}$ where $\mu_{ii} = 1$ for all i, $\mu_{ij} = 0$ if $j > i$, and $u_i = u_i^* + \sum_{j<i} \mu_{ij} u_j^*$ with $\mu_{ij} = \frac{u_i \bullet u_j^*}{\|u_j^*\|^2}$ for $j < i$. Surely, $U = M \cdot U^*$. For brevity, we say '**GSO** of U' to refer to the matrices M and U^*. The notation $\lceil \mu_{ij} \rfloor$ is used to mean the integer closest to μ_{ij}. The space generated by the vectors $u_1, \ldots, u_k$ over rationals is denoted by $\mathcal{U}_k$.

High-level idea - The LLL algorithm is presented in the next page. The high level idea is the following: Given a basis V, compute a basis of $\mathcal{L}$ that is 'close' to the orthogonal basis of V, and which also has the property of a reduced basis. By computing a basis that is close to the orthogonal basis, we hope to find a short vector in the computed basis (by Lemma 4) - the notion of reduced basis actually helps in proving this. (You may find more details on the algorithm and its analysis in chapter 16 of [GG03].)

Correctness and time complexity - Because of the check, $\|u_{i-1}^*\|^2 > 2\|u_i^*\|^2$, in step 5 of the algorithm, it is easy to see that the algorithm outputs a reduced basis whenever (and if) it halts. However, it is not clear as such, whether the algorithm halts in polynomial time. The following lemma proves this.

Algorithm 1 LLL basis reduction

1. Initialize $U = V$ and compute GSO of U. Set $i = 2$.
2. while $i \leq n$ do
3. for $j = i - 1$ to 1 do
4. Set $u_i = u_i - \lceil \mu_{ij} \rfloor u_j$ and update GSO of U.
5. if $i > 1$ and $\|u_{i-1}^*\|^2 > 2\|u_i^*\|^2$ then
6. Swap u_i and u_{i-1} and update GSO of U. Set $i = i - 1$.
7. else, set $i = i + 1$.
8. Return U.

Lemma 7 *The* LLL *algorithm halts and finds a reduced basis in polynomial time.*

Proof We need to show that the outer `while`-loop of the algorithm executes only polynomially many times. Moreover, we also need to show that the size of the numerator and denominator of any rational number involved in the computation is polynomially bounded.

The matrix U and its GSO are updated in step 4 and step 6. Right before an update, let U be the matrix with GSO data M and U^*. After an update, suppose U, M and U^* get altered to $\tilde{U}$, $\tilde{M}$ and $\tilde{U}^*$ respectively, with the corresponding entries as $\tilde{u}_k$, $\tilde{\mu}_{k\ell}$ and $\tilde{u}_k^*$ for $1 \leq k, \ell \leq n$. $\tilde{\mathcal{U}}_k$ be the space generated by $\tilde{u}_1, \ldots, \tilde{u}_k$.

First, let us focus on step 4. Indices i and j are fixed as in step 4. Let N be the $n \times n$ matrix with ones on the diagonal and $-\lceil \mu_{ij} \rfloor$ as the $(i,j)^{th}$ entry. The remaining entries of N are zeroes. Then, $\tilde{U} = N \cdot U$. Notice that, $\tilde{\mathcal{U}}_k = \mathcal{U}_k$ for all k. Since $\tilde{u}_{k+1}^*$ is the projection of $\tilde{u}_{k+1}$ on the orthogonal complement of $\tilde{\mathcal{U}}_k$, we can infer that $\tilde{U}^* = U^*$. It is also not hard to verify that $\tilde{M} = N \cdot M$. Since $\tilde{\mu}_{ij} = \mu_{ij} - \lceil \mu_{ij} \rfloor$, $|\tilde{\mu}_{ij}| \leq \frac{1}{2}$. Also, $\tilde{\mu}_{i\ell} = \mu_{i\ell}$ for any $\ell > j$. Therefore, by induction on j, $|\tilde{\mu}_{i\ell}| \leq \frac{1}{2}$ for all $j \leq \ell < i$ and $|\tilde{\mu}_{k\ell}| \leq \frac{1}{2}$ for all $1 \leq \ell < k < i$. To summarize, after an update in step 4, the $(k.\ell)^{th}$ entry of the projection matrix has absolute value at most $\frac{1}{2}$ for all $1 \leq \ell < k \leq i$, and the orthogonal basis remains unaltered.

Let us see what happens after an update in step 6. Index i is as in step 6. This time $\tilde{U}$ is simply a permutation matrix times U. The permutation matrix has the effect of swapping the $(i-1)^{th}$ and the i^{th} rows of U. Since $\tilde{\mathcal{U}}_k = \mathcal{U}_k$ for all $k \neq i-1$, hence $\tilde{u}_k^* = u_k^*$ for all $k \notin \{i-1, i\}$. Now notice that $\tilde{u}_{i-1}^* = u_i^* + \mu_{ii-1} u_{i-1}^*$, implying $\|\tilde{u}_{i-1}^*\|^2 \leq \|u_i^*\|^2 + |\mu_{ii-1}|^2 \cdot \|u_{i-1}^*\|^2$. In step 6, $\|u_i^*\|^2 < \frac{1}{2}\|u_{i-1}^*\|^2$ and $\mu_{ii-1} \leq \frac{1}{2}$, as argued in the previous paragraph. Therefore, $\|\tilde{u}_{i-1}^*\|^2 < \frac{3}{4}\|u_{i-1}^*\|^2$. Also, since $\tilde{u}_i^*$ is the projection of u_{i-1} on the orthogonal complement of $\tilde{\mathcal{U}}_{i-1} \supseteq \mathcal{U}_{i-2}$, it is also the projection of u_{i-1}^* on the orthogonal complement of $\tilde{\mathcal{U}}_{i-1}$, implying that $\|\tilde{u}_i^*\| \leq \|u_{i-1}^*\|$. To summarize, after an update in step 6, $\tilde{u}_k^* = u_k^*$ for all $k \notin \{i-1, i\}$, $\|\tilde{u}_{i-1}^*\|^2 < \frac{3}{4}\|u_{i-1}^*\|^2$ and $\|\tilde{u}_i^*\| \leq \|u_{i-1}^*\|$, i.e. $\max_k\{\|\tilde{u}_k^*\|\} \leq \max_k\{\|u_k^*\|\}$.

Let U_k be a $k \times m$ matrix with rows $u_1, \ldots, u_k$. Define $d_k = \det(U_k \cdot U_k^T) \in \mathbb{Z}^+$. Surely, $U_k = M_k \cdot U_k^*$, where M_k is the Gram-Schmidt projection matrix and U_k^* is the orthogonal basis matrix with $u_1^*, \ldots, u_k^*$ as the rows. Since $\det(M_k) = 1$ and $u_1^*, \ldots, u_k^*$ are mutually orthogonal, $d_k = \prod_{\ell=1}^k \|u_\ell^*\|^2$. Thus, in step 4, d_k remains unchanged for every k. In step 6, d_k remains unchanged for every $k \neq i-1$ and d_{i-1} reduces by a factor of at least $\frac{3}{4}$. The reason d_k remains the same for $k \neq i-1$ is because in step 6, U_i only changes to $P \cdot U_i$ for some permutation matrix P. Define $D = \prod_{k=1}^n d_k$. Therefore, D remains unchanged in step 4 but decreases by a factor of at least $\frac{3}{4}$ each time step 6 is executed. At the start of the algorithm the value of D is at most $\prod_{k=1}^n A^{2k} \leq A^{n^2}$ (as $\|v_i^*\| \leq \|v_i\| \leq A$, for all i). Hence, step 6 can be executed at most $O(n^2 \log A)$ times. This proves that the algorithm halts after polynomially many executions of the `while`-loop as the index i can decrease for at most $O(n^2 \log A)$ times. (to be continued in the next class...) ■

Exercises:

1. Suppose $v_1, \ldots, v_n$ are vectors in $\mathbb{Z}^m$ such that $\|v_i\| \leq A \in \mathbb{Z}^+$ for all $1 \leq i \leq n$. Design an algorithm

that finds linearly independent vectors $v'_1, \ldots, v'_{n'} \in \mathbb{Z}^m$ in time polynomial in m and $\log A$, such that $\mathcal{L} = \sum_{i=1}^{n} \mathbb{Z}v_i = \sum_{i=1}^{n'} \mathbb{Z}v'_i$.
2. Fill in the missing details in the proof of Lemma 7.

References

[Ajt98] Miklós Ajtai. The Shortest Vector Problem in L_2 is NP-hard for Randomized Reductions (Extended Abstract). In *STOC*, pages 10–19, 1998.

[AR04] Dorit Aharonov and Oded Regev. Lattice Problems in NP ∩ coNP. In *FOCS*, pages 362–371, 2004.

[BS08] Joseph P. Buhler and Peter Stevenhagen, editors. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography*, volume 44 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press, 2008.

[GG98] Oded Goldreich and Shafi Goldwasser. On the Limits of Non-Approximability of Lattice Problems. In *STOC*, pages 1–9, 1998.

[GG03] Joachim Von Zur Gathen and Jurgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2003.

[HR07] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *STOC*, pages 469–477, 2007.

[Jr.08] Hendrik W. Lenstra Jr. Lattices. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography, MSRI Publications*, 44:127–181, 2008.

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005.

[LJL82] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[Mic00] Daniele Micciancio. The Shortest Vector Problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2000.

[Min10] Hermann Minkowski. *Geometrie der Zahlen*. B. G. Teubner, Leipzig, 1910.