LLL Algorithm

Debadatta Kar, PhD Scholar,
Electrical Engineering and Computer Science,
Indian Institute of Science Education and Research, Bhopal

July, 2025

$\boxed{V}$

---

1 Introduction

The Lenstra-Lenstra-Lovász (LLL) algorithm, introduced in 1982, is a popular algorithm in cryptography. This algorithm appeared in the paper titled "Factoring Polynomials with Rational Coefficients" in the journal Mathematische Annalen. LLL is a lattice basis reduction algorithm. Given a basis in a lattice, the LLL algorithm efficiently transforms it into an LLL-reduced basis.
Its applications are vast, ranging from factoring polynomials over rational numbers to solving integer programming problems. It also plays a vital role in analysing the security of modern lattice-based cryptographic schemes, which are considered promising candidates for quantum-resistant cryptography.

2 Shortest Vector problem(SVP)

Input: Basis of lattice B
Output: $\mathbf{v} \in \mathcal{L}$ such that

$$||\mathbf{v}|| = \lambda_1$$

This problem specifically finds a short vector v from the lattice vectors.

3 LLL Reduced Basis

The LLL algorithm (triple L algorithm, 3L algorithm) is the most widely used lattice reduction algorithm. It does not provide an exact solution to SVP, but an approximation to SVP.

Open Problem: Does there exist a better algorithm that can solve SVP in higher dimensions or have a better time complexity than the LLL algorithm?

Definition 3.1. **($\delta-$LLL Reduced Basis)** *Let $B = \mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_n}$ be lattice basis and associated GSO basis be $B^* = \{\mathbf{b_1^*}, \mathbf{b_2^*}, \cdots, \mathbf{b_n^*}\}$. B is said to be $\delta-$LLL reduced if*

1. $|\mu_{i,j}| \leq 1/2 \quad \forall 1 \leq j < i < n$ **(size condition)**
2. $\delta||\mathbf{b_i^*}||^2 \leq ||\mu_{i+1,i}\mathbf{b_i^*} + \mathbf{b_{i+1}^*}||^2$ **(Lovász condition)**

✓ *Where,* $\mu_{i,j} = \frac{\langle \mathbf{b_i}, \mathbf{b_j^*} \rangle}{\langle \mathbf{b_j^*}, \mathbf{b_j^*} \rangle}$

✓ *and,* $\delta \in (0.25, 1)$

Definition 3.2. (LLL Reduced Basis ($\delta = 3/4$)**)** *Let* $B = \mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_n}$ *be lattice basis and associated GSO basis be* $B^* = \{\mathbf{b_1^*}, \mathbf{b_2^*}, \cdots, \mathbf{b_n^*}\}$*. B is said to be LLL reduced if*

1. $|\mu_{i,j}| \leq 1/2 \quad \forall 1 \leq j < i < n$ **(size condition)**
2. $\frac{3}{4}||\mathbf{b_i^*}||^2 \leq ||\mu_{i+1,i}\mathbf{b_i^*} + \mathbf{b_{i+1}^*}||^2$ **(Lovász condition)**

✓ *Where,* $\mu_{i,j} = \frac{\langle \mathbf{b_i}, \mathbf{b_j^*} \rangle}{\langle \mathbf{b_j^*}, \mathbf{b_j^*} \rangle}$

The Lovász condition can also be simplified as

$$\begin{aligned}
\delta||\mathbf{b_i^*}||^2 &\leq ||\mu_{i+1,i}\mathbf{b_i^*} + \mathbf{b_{i+1}^*}|| \\
&= |\mu_{i+1,i}\mathbf{b_i^*}||^2 + ||\mathbf{b_{i+1}^*}|| \\
&= \mu_{i+1,i}^2||\mathbf{b_i^*}||^2 + ||\mathbf{b_{i+1}^*}|| \\
&\leq \left(\frac{1}{2}\right)^2 ||\mathbf{b_i^*}||^2 + ||\mathbf{b_{i+1}^*}|| \quad \text{(maximum value of } \mu_{i+1,i} = 1/2) \\
\delta||\mathbf{b_i^*}||^2 &\leq \frac{1}{4}\|\mathbf{b_i^*}\|^2 + \|\mathbf{b_{i+1}^*}\|^2 \\
\|\mathbf{b_{i+1}^*}\|^2 &\geq \left(\delta - \frac{1}{4}\right) ||\mathbf{b_i^*}||^2
\end{aligned}$$

So, one can also write Lovász condition as

$$\|\mathbf{b_{i+1}^*}\|^2 \geq \left(\delta - \frac{1}{4}\right) ||\mathbf{b_i^*}||^2$$

when $\delta = 3/4$,

$$\|\mathbf{b_{i+1}^*}\|^2 \geq \frac{1}{2}||\mathbf{b_i^*}||^2$$

Lemma 3.3. *In a LLL reduced basis B we have* $||\mathbf{b_{i+1}^*}||^2 \geq \frac{1}{2}||\mathbf{b_i^*}||^2$ *for* $1 \leq i < n$

Proof. Proof follows from the arguments just discussed above on the simplification of Lovász condition □

Theorem 3.4. *Let* $\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_n}$ *be* $\delta - LLL$ *reduced, then*

$$\|\mathbf{b_1}\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right) \lambda_1(\mathcal{L})$$

Proof. We have already shown,

$$\|\mathbf{b_{i+1}^*}\|^2 \geq \left(\delta - \frac{1}{4}\right) ||\mathbf{b_i^*}||^2$$

Thus we can write

$$\left(\frac{4}{4\delta - 1}\right) ||\mathbf{b_{i+1}^*}\|^2 \geq ||\mathbf{b_i^*}||^2$$

Applying induction on this, we get,

$$||\mathbf{b_1}||^2 \leq \left(\frac{4}{4\delta-1}\right) ||\mathbf{b_2^*}||^2 \leq \cdots \leq \left(\frac{4}{4\delta-1}\right)^{i-1} ||\mathbf{b_i^*}||^2 \leq \cdots \leq \left(\frac{4}{4\delta-1}\right)^{n-1} ||\mathbf{b_n^*}||^2$$

Overall, we have this with us,

$$||\mathbf{b_1}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{i-1} ||\mathbf{b_i^*}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{n-1} ||\mathbf{b_n^*}||^2 \qquad 1 < i < n$$

Taking the minimum we have,

$$||\mathbf{b_1}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{i-1} ||\mathbf{b}^*_{\text{min}}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{n-1} ||\mathbf{b_n^*}||^2$$

Since $i < n$ we have $i-1 < n-1$ and the inequality follows,

$$||\mathbf{b_1}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{i-1} ||\mathbf{b}^*_{\text{min}}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{n-1} ||\mathbf{b}^*_{\text{min}}||^2$$

So we get,

$$||\mathbf{b_1}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{n-1} ||\mathbf{b}^*_{\text{min}}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{n-1} \lambda_1(\mathcal{L})$$

Recall: For any lattice $\mathcal{L}$, we have $\min_i ||\mathbf{b_i^*}|| \leq \lambda_1(\mathcal{L})$

Finally, we have with us,

$$||\mathbf{b_1}||^2 \leq \left(\frac{4}{4\delta-1}\right)^{n-1} \lambda_1(\mathcal{L})$$

and we get,

$$||\mathbf{b_1}|| \leq \left(\frac{2}{\sqrt{4\delta-1}}\right)^{n-1} \lambda_1(\mathcal{L})$$

□

Corollary 3.5. *In an LLL reduced basis B we have $||\mathbf{b_1}|| \leq 2^{(n-1)/2}\lambda_1(\mathcal{L})$*

Proof. Put the value of $\delta = 3/4$ in the previous theorem to obtain,

$$\begin{aligned} ||\mathbf{b_1}|| &\leq \left(\frac{2}{\sqrt{4\times(3/4)-1}}\right)^{n-1} \lambda_1(\mathcal{L}) \\ &\leq \left(\frac{2}{\sqrt{3-1}}\right)^{n-1} \lambda_1(\mathcal{L}) \\ &\leq 2^{\frac{n-1}{2}} \lambda_1(\mathcal{L}) \end{aligned}$$

Thus, we have

$$||\mathbf{b_1}|| \leq 2^{(n-1)/2}\lambda_1(\mathcal{L})$$

□

Algorithm 1 LLL Lattice Reduction Algorithm

```
function LLL($B \in \mathbb{Z}^{n\times n}$)    ▷ Outputs an LLL-reduced basis of the lattice $L(B)$
    $B \leftarrow$ SizeReduce($B$)
    for $i = 1, 2, \cdots, n$ do
        if Lovász condition fails then
            swap $b_i$ and $b_{i+1}$
            LLL($B$)
        end if
    end for
    return $B$
end function
function SizeReduce($B \in \mathbb{Z}^{n\times n}$)  ▷ Outputs a size-reduced basis of $L(B)$, preserving the Gram-Schmidt vectors of $B$
    Compute (or update) the GSO $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \cdots, \mathbf{b_i^*}\}$
    for $j \leftarrow 2$ to $n$ (in any order) do
        for $i \leftarrow j-1$ down to 1 do
            let $u_{i,j} = \frac{\langle \mathbf{b}_j, \mathbf{b}_i^* \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle}$
            let $\mathbf{b}_j \leftarrow \mathbf{b}_j - \lfloor u_{i,j} \rceil \mathbf{b}_i$
        end for
    end for
    return $B$
end function
```

4 LLL Basis Reduction Algorithm

Algorithm 1 is the LLL Lattice reduction algorithm. When fed with a lattice basis B, it generates the LLL reduced basis, which is a "good basis". It alternatively does size-reduction and checks the Lovász condition, on failing, it just swaps the pair. This description intuitively is appropriate to say that the algorithm outputs the LLL reduced basis if it at all terminates.

5 Analysis

5.1 Correctness

The algorithm, if it terminates, will generate the LLL reduced basis, because it only runs when the condition of the LLL reduced basis is violated. If the Lovász condition is satisfied, then the need to swap and iterate again is not required.

5.2 Number of Iterations

Definition 5.1. (Potential Function) *The potential function is a measure of progress of the LLL algorithm, capturing how nice or reduced the basis is. It is defined as*

$$\phi(B) = \prod_{i=1}^{n} \phi_i(B)$$

where,

$$\phi_i(B) = |\det(\mathcal{L}(\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_i}))| = ||\mathbf{b_1^*}||||\mathbf{b_2^*}|| \cdots ||\mathbf{b_i^*}||$$

Lemma 5.2. $\phi(B) \in \mathbb{Z}^+$ *for any step in the LLL algorithm*

Proof. Clearly, $\det(\mathcal{L}(\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_i})) \neq 0$ since all the vectors in a basis are linearly independent. Also, $|\det(\mathcal{L}(\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_i}))|$ defines volume of $i-$dimensional parallelepiped of lattice generated by $\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_i}$. Thus, $\phi_i(B) = |\det(\mathcal{L}(\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_i}))| \in \mathbb{Z}^+$
Finally,

$$\phi(B) = \prod_{i=1}^{n} \phi_i(B) = \prod_{i=1}^{n} \text{(some positive integers)}$$

□

Lemma 5.3. *$\phi(B)$ is not too large to begin with*

Proof.

$$\begin{aligned}
\phi(B_{\text{init}}) &= \prod_{i=1}^{n} \phi_i(B) = \prod_{i=1}^{n} ||\mathbf{b_1^*}||^n ||\mathbf{b_2^*}||^{n-1} \cdots ||\mathbf{b_{n-1}^*}||^2 ||\mathbf{b_n^*}|| \\
&\leq \prod_{i=1}^{n} ||\mathbf{b_1}||^n ||\mathbf{b_2}||^{(n-1)} \cdots ||\mathbf{b_{n-1}}||^2 ||\mathbf{b_n}|| \\
&\leq \max_i ||\mathbf{b_i}||^{n+(n-1)+\cdots+2+1} \\
&\leq \max_i ||\mathbf{b_i}||^{n(n+1)/2} = \max_i ||\mathbf{b_i}||^{O(n^2)} \\
\log(\phi(B_{\text{init}})) &\leq O(n^2) \log(\max_i ||b_i||) \\
\log(\phi(B_{\text{init}})) &\leq O(n^2) \log(\max_i ||b_i||) = \text{poly}(n, W)
\end{aligned}$$

Where W is the bit length of vectors given as input, we call it the input size.

□

Claim: The GSO vector $\mathbf{b_j^*}$ corresponding to the vector $\mathbf{b_j}$ satisfy $\|\mathbf{b_j^*}\| \leq \|\mathbf{b_j}\|$

Proof. We have,

$$\mathbf{b_j} = \mathbf{b_j^*} + \sum_{i<j} \mu_{j,i} \mathbf{b_i^*} = \mathbf{b_j^*} + \mathbf{u}$$

We have, $\mathbf{u} = \sum_{i<j} \mu_{j,i} \mathbf{b_i^*}$, which lies in the span$(\mathbf{b_1^*}, \mathbf{b_2^*}, \cdots, \mathbf{b_{j-1}^*})$
By construction $\mathbf{b_j^*}$ is orthogonal to $\mathbf{u}$ so we apply pythogoras theorem,

$$\|\mathbf{b_j}\|^2 = \|\mathbf{b_j^*} + \mathbf{u}\|^2 = \|\mathbf{b_j^*}\|^2 + \|\mathbf{u}\|^2$$

and finally, we obtain,

$$\|\mathbf{b_j^*}\| \leq \|\mathbf{b_j}\|$$

□

Lemma 5.4. *The reduction step doesn't change the potential function*

Proof. The reduction step does not change the Gram-Schmidt basis.Thus,

$$\phi_i(B) = ||\mathbf{b_1^*}||||\mathbf{b_2^*}|| \cdots ||\mathbf{b_n^*}||$$

□

The next question now is, why does the GSO basis not change in the reduction step? We have purposefully updated GSO in our algorithm once in the beginning, and GSO is neither updated nor computed in the inner for loops.

Let $\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3} \in \mathbb{R}^n$ be lattice basis vectors. We perform Gram–Schmidt orthogonalization to obtain,

$$\begin{aligned} \mathbf{b_1^*} &= \mathbf{b_1} \\ \mathbf{b_2^*} &= \mathbf{b_2} - \mu_{2,1}\mathbf{b_1^*}, \quad \mu_{2,1} = \frac{\langle \mathbf{b_2}, \mathbf{b_1^*}\rangle}{\langle \mathbf{b_1^*}, \mathbf{b_1^*}\rangle} \\ \mathbf{b_3^*} &= \mathbf{b_3} - \mu_{3,1}\mathbf{b_1^*} - \mu_{3,2}\mathbf{b_2^*} \quad \begin{cases} \mu_{3,1} = \frac{\langle \mathbf{b_3}, \mathbf{b_1^*}\rangle}{\langle \mathbf{b_1^*}, \mathbf{b_1^*}\rangle} \\ \mu_{3,2} = \frac{\langle \mathbf{b_3}, \mathbf{b_2^*}\rangle}{\langle \mathbf{b_2^*}, \mathbf{b_2^*}\rangle} \end{cases} \end{aligned}$$

Now perform size reduction on $\mathbf{b_3}$,

$$\mathbf{b_3'} = \mathbf{b_3} - k_2\mathbf{b_2}, \quad \text{where } k_2 = \lfloor \mu_{3,2} \rceil$$

Only $\mathbf{b_3}$ changes. Let's show that $\mathbf{b_1^*}, \mathbf{b_2^*}, \mathbf{b_3^*}$ remain unchanged.

Step 1: $\mathbf{b_1^*}$ is unchanged since $\mathbf{b_1}$ is unchanged.

$$\boxed{\mathbf{b_1^{*'}} = \mathbf{b_1^*} = \mathbf{b_1}}$$

Step 2: $\mathbf{b_2^*}$ is unchanged since $\mathbf{b_2}, \mathbf{b_1^*}$ is not modified.

$$\boxed{\mathbf{b_2^{*'}} = \mathbf{b_2^*} = \mathbf{b_2} - \mu_{2,1}\mathbf{b_1^*}}$$

Step 3: Compute $\mu'_{3,1}$ and $\mu'_{3,2}$,

$$\begin{aligned} \mu'_{3,1} &= \frac{\langle \mathbf{b_3} - k_2\mathbf{b_2}, \mathbf{b_1^*}\rangle}{\langle \mathbf{b_1^*}, \mathbf{b_1^*}\rangle} = \mu_{3,1} - k_2\mu_{2,1} \\ \mu'_{3,2} &= \frac{\langle \mathbf{b_3} - k_2\mathbf{b_2}, \mathbf{b_2^*}\rangle}{\langle \mathbf{b_2^*}, \mathbf{b_2^*}\rangle} = \mu_{3,2} - k_2 \end{aligned}$$

Step 4: Now expand $\mathbf{b_3^{*'}}$ using Gram–Schmidt on $\mathbf{b_3'}$:

$$\begin{aligned} \mathbf{b_3^{*'}} &= \mathbf{b_3'} - \mu'_{3,1}\mathbf{b_1^*} - \mu'_{3,2}\mathbf{b_2^*} \\ &= (\mathbf{b_3} - k_2\mathbf{b_2}) - (\mu_{3,1} - k_2\mu_{2,1})\mathbf{b_1^*} - (\mu_{3,2} - k_2)\mathbf{b_2^*} \\ &= \mathbf{b_3} - \mu_{3,1}\mathbf{b_1^*} - \mu_{3,2}\mathbf{b_2^*} + k_2(\mu_{2,1}\mathbf{b_1^*} + \mathbf{b_2^*} - \mathbf{b_2}) \end{aligned}$$

Note that $\mathbf{b_2} = \mathbf{b_2^*} + \mu_{2,1}\mathbf{b_1^*} \implies -\mathbf{b_2} + \mathbf{b_2^*} + \mu_{2,1}\mathbf{b_1^*} = 0$. Therefore,

$$\boxed{\mathbf{b_3^{*'}} = \mathbf{b_3^*}}$$

The above calculation justifies the fact that the reduction step does not change the Gram-Schmidt basis.

Lemma 5.5. *The swap step reduces ϕ by a constant factor.*

Proof. Let's assume $\mathbf{b_i}, \mathbf{b_{i+1}}$ are swapped.

$$\begin{aligned}
\frac{\phi_i^{\text{new}}(B)}{\phi_i^{\text{old}}(B)} &= \frac{\det\left(\mathcal{L}(\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_{i-1}}, \mathbf{b_{i+1}})\right)}{\det\left(\mathcal{L}(\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_i})\right)} \\
&= \frac{\prod_{j=1}^{i-1}(||\mathbf{b_j^*}||) \times ||\mathbf{b_{i+1}^*}||}{\prod_{j=1}^{i} ||\mathbf{b_j^*}||} \\
&= \frac{||\mathbf{b_{i+1}^*}||}{||\mathbf{b_i^*}||}
\end{aligned}$$

$$\boxed{\begin{aligned}
\mathbf{b_{i+1}^*} &= \mathbf{b_{i+1}} - \sum_{j=1}^{i-1} \mu_{i+1,j}\mathbf{b_j^*} \\
&= \mathbf{b_{i+1}^*} + \sum_{j=1}^{i} \mu_{i+1,i}\mathbf{b_j^*} - \sum_{j=1}^{i-1} \mu_{i+1,j}\mathbf{b_j^*} \\
&= \mathbf{b_{i+1}^*} + \mu_{i+1,i}\mathbf{b_i^*}
\end{aligned}}$$

$$\begin{aligned}
\frac{\phi_i^{\text{new}}(B)}{\phi_i^{\text{old}}(B)} &= \frac{||\mathbf{b_{i+1}^*}||}{||\mathbf{b_i^*}||} \\
&= \frac{||\mathbf{b_{i+1}^*} + \mu_{i+1,i}\mathbf{b_i^*}||}{||\mathbf{b_i^*}||} \\
&< \delta
\end{aligned}$$

Therefore we have, $\phi_i^{\text{new}}(B) \leq \delta\ \phi_i^{\text{old}}(B)$. As long as $\delta < 1$ we know that the potential function decreases by a constant factor. □

We now know that, reduction step doesnot chnage ϕ but swap step changes ϕ with a constant decrement in each step.
Let,

$$\begin{aligned}
&\phi^{(1)} \text{ occurs in first swap} \\
&\phi^{(2)} \text{ occurs in second swap} \\
&\qquad\quad\vdots \\
&\phi^{(m)} \text{ occurs after } m \text{ swaps}
\end{aligned}$$

Thus,

$$\phi^{(m)} < \delta\phi^{(m-1)} \cdots < \delta^{m-1}\phi^{(1)} < \delta^m\phi$$

We also have $\phi^{(m)} \geq 1$, so,

$$\begin{aligned}
&1 \leq \phi^{(m)} < \delta\phi^{(m-1)} \cdots < \delta^{m-1}\phi^{(1)} < \delta^m\phi \\
\implies &\delta^m\phi \geq 1 \\
\implies &\phi \geq \left(\frac{1}{\delta}\right)^m \\
\implies &\log_{1/\delta}\phi \geq \log_{1/\delta}\left(\frac{1}{\delta}\right)^m \\
\implies &log_{1/\delta}\phi \geq m \\
\implies &m \leq \frac{\ln\phi}{\ln(1/\delta)} = O(\log(\phi(B)))
\end{aligned}$$

Swap in LLL is done at most n many times, so the total number of iterations would be

$$O(n\log(\phi(B)))$$

Result 5.6. *GSO runs in polynomial time*

Proof. Let $\mathbf{b_1^*}, \mathbf{b_2^*}, \cdots, \mathbf{b_n^*}$ be the GSO of a given basis $\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_n}$ and we express GSO w.r.t the given basis as,

$$\mathbf{b_n^*} = \mathbf{b_n} + \sum_{i=1}^{n-1} a_i\mathbf{b_i}$$

where, $a_i \in \mathbb{R}$
We also know that $\langle \mathbf{b_n^*}, \mathbf{b_i}\rangle = 0$ for all $i < n$.
Now we have,

$$\begin{aligned}
\langle \mathbf{b_n^*}, \mathbf{b_i}\rangle = &\left\langle \mathbf{b_n} + \sum_{i=1}^{n-1} a_i\mathbf{b_i}, \mathbf{bi}\right\rangle = 0 \\
\implies &\langle \mathbf{b_n}, \mathbf{b_i}\rangle + \langle a_1\mathbf{b_1}, \mathbf{b_i}\rangle + \cdots + \langle a_{n-1}\mathbf{b_{n-1}}, \mathbf{b_i}\rangle = 0 \\
\implies &a_1\langle \mathbf{b_1}, \mathbf{b_i}\rangle + \cdots + a_{n-1}\langle \mathbf{b_{n-1}}, \mathbf{b_i}\rangle = -\langle \mathbf{b_n}, \mathbf{b_i}\rangle
\end{aligned}$$

Now we have $i = 1, 2, \cdots n-1$, thus we obtain a system of equation as follows

$$\begin{aligned}
a_1\langle \mathbf{b_1}, \mathbf{b_1}\rangle + \cdots + a_{n-1}\langle \mathbf{b_{n-1}}, \mathbf{b_1}\rangle &= -\langle \mathbf{b_n}, \mathbf{b_1}\rangle \\
a_1\langle \mathbf{b_1}, \mathbf{b_2}\rangle + \cdots + a_{n-1}\langle \mathbf{b_{n-1}}, \mathbf{b_2}\rangle &= -\langle \mathbf{b_n}, \mathbf{b_2}\rangle \\
&\vdots \\
a_1\langle \mathbf{b_1}, \mathbf{b_n}\rangle + \cdots + a_{n-1}\langle \mathbf{b_{n-1}}, \mathbf{b_n}\rangle &= -\langle \mathbf{b_n}, \mathbf{b_n}\rangle
\end{aligned}$$

□

Apply Cramer's rule and solve the system of equations, which is polynomial in the size of the input.
Thus, GSO runs in polynomial time.

6 Applications

The LLL algorithm was initially framed to factorise polynomials over the rationals. Later, it came to cryptographic applications. It has numerous applications in other domains, such as optimisation, algorithm design, and number theory. LLL also gives us an algorithm to find the exact shortest vector. The algorithm, however, is of exponential cost but solves the SVP.
LLL algorithm is applicable to solve the SVP exactly in n dimensions in time $2^{O(n)}\,\text{poly}(\|B\|)$.

6.1 Solving SVP

Lemma 6.1. *Let $B = \{\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_n}\}$ be LLL reduced basis and let $\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{b_i}$ be any shortest vector in the lattice $\mathcal{L}(B)$. Then for all i, $|c_i| = 2^{O(n)}$.*

Proof. Let $B^* = \{\mathbf{b_1^*}, \mathbf{b_2^*}, \cdots, \mathbf{b_n^*}\}$ be the GSO of B.
Let $\mathbf{v}$ be a shortest vector of $\mathcal{L}$, we can express it in terms of the GSO basis as

$$\mathbf{v} = \sum_i c_i \mathbf{b_i} = \sum_i c_i^* \mathbf{b_i^*}$$

Consider the last i such that c_i is non-zero.
Then $c_i = c_i^*$ because it is the only coefficient contributing in the $\mathbf{b_i^*}$ direction.
$\|\mathbf{v}\| \leq \|\mathbf{b_1}\|$ since it is shortest vector.
and,

$$\|\mathbf{v}\| = \left\| \sum_i c_i^* \mathbf{b_i^*} \right\| \geq c_i^* \|\mathbf{b_i^*}\| \geq c_i \|\mathbf{b_i^*}\|$$

Now we have a relation $\|\mathbf{b_1}\| \geq \|\mathbf{v}\| \geq |c_i| \|\mathbf{b_i^*}\|$
From this, we obtain,

$$|c_i| \leq \frac{\|b_1\|}{\|b_i^*\|} \leq \left(\frac{2}{\sqrt{4\delta - 1}} \right)^{i-1}$$

Taking $\delta = 3/4$ we can further simplify this as

$$|c_i| \leq 2^{O(n)}$$

$\square$

Applying the above lemma, one can simply iterate over $[-2^{O(n)}, 2^{O(n)}]$ to find c_i and compute $\sum_{i=1}^{n} c_i \mathbf{b_i}$ and output the shortest among them. For exah i we have $2^{O(n)}$ choices, so we end up by having $\left(2^{O(n)}\right)^n$ trials, and this comes up as $2^{O(n^2)}$. This solves the SVP in exponential time.

7 Conclusion

We proved the correctness of the LLL algorithm, then we discussed the run-time analysis. We take this as a result that computing GSO is in polynomial time, which has been used in the LLL algorithm many times, so we end up solving the SVP in exponential time.

References

[1] Oded Regev, *Lecture Notes on Lattices in Computer Science*, Tel Aviv University, Fall 2009.

[2] Vinod Vaikuntanathan, *Advanced Topics in Cryptography: From Lattices to Program Obfuscation*, MIT, Fall 2024.

[3] Lenstra, A.K., Lenstra, H.W. & Lovász, *L. Factoring polynomials with rational coefficients. Math. Ann. 261, 515–534 (1982).* https://doi.org/10.1007/BF01457454

[4] Galbraith, S. D. *Mathematics of public key cryptography* Cambridge University Press, 2012

[5] Richard E. Blahut, *Cryptography and Secure Communication*, Cambridge University Press, April 2014