LLL ALGORITHM

NEIL KRISHNAN, SAMBHU GANESAN, ISAAC SUN

1. Introduction

Let $B = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ be a basis of vectors over $\mathbb{R}^n$. Define a lattice, $\mathcal{L}$ as follows.

Definition 1.1. A lattice, $\mathcal{L}$, is composed of integer combinations of basis vectors $\mathbf{b}_i$ in basis B.

We define the length of a vector, v, to be its norm, $||v||$ in $\mathbb{R}^n$. The main problem this paper focuses on is the shortest vector problem (SVP), a problem widely discussed in the fields of mathematics, especially cryptography, which asks what is the shortest vector in a lattice $\mathcal{L}$ with basis B. We denote $\lambda(\mathcal{L})$ to be the length shortest vector of $\mathcal{L}$.

2. Minkowski's Theorem

The first major attempt to solve this problem was by Minkowski who proved an upper bound of the shortest vector problem using the determinant of a lattice, $\mathcal{L}$ defined as follows.

Definition 2.1. The determinant of a lattice, $\mathcal{L}$, with basis $B = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ is

$$\det(\mathcal{L}) = \det\left(\begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ \vdots & \vdots & & \vdots \end{bmatrix}\right).$$

Let $R \subseteq \mathbb{R}^n$ be a region $\mathbb{R}^n$. Define the properties of convexity, centrally symmetric, and volume as follows.

Definition 2.2. A region, R is convex if for any points, $\mathbf{x}, \mathbf{y} \in R$, the line segment defined by $\ell = \{\alpha\mathbf{x} + (1-\alpha)\mathbf{y} : 0 \leq \alpha \leq 1\}$ is contained in R.

Definition 2.3. A region, R is centrally symmetric if for a point $\mathbf{x} \in R$, $-\mathbf{x} \in R$ as well.

Definition 2.4. For a region R, define $\nu_R(\mathbf{x})$ as 1 if $\mathbf{x} \in R$ and 0 if $\mathbf{x} \notin R$. The volume of R, denoted as $\text{vol}(R)$, is

$$\text{vol}(R) = \int_{\mathbf{x}\in\mathbb{R}^n} \nu_R(\mathbf{x}).$$

With this in mind, let us introduce the theorem of Minkowski.

Theorem 2.5 (Minkowski). *Given some lattice, $\mathcal{L}$, if a region R is convex and symmetric about the origin and has volume of at least $2^n \det(\mathcal{L})$ where n is the number of basis vectors defining $\mathcal{L}$, R contains a nonzero lattice point.*

Let us start by defining the Fundamental Region of a lattice.

Date: December 11, 2023.

Definition 2.6. The fundamental region is a set, $\mathcal{F} \in \mathbb{R}^n$ such that the translations of $\mathcal{F}$ defined by $\mathbf{x} + \mathcal{F} = \{\mathbf{x} + \mathbf{y} : \mathbf{y} \in \mathcal{F}\}$ for all $\mathbf{x} \in \mathcal{L}$ satisfy

(1) Intersection condition - $\bigcap_{\mathbf{x} \in \mathcal{L}} \mathbf{x} + \mathcal{F} = \emptyset$,
(2) Union condition - $\bigcup_{\mathbf{x} \in \mathcal{L}} \mathbf{x} + \mathcal{F} = \mathbb{R}^n$.

We will now prove the theorem.

Proof. Define $R' = R/2$ or rather $R' = \{\mathbf{x}/2 : \mathbf{x} \in R\}$. As a result $\text{vol}(R') \geq \det(\mathcal{L})$. Between R and R', the ratio of the side length of the hyper-cubes is $2:1$, so $\text{vol}(R) = 2^n \text{vol}(R')$. Thus, $\text{vol}(R') = \det(\mathcal{L})$. Note that $\text{vol}(\mathcal{F}) = \det(\mathcal{L})$. Denote $R'_{\mathbf{v}}$ to be $R' \cap (\mathbf{v} + \mathcal{F})$ where $\mathbf{v} \in \mathcal{L}$. Thus, $R'_{\mathbf{v}} - \mathbf{v} \subseteq \mathcal{F}$. Thus, we have

$$\sum_{\mathbf{v} \in \mathcal{L}} \text{vol}(R'_{\mathbf{v}} - \mathbf{v}) = \sum_{\mathbf{v} \in \mathcal{L}} \text{vol}(R'_{\mathbf{v}}) = \text{vol}(R') > \det(\mathcal{L}) = \text{vol}(\mathcal{F}),$$

so there must be a point, $\mathbf{z} \in \mathcal{F}$ which lies on $R'_{\mathbf{v}} - \mathbf{v}$ and $R'_{\mathbf{u}} - \mathbf{u}$ where $\mathbf{v} \neq \mathbf{u}$. Let $\mathbf{x}, \mathbf{y} \in R'$ be defined as $\mathbf{x} = \mathbf{z} + \mathbf{v}$ and let $\mathbf{y} = \mathbf{z} + \mathbf{u}$. By definition then, $2\mathbf{x} \in R$ and $2\mathbf{y} \in R$ as well. Because, R is centrally symmetric, we have that $-2\mathbf{y} \in R$, and because R is convex, we also know that the midpoint of $2\mathbf{x}$ and $-2\mathbf{y}$, $\mathbf{x} - \mathbf{y}$ which is nonzero, is contained within R. ■

As a corollary, we can see that this bounds the length of the shortest vector of $\mathcal{L}$.

Corollary 2.7. *Given some lattice, $\mathcal{L}$, we have that $\lambda(\mathcal{L}) \leq \sqrt{n} \det(\mathcal{L})^{1/n}$.*

Proof. Define R to be the n-dimensional sphere of radius $\sqrt{n} \det(\mathcal{L})^{-1/n}$. Because of this, R must contain the n-dimensional hypercube, $[-\det(\mathcal{L})^{-1/n}, \det(\mathcal{L})^{-1/n}]^n$, so we have that $\text{vol}(R) > 2^n \det(\mathcal{L})$. Thus, by Minkowski's theorem, R contains a nonzero lattice point, and because the radius of R is $\sqrt{n} \det(\mathcal{L})^{-1/n}$, there is a vector in $\mathcal{L}$ with length less than $\sqrt{n} \det(\mathcal{L})^{-1/n}$. Therefore, $\lambda(\mathcal{L}) \leq \sqrt{n} \det(\mathcal{L})^{1/n}$. ■

Though this upper bound is useful, we still have not solved SVP, and to this day, we do not have an efficient algorithm that can solve SVP. The LLL algorithm, rather, just serves as an approximation of SVP which runs in polynomial time in terms of the rank of the matrix constructed from the basis vectors:

$$\begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ \vdots & \vdots & & \vdots \end{bmatrix}.$$

3. Basis Reduction

The process of basis reduction is given some basis B, we try to construct a different basis B' over the same lattice where the basis vectors of B' generally have a smaller norm and are more orthogonal. The definition of a reduced basis varies between algorithms.

Example. In Figure 1, the left image depicts a lattice spanned by a basis, and the right image depicts the same lattice spanned by a reduced basis.

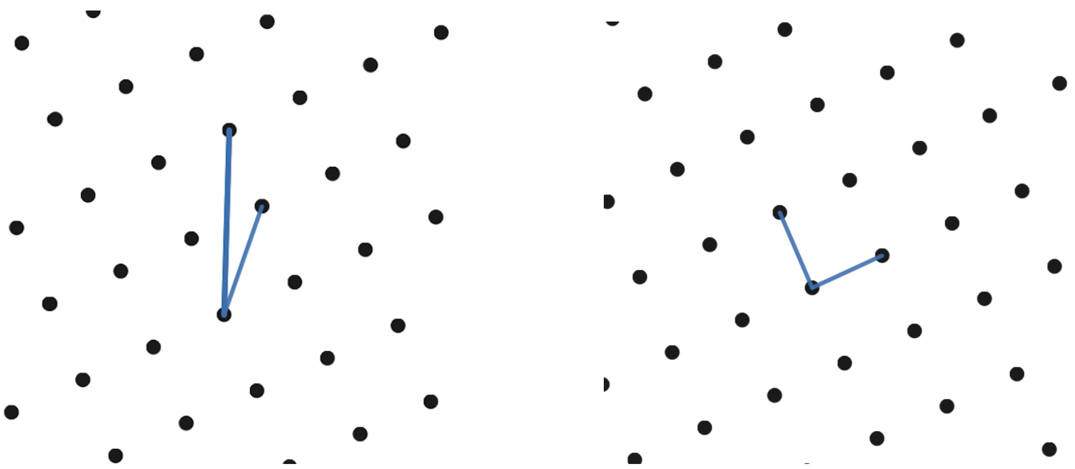

Figure 1. Two basis which span the same lattice. The right image depicts a reduced basis.

The reason why basis reduction is useful for the SVP is because once we cannot reduce the basis anymore, one of the basis vectors must be the shortest vector, solving the problem entirely.

This type of strategy is useful in solving SVP for the 2-dimensional case which was done by Gauss in the 1800s. Let us start with a definition of a reduced basis in two dimensions.

Definition 3.1. A basis, $B = (\mathbf{b}_1, \mathbf{b}_2)$, is reduced if the following hold

$$||\mathbf{b}_1|| \leq ||\mathbf{b}_2||.$$
$$|u| = \left|\frac{\mathbf{b}_1 \cdot \mathbf{b}_2}{||\mathbf{b}_1||^2}\right| \leq \frac{1}{2}.$$

We need to show that in a reduced basis, $\mathbf{b}_1$ is the shortest vector of the lattice.

Theorem 3.2. *If a basis $B = (\boldsymbol{b}_1, \boldsymbol{b}_2)$ is reduced, then $\boldsymbol{b}_1$ is the shortest vector.*

Proof. Let $\mathbf{v} = r\mathbf{b}_1 + t\mathbf{b}_2$ be a vector of the lattice constructed from B. Then,

$$\begin{aligned}
||\mathbf{v}||^2 &= ||r\mathbf{b}_1 + t\mathbf{b}_2||^2, \\
&= ||r\mathbf{b}_1 + t(\mathbf{b}_2^* + u\mathbf{b}_1)||^2, \\
&= ||(r + tu)\mathbf{b}_1 + t\mathbf{b}_2^*||^2, \\
&= ||(r + tu)\mathbf{b}_1||^2 + ||t\mathbf{b}_2^*||^2,
\end{aligned}$$

where $\mathbf{b}_2^*$ refers to the part of $\mathbf{b}_2$ orthogonal to $\mathbf{b}_1$. Note that

$$||\mathbf{b}_2^*||^2 = ||\mathbf{b}_2||^2 - ||u\mathbf{b}_1||^2.$$

because $||\mathbf{b}_2||$, $||\mathbf{b}_2^*||$ and $||u\mathbf{b}_1||$ are the lengths of the hypotenuse and legs, respectively of a right triangle constructed from $\mathbf{b}_2, \mathbf{b}_2^*$, and $u\mathbf{b}_1$. Therefore,

$$||\mathbf{b}_2^*||^2 = ||\mathbf{b}_2||^2 - u^2||\mathbf{b}_1||^2 \geq ||\mathbf{b}_1||^2 - \frac{1}{4}||\mathbf{b}_1||^2 = \frac{3}{4}||\mathbf{b}_1||^2.$$

As a result, we have that

$$\begin{aligned}||\mathbf{v}||^2 &= ||(r+tu)\mathbf{b}_1||^2 + ||t\mathbf{b}_2^*||^2,\\ &\geq (r+tu)^2||\mathbf{b}_1||^2 + t^2\frac{3}{4}||\mathbf{b}_1||^2,\\ &\geq ||\mathbf{b}_1||^2\end{aligned}$$

Thus, $\mathbf{b}_1$ is the shortest vector of the lattice in a reduced basis. ■

Using this, Gauss developed the following algorithm

(1) Start with basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. If $||\mathbf{b}_1|| > ||\mathbf{b}_2||$, then swap them
(2) If $|u| = \frac{\mathbf{b}_1 \cdot \mathbf{b}_2}{||\mathbf{b}_1||^2} \leq \frac{1}{2}$, terminate.
(3) Otherwise, let $m = \lfloor u \rfloor$ if u is positive and $\lceil u \rceil$ if u is negative. Make $\mathbf{b}_2 = \mathbf{b}_1 - m\mathbf{b}_2$. Repeat the first step.

Example. Let us try Gauss's method with the basis $B = (\mathbf{b}_1 = \langle 3,4\rangle, \mathbf{b}_2 = \langle 1,6\rangle)$. We have:

$$\begin{aligned}
&(1) \quad ||\langle 3,4\rangle|| = 5 \leq ||\langle 1,6\rangle|| = \sqrt{37}.\\
&(2) \quad |u| = \left|\frac{\langle 3,4\rangle \cdot \langle 1,6\rangle}{||\langle 3,4\rangle||^2}\right| = \frac{27}{25} > \frac{1}{2}.\\
&(3) \quad m = \lfloor u \rfloor = 1 \Rightarrow \mathbf{b}_2 = \langle 1,6\rangle - \langle 3,4\rangle = \langle -2,2\rangle.\\
&(1) \quad ||\langle 3,4\rangle|| = 5 > ||\langle -2,2\rangle|| = \sqrt{8} \Rightarrow \mathbf{b}_1 = \langle -2,2\rangle, \mathbf{b}_2 = \langle 3,4\rangle.\\
&(2) \quad |u| = \left|\frac{\langle -2,2\rangle \cdot \langle 3,4\rangle}{||\langle -2,2\rangle||^2}\right| = \frac{1}{4} \leq \frac{1}{2}.
\end{aligned}$$

Thus, the shortest vector in the lattice is $\langle -2,2\rangle$.

4. Gram-Schmidt Orthogonalization

Recall that an orthogonal basis is a basis where the pairwise dot products of the basis vectors is 0. The idea behind using the Gram-Schmidt Orthogonalization method is to use its orthogonal basis to construct a nearly orthogonal basis over $\mathcal{L}$. Then, as we will show, the shortest vector is close to the smallest basis vector of the orthogonal basis. Let us start by proving the Gram-Schmidt Orthogonalization Method.

Theorem 4.1 (Gram-Schmidt Orthogonalization Method)**.** *Let B be a basis with basis vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$. Define*

$$\begin{aligned}\boldsymbol{b}_1^* &= \boldsymbol{b}_1,\\ \boldsymbol{b}_2^* &= \boldsymbol{b}_2 - u_{1,2}\boldsymbol{b}_1^*,\\ \boldsymbol{b}_3^* &= \boldsymbol{b}_3 - u_{1,3}\boldsymbol{b}_1^* - u_{2,3}\boldsymbol{b}_2^*,\\ &\vdots\\ \boldsymbol{b}_n^* &= \boldsymbol{b}_n - \sum_{i<n} u_{i,n}\boldsymbol{b}_i^*,\end{aligned}$$

where $u_{i,j} = \frac{\boldsymbol{b}_j \cdot \boldsymbol{b}_i^*}{||\boldsymbol{b}_i^*||^2}$. Then, $B^* = (\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \boldsymbol{b}_3^*, \ldots, \boldsymbol{b}_n^*)$ is an orthogonal basis of $\mathbb{R}^n$.

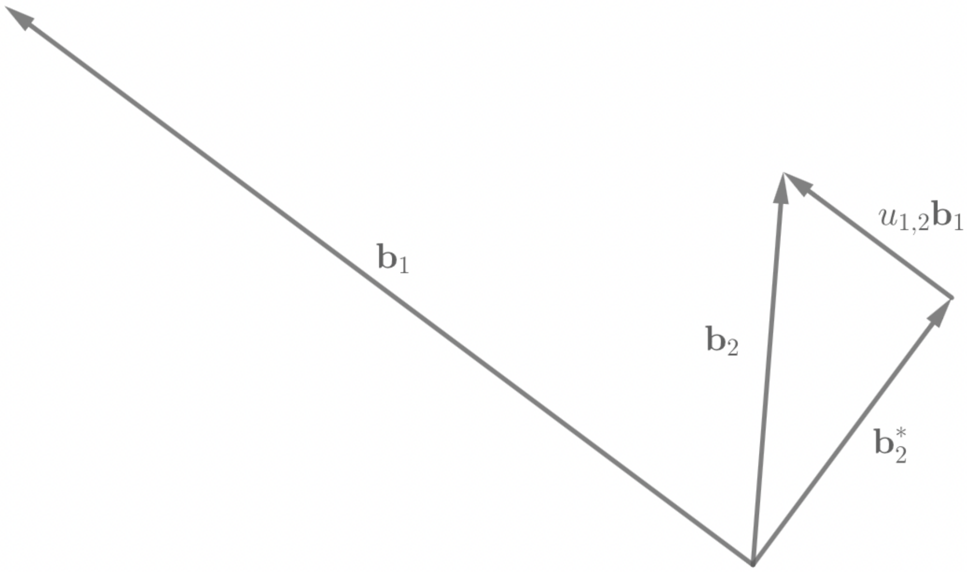

Figure 2. Construction of $\mathbf{b}_2^*$ using orthogonalization process

The intuition behind the Gram-Schmidt Orthogonalization Method and specifically the orthogonalization coefficients, $u_{i,j}$, is that for any basis vector $\mathbf{b}_j$, $u_{i,j}$ is the portion of $\mathbf{b}_j$ that lies on $\mathbf{b}_i^*$. Thus, after deleting the portion of $\mathbf{b}_j$ on $\mathbf{b}_1^*$, the portion of $\mathbf{b}_j$ on $\mathbf{b}_2^*$ and so on, from $\mathbf{b}_j$ to create $\mathbf{b}_j^*$, the result is a vector which is orthogonal to all of the previous orthogonal basis vectors. Let us prove that the Gram-Schmidt Orthogonalization Method does indeed provide an orthogonal basis.

Definition 4.2. The Gram-Schmidt Orthogonalization Method does indeed provide an orthogonal basis.

Proof. We can do this by induction on the number of vectors we are considering, k, and at each step show that the vectors are orthogonal.

For $k = 1$ as the base case, $\mathbf{b}_1^* = \mathbf{b}_1$ and can be considered to be orthogonal. For the inductive case say we have shown that $\mathbf{b}_1^*, \mathbf{b}_2^*, \dots, \mathbf{b}_k^*$ are all orthogonal to each other. Each of the terms, $u_{i,k}\mathbf{b}_i^*$ represent the projection of $\mathbf{b}_i^*$ onto $\mathbf{b}_k$. Thus, after the first subtraction, $\mathbf{b}_{k+1} - u_{1,k}\mathbf{b}_1^*$, the result is a vector orthogonal to $\mathbf{b}_1^*$. Because $\mathbf{b}_1^*$ is orthogonal to $\mathbf{b}_2^*$ by the inductive hypothesis, the projection of $\mathbf{b}_{k+1} - u_{1,k}\mathbf{b}_1^*$ onto $\mathbf{b}_2^*$ is the same as the projection of $\mathbf{b}_k$ onto $\mathbf{b}_2^*$ which is $u_{2,k}\mathbf{b}_2^*$. Thus, after subtraction we get a vector orthogonal to both $\mathbf{b}_1^*$ and $\mathbf{b}_2^*$. As this process continues, we see that $\mathbf{b}_{k+1}^*$ is orthogonal to $\mathbf{b}_i^*$ for $1 \le i \le k$.

To show that the orthogonal basis spans $\mathbb{R}^n$, note that every vector in the original basis, B, can be represented as a linear combination of basis vectors in B^*. Thus, the basis must span $\mathbb{R}^n$. ■

As a corollary of the Gram-Schmidt Orthogonalization method, we have the following.

Corollary 4.3. $||\boldsymbol{b}_i^*|| \le ||\boldsymbol{b}_i||$.

Proof. We have that:

$$\mathbf{b}_k = \sum_{i<k} u_{i,k}\mathbf{b}_i^*.$$

Thus, because all of $\mathbf{b}_1^*, \mathbf{b}_2^*, \dots, \mathbf{b}_k^*$ are orthogonal to each other:

$$||\mathbf{b}_k||^2 = \sum_{i<k} u_{i,k} ||\mathbf{b}_k^*||^2 \Rightarrow ||\mathbf{b}_k||^2 \geq ||\mathbf{b}_k^*||^2 \Rightarrow ||\mathbf{b}_k|| \geq ||\mathbf{b}_k^*||.$$

■

From the Gram-Schmidt Orthogonalization Method, we can see that the smallest vector of lattice over basis $B = (\mathbf{b}_1^*, \mathbf{b}_2^*, \dots, \mathbf{b}_n^*)$ must be larger than the smallest vector of the orthogonal basis.

Corollary 4.4. $\lambda(\mathcal{L}) \geq \min_i ||\boldsymbol{b}_i^*||$.

Proof. Let $\mathbf{v} = \sum_{i=1}^n c_i \mathbf{b}_i$ be some vector of $\mathcal{L}$. Say that the largest i for which c_i is nonzero is $i = k$. Then we have that:

$$\begin{aligned}
\mathbf{v} &= \sum_{i=1}^{k} c_i \mathbf{b}_i, \\
&= \sum_{i=1}^{k} c_i \sum_{j=1}^{i} u_{i,j} \mathbf{b}_j^*, \\
&= \sum_{j=1}^{k} \mathbf{b}_j^* \sum_{i=j}^{k} c_i u_{i,j}, \\
&= c_k \mathbf{b}_k^* + \sum_{j=1}^{k-1} \mathbf{b}_j^* \sum_{i=j}^{k} c_i u_{i,j}.
\end{aligned}$$

Notice that every vector on the sum is based off of $\mathbf{b}_j^*$ where $1 \leq j \leq k-1$, so all of those vectors are orthogonal to $\mathbf{b}_k^*$, so the sum itself is orthogonal to $\mathbf{b}_k^*$. As a result, we see that:

$$\begin{aligned}
& ||\mathbf{v}||^2 = ||c_k \mathbf{b}_k^*||^2 + ||\sum_{j=1}^{k-1} \mathbf{b}_j^* \sum_{i=j}^{k} c_i u_{i,j}||^2, \\
\Rightarrow \quad & ||\mathbf{v}||^2 \geq ||c_k \mathbf{b}_k^*||^2, \\
\Rightarrow \quad & ||\mathbf{v}|| \geq ||c_k \mathbf{b}_k^*||.
\end{aligned}$$

Therefore, $||\mathbf{v}|| \geq ||c_k \mathbf{b}_k^*|| \geq ||\mathbf{b}_k^*||$. Thus, any vector of the lattice must have a magnitude larger than one of the basis vectors in the Gram-Schmidt Orthogonalized basis, meaning that the shortest vector is longer than $\min(||\mathbf{b}_1^*||, ||\mathbf{b}_2^*||, \dots, ||\mathbf{b}_n^*||)$. ■

5. LLL Reduction

The main problem occurs when SVP is extended to higher dimensions. SVP is solved in 2-dimensions using a two dimensional basis and Gauss's Algorithm. But extending SVP to higher dimensions remains an open problem. In the two dimension case we noticed that to reduce a base we had to make the basis vectors as orthogonal as possible. Extracting this same idea, A. Lenstra, H. Lenstra, and L. Lovasz came up with an approximation called the LLL algorithm.

Definition 5.1. Let $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, \dots \mathbf{b}_n\}$ be a n-dimensional basis and define the Gram-Schmidt Process orthogonal basis to be $\mathbf{b}^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \dots \mathbf{b}_n^*\}$. Let the Gram-Schmidt coefficients be

$$u_{i,j} = \frac{\mathbf{b}_j \cdot \mathbf{b}_i^*}{\mathbf{b}_i^* \cdot \mathbf{b}_i^*}$$

for any i, j. An LLL-reduced basis satisfies:

- (Size Reduction Condition) For all $i \neq j$, $u_{i,j} \leq \frac{1}{2}$.
- (Lovasz Condition) For each i,
$$\delta ||\mathbf{b}_i^*||^2 \leq ||\mathbf{b}_{i+1}^* + u_{i+1,i}\mathbf{b}_i^*||^2$$
for some $\delta \in [0.25, 1)$.

The first condition is called the size-reduction and by definition it guarantees the length reduction of the basis. The second condition is called the Lovasz Condition which ensures the ordering of the basis. The δ is introduced because polynomial-time complexity is only guaranteed for $\delta \in [0.25, 1)$. In the original paper, A. Lenstra, H. Lenstra and L. Lovász used $\delta = \frac{3}{4}$.

Theorem 5.2. *Using this idea of a reduced basis, we have the LLL-reduction algorithm as follows. Given some basis $B = (\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \dots, \boldsymbol{b}_n)$,*

1. *Gram Schmidt - Perform the Gram-Schmidt Orthogonalization to obtain the basis $B = (\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \dots, \boldsymbol{b}_n^*)$.*
2. *Reduction step - For each $1 \leq i \leq n$, for each j from $i-1$ to 1, let $m_{j,i} = \lfloor u_{j,i} \rceil$ and make $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i - m_{j,i}\boldsymbol{b}_j$.*
3. *Ordering step - For every pair of consecutive vectors, $\boldsymbol{b}_i, \boldsymbol{b}_{i+1}$, check that $\delta||\boldsymbol{b}_i^*||^2 \leq ||\boldsymbol{b}_{i+1}^* + u_{i,i+1}\boldsymbol{b}_i^*||^2$. If not, swap the two and return back to step 1.*

Example. Let us perform the LLL algorithm to reduce the basis of

$$\mathbf{b}_1 = \langle 17, 20, 34 \rangle, \mathbf{b}_2 = \langle 60, 1, 15 \rangle, \mathbf{b}_3 = \langle 3, 6, 50 \rangle.$$

We start by performing the Gram-Schmidt orthogonalization to get the vectors

$$\mathbf{b}_1^* = \langle 70, 20, 34 \rangle, \mathbf{b}_2^* = \langle 45.718, -15.802, -13.564 \rangle, \mathbf{b}_3^* = \langle -2.722, -18.263, 12.104 \rangle.$$

We can now reduce all of the vectors. For $\mathbf{b}_2$, we have that $m_{1,2} = \lfloor u_{1,2} \rceil = 1$. Thus, $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - \mathbf{b}_1 = \langle 43, -19, -19 \rangle$. For $\mathbf{b}_3$, we need to reduce first by $\mathbf{b}_2$ and then by $\mathbf{b}_1$. We have $m_{2,3} = \lfloor u_{2,3} \rceil = 0$, so $\mathbf{b}_3$ does not change. Also, $m_{1,3} = \lfloor u_{1,3} \rceil = 1$, so $x_3 \leftarrow x_3 - x_1 = \langle -14, -14, 16 \rangle$.

Let us start ordering the vectors. In order to check the Lovasz Condition, we can simplify it in order to reduce the amount of work.

$$\delta||\mathbf{b}_i^*||^2 \leq ||\mathbf{b}_{i+1}^* + u_{i+1,i}\mathbf{b}_i^*||^2.$$
$$\Rightarrow \delta||\mathbf{b}_i^*||^2 \leq ||\mathbf{b}_{i+1}^*||^2 + u_{i,i+1}^2||\mathbf{b}_i^*||^2.$$
$$\Rightarrow ||\mathbf{b}_{i+1}^*||^2 \geq (\delta - u_{i,i+1}^2)||\mathbf{b}_i^*||^2.$$

For $i = 1$, we have that $||\mathbf{b}_2^*||^2 = 2523.83$ and $(\delta - u_{1,2}^2)||\mathbf{b}_1^*||^2 = 1336.58$. Thus, this pair satisfies the Lovasz condition. For $i = 2$ on the other hand, we have that $||\mathbf{b}_3^*||^2 = 487.44$ and $(\delta - u_{2,3}^2)||\mathbf{b}_2^*||^2 = 1732.68$ which does not satisfy the Lovasz condition. Thus, we must swap $\mathbf{b}_2$ and $\mathbf{b}_3$. The vectors are now:

$$\mathbf{b}_1 = \langle 17, 20, 34 \rangle, \mathbf{b}_2 = \langle -14, -14, 16 \rangle, \mathbf{b}_3 = \langle 43, -19, -19 \rangle.$$

Performing the Gram-Schmidt Orthogonalization, we have that:

$$\mathbf{b}_1^* = \langle 17, 20, 34\rangle, \mathbf{b}_2^* = \langle -14.240, -14.282, 15.54\rangle, \mathbf{b}_3^* = \langle 31.738, -29.824, 1.675\rangle.$$

We can now reduce the vectors. We have that $m_{1,2} = \lfloor u_{1,2} \rceil = 0$, so $\mathbf{b}_2$ does not change. As for $\mathbf{b}_3$, reducing by $\mathbf{b}_2$, we get $m_{2,3} = \lfloor u_{2,3} \rceil = -1$, so $\mathbf{b}_3 \leftarrow \mathbf{b}_3 + \mathbf{b}_2 = \langle 29, -33, -3\rangle$. Reducing by $\mathbf{b}_1$, we have $m_{1,3} = \lfloor u_{1,3} \rceil = 0$, so $\mathbf{b}_3$ does not change.

Checking the ordering, we have $||\mathbf{b}_2^*||^2 = 647.63$ and $(\delta - u_{1,2}^2)||\mathbf{b}_1^*||^2 = 1383.38$, breaking the Lovasz condition, so we have to swap the two vectors.

Our basis is now:

$$\mathbf{b}_1 = \langle -14, -14, 16\rangle, \mathbf{b}_2 = \langle 17, 20, 34\rangle, \mathbf{b}_3 = \langle 29, -33, -3\rangle.$$

The Gram-Schmidt Orthogonalized basis is then:

$$\mathbf{b}_1^* = \langle -14, -14, 16\rangle, \mathbf{b}_2^* = \langle 17.562, 20.562, 33.358\rangle, \mathbf{b}_3^* = \langle 31.738, -29.824, 1.675\rangle.$$

Reducing the vectors, we see that $m_{1,2} = \lfloor u_{1,2} \rceil = 0$, $m_{2,3} = \lfloor u_{2,3} \rceil = 0$, and $m_{1,3} = \lfloor u_{1,3} \rceil = 0$, so nothing changes as a result of the reduction.

Checking the ordering of the basis vectors, we have that:

$$1843.96 = ||\mathbf{b}_2^*||^2 \geq (\delta - u_{1,2}^2)||\mathbf{b}_1^*||^2 = 484.96.$$
$$1899.565 = ||\mathbf{b}_3^*||^2 \geq (\delta - u_{2,3}^2)||\mathbf{b}_2^*||^2 = 1343.63.$$

Thus, the basis,

$$\mathbf{b}_1 = \langle -14, -14, 16\rangle, \mathbf{b}_2 = \langle 17, 20, 34\rangle, \mathbf{b}_3 = \langle 29, -33, -3\rangle,$$

is an LLL reduced basis.

We will now present some properties of the LLL reduced basis.

Theorem 5.3. *Let $\boldsymbol{b} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \dots \boldsymbol{b}_n\}$ be an n-dimensional LLL reduced basis of lattice $\mathcal{L}$. Then*

$$||\boldsymbol{b}_1|| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \cdot \lambda(\mathcal{L})$$

for some $\delta \in [0.25, 1)$ and where $\lambda(\mathcal{L})$ is the shortest vector in $\mathcal{L}$.

Proof. Note that by our definition of LLL base reduction we have that

$$||\mathbf{b}_i^*||^2 \leq \frac{1}{\delta}||\mathbf{b}_{i+1}^* + u_{i,i+1}\mathbf{b}_i^*||^2$$
$$= \frac{1}{\delta}||\mathbf{b}_{i+1}^*||^2 + \frac{1}{\delta}u_{i,i+1}^2||\mathbf{b}_i^*||^2.$$

By the size-reduction condition we know that $u_{i,i+1} \leq \frac{1}{2}$ which means that $u_{i,i+1}^2 \leq \frac{1}{4}$. Hence

$$||\mathbf{b}_i||^2 \leq \frac{1}{\delta}||\mathbf{b}_{i+1}^*||^2 + \frac{1}{4\delta}||\mathbf{b}_i^*||^2 \Rightarrow \frac{4\delta - 1}{4}||\mathbf{b}_i^*||^2 \leq ||\mathbf{b}_{i+1}^*||^2$$

By chaining the inequalities:

$$\frac{4\delta-1}{4}||\mathbf{b}_i^*||^2 \leq ||\mathbf{b}_{i+1}^*||^2,$$
$$\frac{4\delta-1}{4}||\mathbf{b}_{i-1}^*||^2 \leq ||\mathbf{b}_i^*||^2,$$
$$\vdots$$
$$\frac{4\delta-1}{4}||\mathbf{b}_1^*||^2 \leq ||\mathbf{b}_2^*||^2,$$

we get

$$||\mathbf{b}_{i+1}^*||^2 \geq \left(\frac{4\delta-1}{4}\right)^i ||\mathbf{b}_1||^2.$$

Note that as a result of Corollary 4.4 from before, $\lambda(\mathcal{L}) \geq \min(||\mathbf{b}_i^*||)$. Therefore, we know that:

$$\lambda(\mathcal{L})^2 \geq \min_i(||\mathbf{b}_i^*||) \geq \left(\frac{4\delta-1}{4}\right)^{n-1} ||\mathbf{b}_1||^2.$$

Taking the square root of both sides of the inequality, we find that:

$$\lambda(\mathcal{L}) \geq \left(\frac{4\delta-1}{4}\right)^{\frac{n-1}{2}} ||\mathbf{b}_1|| \Rightarrow ||\mathbf{b}_1|| \leq \left(\frac{2}{\sqrt{4\delta-1}}\right)^{n-1} \cdot \lambda(\mathcal{L})$$

■

6. Time Complexity

Analyzing the time complexity of the LLL algorithm is a complicated task. We will first bound the number of iterations and then find the time for a single iteration. Before we delve into that let's look at the potential of a lattice base.

For the sake of readability we will let $X = \max_i ||\mathbf{b}_i||$. Let us start by defining the potential of a basis.

Definition 6.1. Let $B = \{\mathbf{b}_1, \mathbf{b}_2, \dots \mathbf{b}_n\}$, and let the Gram-Schmidt Orthogonalized basis of B be $B^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \dots \mathbf{b}_n^*\}$ The potential of B, denoted by D_B, is given by

$$D_B = \prod_{i=1}^{n} ||\mathbf{b}_i^*||^{n+1-i} = \prod_{i=1}^{n} ||\mathbf{b}_1^*|| \cdot ||\mathbf{b}_2^*|| \cdot ||\mathbf{b}_3^*|| \cdots ||\mathbf{b}_i^*|| = \prod_{i=1}^{n} D_{B,i}$$

where $D_{B,i} = \det(\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_i)) = ||\mathbf{b}_1^*|| \cdot ||\mathbf{b}_2^*|| \cdots ||\mathbf{b}_i^*||$ where $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_i)$ denotes the lattice formed by the basis vectors $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_i$.

We can make a few observations about this potential value as the algorithm proceeds.

Proposition 6.2. $D_B < X^{n(n+1)/2}$.

Proof. For the upper bound, note that $||\mathbf{b}_i^*|| < ||\mathbf{b}_i|| < \max_i ||\mathbf{b}_i|| = X$. Thus, we have that:

$$D_B = \prod_{i=1}^{n} ||\mathbf{b}_i^*||^{n+1-i} < \prod_{i=1}^{n} X^{n+1-i} = X^{1+2+3+\ldots+n} = X^{n(n+1)/2}.$$

■

Proposition 6.3. *D_B where B spans some fixed n-dimensional lattice, $\mathcal{L}$, has a positive lower bound.*

Proof. We have that $\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots$ are all positively lower bounded because the basis vectors are contained within a fixed lattice. As a result D_B must have a positive lower bound as well. ∎

Proposition 6.4. *During the reduction step (step 2) of the algorithm, D_B does not change.*

Proof. Note that D_B is solely based off of the Gram-Schmidt basis vectors which do not change during the reduction step. ∎

Proposition 6.5. *The ordering step only decreases D_B by a constant factor.*

Proof. Let the new potential be D'_B with $D'_{B,i} = ||\mathbf{b}_1^{*\prime}|| \cdot ||\mathbf{b}_2^{*\prime}|| \cdots ||\mathbf{b}_i^{*\prime}||$. Say that vectors $\mathbf{b}_k$ and $\mathbf{b}_{k+1}$ are swapped. Let the new Gram-Schmidt orthogonalized basis be $\mathbf{b}_1^{*\prime}, \mathbf{b}_2^{*\prime}, \ldots, \mathbf{b}_n^{*\prime}$. For the vectors $\mathbf{b}_i^{*\prime}$ where $1 \le i \le k-1$, they remain the exact same as there counter parts in B^*. As for $\mathbf{b}_k^{*\prime}$, we have:

$$\mathbf{b}_k^{*\prime} = \mathbf{b}_k + 1 - \frac{\mathbf{b}_{k+1} \cdot \mathbf{b}_1^*}{||\mathbf{b}_1^*||^2}\mathbf{b}_1^* - \frac{\mathbf{b}_{k+1} \cdot \mathbf{b}_2^*}{||\mathbf{b}_2^*||^2}\mathbf{b}_2^* - \ldots - \frac{\mathbf{b}_{k+1} \cdot \mathbf{b}_{k-1}^*}{||\mathbf{b}_{k-1}^*||^2}\mathbf{b}_{k-1}^* = \mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*.$$

Thus, $D_{B,i} = D'_{B,i}$ for $1 \le i \le k-1$, $D'_{B,k} = D_{B,k}\frac{\mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*}{\mathbf{b}_k^*}$, and $D'_{B,i} = D_{B,i}$ for $k+1 \le i \le n$ because both are determinants of a lattice formed by the same basis vectors. Thus, D'_B, is:

$$\begin{aligned}
D'_B &= \prod_{i=1}^{n} D'_{B,i}, \\
&= \prod_{i=1}^{k-1} D'_{B,i} \cdot D'_{B,k} \cdot \prod_{i=k+1}^{n} D'_{B,i}, \\
&= \prod_{i=1}^{k-1} D_{B,i} \cdot D_{B,k}\frac{||\mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*||}{||\mathbf{b}_k^*||} \cdot \prod_{i=k+1}^{n} D_{B,i}, \\
&= \frac{||\mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*||}{||\mathbf{b}_k^*||} \prod_{i=1}^{n} D_{B,i}, \\
&= \frac{||\mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*||}{||\mathbf{b}_k^*||} D_B.
\end{aligned}$$

Notice that as a result of the Lovasz condition being broken between $\mathbf{b}_k$ and $\mathbf{b}_{k+1}$ as a result of the swap, we know that:

$$\delta||\mathbf{b}_k^*||^2 > ||\mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*||^2 \Rightarrow \sqrt{\delta} > \frac{||\mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*||}{||\mathbf{b}_k^*||}.$$

Thus,

$$D'_B = \frac{||\mathbf{b}_{k+1}^* + u_{k,k+1}\mathbf{b}_k^*||}{||\mathbf{b}_k^*||} D_B < \sqrt{\delta} D_B.$$

Thus, as long as $\delta < 1$ which is true since $\delta \in [0.25, 1)$, D_B decreases by a constant factor. ∎

Lemma 6.6. *The number of iterations is some polynomial in terms of n.*

Proof. After every iteration, the potential decreases by a factor of at most $\sqrt{\delta}$, and the potential also has an upper bound. Because all of the basis in the algorithm represent the same lattice, $\mathcal{L}$, D_B also has a lower bound, q. Thus, the number of iterations is at most:

$$\log_{\sqrt{\delta}} \frac{X^{n(n+1)/2}}{q} = n(n+1)/2 \log_{\sqrt{\delta}} X - \log_{\sqrt{\delta}} q = O(n^2).$$

Thus, the number of iterations is some polynomial on n. ■

Lemma 6.7. *Each iteration happens in polynomial time.*

Proof. For each iteration, there are three steps: Gram-Schmidt, Reduction step, and Ordering step. Let us show that each of them operate on polynomial time.

Let us start with Gram-Schmidt. To prove this, let us break up Gram-Schmidt into smaller parts. Primarily, Gram-Schmidt has to be able to calculate $u_{i,j}\mathbf{b}_i^*$. In order to do this, it must perform 2 dot products each which is in polynomial time and also must multiply $u_{i,j}$ by $\mathbf{b}_i^*$ which also happens in polynomial time on n. This must be done on the order of $O(n^2)$ times for finding the Gram-Schmidt Orthogonalized basis, so Gram-Schmidt as a whole operates on polynomial time.

Moving on, the reduction step has the exact same structure as Gram-Schmidt except, the orthogonalization coefficients are rounded instead. This does not change the fact that the algorithm operates on polynomial time, so the reduction step operates on polynomial time.

Finally, for the Ordering step, the algorithm just has to verify an inequality involving a Gram-Schmidt orthogonalization coefficient for consecutive pairs of terms. Verifying the inequality happens in polynomial time, and the orthogonalization coefficient can be calculated in polynomial time. Thus, because there are at most $n-1$ consecutive pairs of basis vectors, the Ordering step must operate on polynomial time. ■

Theorem 6.8. *The LLL algorithm operates in polynomial time based on n.*

Proof. Thus, because each iteration operates on polynomial time and the number of iterations is some polynomial in terms of n by Lemmas 6.6 and 6.7, the LLL algorithm must operate on polynomial time as well. ■

7. Mertens Conjecture

One well known application of the LLL Algorithm is in disproving Mertens conjecture, which states that the Mertens function $M(n)$ is bounded by $\pm\sqrt{n}$. Conjectured in 1885 by Thomas Joannes Stieltjes, it was disproved 100 years later by Andrew Odlyzko and Herman te Riele in 1985. The Mertens Conjecture holds a lot of significance because $|M(n)| \leq k\sqrt{n}$ for any constant k would imply the Riemann Hypothesis, since $M(n) = O(n^{1/2+\epsilon})$ for any $\epsilon < \frac{1}{2}$ is equivalent to the Riemann Hypothesis.

Definition 7.1 (Mertens Function)**.** The Mertens Function, $M(n)$, is defined as follows:

$$M(n) = \sum_{k=1}^{n} \mu(k),$$

where $\mu(k)$ denotes the Möbius function, which denotes the sum of the primitive k-th roots of unity. Note that the values of $\mu(k)$ is contained in $\{-1, 0, 1\}$.

Theorem 7.2 (Mertens Conjecture)**.** *The Mertens Conjecture states that for all $n > 1$,*

$$|M(n)| < \sqrt{n}$$

Through the application of the LLL Algorithm, Andrew Odlyzko and Herman te Riele showed that

$$\lim_{n\to\infty} \sup m(n) > 1.06 \text{ and } \lim_{n\Rightarrow\infty} \inf m(n) < -1.009,$$

where $m(n) = \frac{M(n)}{\sqrt{n}}$. Although this does not directly prove that there actually exists such n, a counterexample was later found to exist below $e^{3.21\cdot 10^{64}}$. As of now, it is not certain, but seems very likely that

$$\lim_{n\to\infty} \sup m(n) = \infty.$$

8. Cryptography

The LLL Algorithm is also prevalent throughout cryptography, and one notable example of this is the Knapsack Problem. It goes as follows: Consider a thief who breaks into a jewelry store. Each gem in the jewelry store has a weight and is valued at a certain price. Which gems should he steal in order to maximize the value of the gems?

This Knapsack Problem, also known as the Subset Sum Problem, can be extended to the Cryptographic Knapsack Scheme, which is one of the earliest public key cryptosystems, founded by Ralph Merkle and Martin Hellman in 1973. It implements the subset sum problem, and is mostly unsolvable, but there are instances that can be solved by the LLL algorithm.

Suppose Bob wants to send a message to Alice, and Alice's public key is $a = (a_1, a_2, \cdots, a_n)$. He wants to encipher a message $x = (x_1, x_2, \cdots, x_n)$, and sends the sum

$$S = \sum_{i=1}^{n} a_i x_i$$

to Alice. If S is potentially eavesdropped and the enciphering key is public, finding the message x from S and a should be intentionally hard. Given a random sequence of integers for a, the only way to find x is to try all 2^n possible values of x. However, if a is chosen randomly, it is also extremely hard for Alice to decipher the message. This can be solved with the Merkle-Hellman trapdoor, which gives her information called the deciphering key. Funnily enough, this trapdoor is also what makes this system insecure. To see how this trapdoor works, let's first define a super-increasing sequence.

Definition 8.1. A sequence $b_1, b_2, \cdots, b_n$ is *super-increasing* if for each $2 \leq i \leq n$,

$$b_i > \sum_{j=1}^{i-1} b_j.$$

Determining whether or not a sequence is super-increasing only takes 1 pass, which is $O(n)$, so determining whether a subset sum, T, is part of a super-increasing set, the computer must find the largest number in the set less than or equal to T then subtract it to get T'. It repeats this process with T'. If $T' = 0$ then the subset sum consists of all the numbers subtracted from T.

Modulo transformations can then be used on the private key to generate a public key, and by having secret information called the deciphering key, through this trapdoor and several modulo transformations, Alice can iteratively determine the message.

To show how LLL Algorithm is useful in cracking this scheme, let's suppose we have the following superincreasing knapsack

$$S = [2, 5, 9, 24, 45, 103, 215, 450, 946],$$

which is our private key. Our public knapsack is $t_i = 1289 \cdot s_i \mod 2003$, so our public key

$$T = [575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570].$$

We encrypt 101100111 as $575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665$.

Given that someone wants to intercept the messages and knows that the public key $T = [575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570]$ and ciphertext 6665, they can't easily decipher the message because they do not possess the deciphering key. Thus, they have to directly find $x_i \in \{0, 1\}$ such that

$$575x_0 + 436x_1 + 1586x_2 + 1030x_3 + 1921x_4 + 569x_5 + 721x_6 + 1183x_7 + 1570x_8 = 6665.$$

This can be written as $T \cdot X = 6665$. Lets rewrite the problem as a matrix below:

$$M = \begin{bmatrix} I_{9\times9} & 0_{1\times9} \\ T_{9\times1} & -C_{1\times1} \end{bmatrix} = \left[\begin{array}{ccccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 575 & 436 & 1586 & 1030 & 1921 & 569 & 721 & 1183 & 1570 & -6665 \end{array}\right]$$

We can think of the matrix as comprising of vertical basis vectors. The solution then can be written as the vector $(x_0, x_1, \ldots, x_8, 0)$ which is an integer combination of the other basis vectors. Note that this solution is one of the shorter vectors in the lattice spanned by the columns of M because we only get a solution when the vector has 1s and 0s in the first 9 entries and a 0 in the last entry. Thus, applying LLL to reduce the vectors we get:

$$M' = \left[\begin{array}{ccccccccc|c}
-2 & 0 & 1 & 0 & 1 & 0 & 1 & -1 & -2 & 1 \\
-1 & 0 & -1 & 1 & -1 & 0 & 0 & -1 & 0 & 1 \\
1 & 0 & 0 & -1 & -1 & -1 & 1 & 0 & -1 & 0 \\
0 & -1 & -2 & -1 & 1 & 0 & 1 & 1 & 0 & -1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & -1 & -1 & -2 \\
0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 2 & -1 & -1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & -1 & 0 & -1 & 1 & 1 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 \\
\hline
0 & 1 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0
\end{array}\right]$$

And indeed, the 7-th column has the right form, which gives us our solution of 101100111 which displays the LLL-Algorithm's application in cryptography.

Appendix

Code for the LLL algorithm:

```python
import sys
import json
import numpy as np
from numpy import linalg as la
# Initialize the basis as the user input.
basis = np.array(json.loads(sys.argv[1])).astype(float)
 # Initialize the Gram-Schmidt basis.
orthobasis = basis.copy()

k = 1 # Initialize the working index.
DELTA = 0.75

def projection_scale(u, v):
    '''Computes <u,v>/<u,u>, which is the scale used in projection.'''
    return np.dot(u, v) / np.dot(u, u)

def proj(u, v):
    '''Computes the projection of vector v onto vector u.
    Assumes u is not zero.'''
    return np.dot(projection_scale(u, v), u)

def gram_schmidt():
    '''Computes Gram Schmidt orthoganalization
    (without normalization) of a basis.'''
    orthobasis[0] = basis[0]
    for i in range(1, basis.shape[1]):  # Loop through dimension of basis.
        orthobasis[i] = basis[i]
        for j in range(0, i):
            orthobasis[i] -= proj(orthobasis[j], basis[i])
    return orthobasis

def lovasz():
    global k
    '''Checks the Lovasz condition for a basis.
    Either swaps adjacent basis vectors and
    recomputes Gram-Scmidt or increments the working index.'''
    c = DELTA - projection_scale(orthobasis[k-1], basis[k])**2
    # Check the Lovasz condition.
    if la.norm(orthobasis[k])**2 >= np.dot(c, la.norm(orthobasis[k-1]**2)):
        # Increment k if the condition is met.
        k += 1
    else:
        # If the condition is not met,
        # swap the working vector and
        # the immediately preceding basis vector.
        basis[[k, k-1]] = basis[[k-1, k]]
        gram_schmidt() # Recompute Gram-Schmidt if swap
        k = max([k-1, 1])

def reduction():
    '''Performs length reduction on a basis.'''
     # Track the total amount by which the working vector is reduced.
    total_reduction = 0
    for j in range(k-1, -1, -1):   # j loop. Loop down from k-1 to 0.
        m = round(projection_scale(orthobasis[j], basis[k]))
        total_reduction += np.dot(m, basis[j])[0]
        # Reduce the working vector by multiples of preceding vectors.
        basis[k] -= np.dot(m, basis[j])
    if total_reduction > 0:
        # Recompute Gram-Scmidt if the working vector has been reduced.
        gram_schmidt()

def main():
    while True:
        x = raw_input("See the steps? Press [Y/N] and Enter. ")
        if x in ['Y','N', 'y', 'n']: break
        else: raw_input("See the steps? Press [Y/N] and Enter. ")
    if x in ['Y', 'y']:
        gram_schmidt()
        steps = 0
        while k <= basis.shape[1] - 1:
            reduction()
            steps += 1
            print 'Step ', steps,'. After the reduction step, the basis is\n',
            basis raw_input("")
            lovasz()
            steps +=1
            print 'Step ', steps,'. After checking the Lovasz condition,
            the basis is\n', basis
            raw_input("")
        print 'LLL Reduced Basis:\n', basis
    else:
        gram_schmidt(basis)
        while k<= basis.shape[1] - 1:
            reduction()
            lovasz()
        print 'LLL Reduced Basis:\n', basis

if __name__ == "__main__":
    main()
```

References

[1] L. Lovász A. K. Lenstra, H. W. Lenstra. Factoring polynomials with rational coefficients, 1982.
[2] Jennifer Bakker. The knapsack problem and the lll algorithm, 2004.
[3] Xinyue Deng. An introduction to lenstra-lenstra-lovasz lattice basis reduction algorithm, 2016.
[4] Swastik Kopparty. Algorithmic number theory, 2014.
[5] Vinod Vaikuntanathan. Advanced topics in cryptography: Lattices, 2015.