LLL Algorithm

Pranav Setpal

Purdue University

December 4, 2025

Outline

Setup

Given: A lattice $\Lambda(B) = \left\{\sum_{i=0}^{n} c_i b_i \mid c_i \in \mathbb{Z}, b_i \in B\right\}$ and a basis $B = \{b_0, b_1, \cdots, b_n\}$

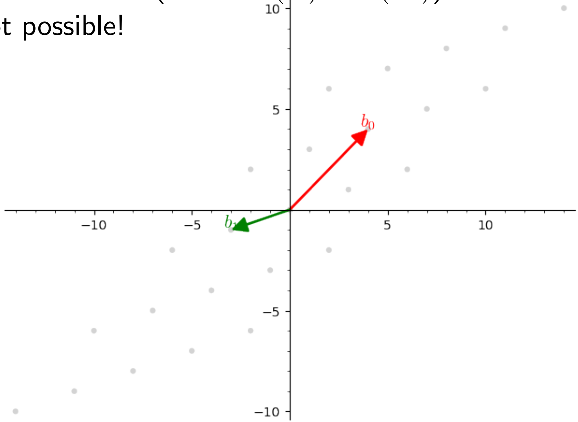
Goal: Find a basis B' with short orthogonal vectors that generate the same lattice. (that is $\Lambda(B) = \Lambda(B')$)

Setup

Given: A lattice $\Lambda(B) = \left\{\sum_{i=0}^{n} c_i b_i \mid c_i \in \mathbb{Z}, b_i \in B\right\}$ and a basis $B = \{b_0, b_1, \cdots, b_n\}$

Goal: Find a basis B' with short orthogonal vectors that generate the same lattice. (that is $\Lambda(B) = \Lambda(B')$)

This is not possible!



Setup

Given: A lattice $\Lambda(B) = \left\{\sum_{i=0}^{n} c_i b_i \mid c_i \in \mathbb{Z}, b_i \in B\right\}$ and a basis $B = \{b_0, b_1, \cdots, b_n\}$

Goal: Find a basis B' with short "nearly" orthogonal vectors that generate the same lattice. (that is $\Lambda(B) = \Lambda(B')$)

Outline

Gram-Schmidt Orthogonalization

Given a basis $B = \{b_0, b_1, \cdots, b_n\}$, we find

$$
\begin{aligned}
b_0^* &= b_0 \\
b_1^* &= b_1 - \mu_{1,0} b_0^* \\
b_2^* &= b_2 - \mu_{2,0} b_0^* - \mu_{2,1} b_1^* \\
&\vdots \\
b_i^* &= b_i - \sum_{j=0}^{i-1} \mu_{i,j} b_j^* \\
&\vdots \\
b_n^* &= b_n - \sum_{j=0}^{n-1} \mu_{i,j} b_j^*
\end{aligned}
$$

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

$B^* = \left\{ b_0^*, b_1^*, \cdots, b_n^* \right\}$ is a orthogonalized basis of B

Rounded Gram-Schmidt Orthogonalization

Given a basis $B = \{b_0, b_1, \cdots, b_n\}$, we find

$$
\begin{aligned}
b'_0 &= b_0 \\
b'_1 &= b_1 - \lfloor \mu_{1,0} \rceil b'_0 \\
b'_2 &= b_2 - \lfloor \mu_{2,0} \rceil b'_0 - \lfloor \mu_{2,1} \rceil b'_1 \\
&\vdots \\
b'_i &= b_i - \sum_{j=0}^{i-1} \lfloor \mu_{i,j} \rceil b'_j \\
&\vdots \\
b'_n &= b_n - \sum_{j=0}^{n-1} \lfloor \mu_{i,j} \rceil b'_j
\end{aligned}
$$

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

$B' = \left\{ b'_0, b'_1, \cdots, b'_n \right\}$ is a "nearly" orthogonalized basis of B

Rounded Gram-Schmidt Orthogonalization

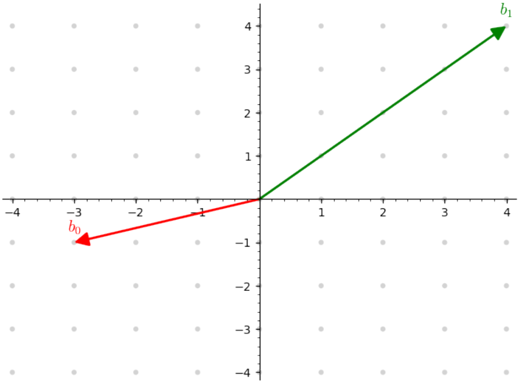
Given a basis $B = \{b_0, b_1, \cdots, b_n\}$, we find

$$
\begin{aligned}
b'_0 &= b_0 \\
b'_1 &= b_1 - \lfloor \mu_{1,0} \rceil b'_0 \\
b'_2 &= b_2 - \lfloor \mu_{2,0} \rceil b'_0 - \lfloor \mu_{2,1} \rceil b'_1 \\
&\vdots \\
b'_i &= b_i - \sum_{j=0}^{i-1} \lfloor \mu_{i,j} \rceil b'_j \\
&\vdots \\
b'_n &= b_n - \sum_{j=0}^{n-1} \lfloor \mu_{i,j} \rceil b'_j
\end{aligned}
$$

with $\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \leftarrow$ projection onto ideal orthogonalization!

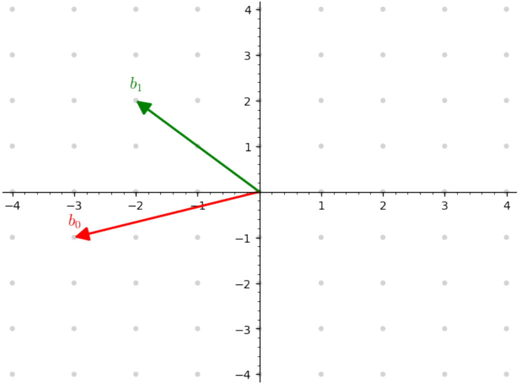
$B' = \left\{ b'_0, b'_1, \cdots, b'_n \right\}$ is a "nearly" orthogonalized basis of B

Consider



$$b_0 = \begin{bmatrix} -3 \\ -1 \end{bmatrix},\ b_1 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

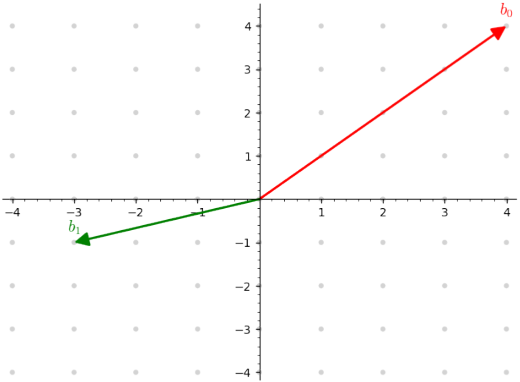
Consider



$$b'_0 = \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \; b'_1 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

The Problem

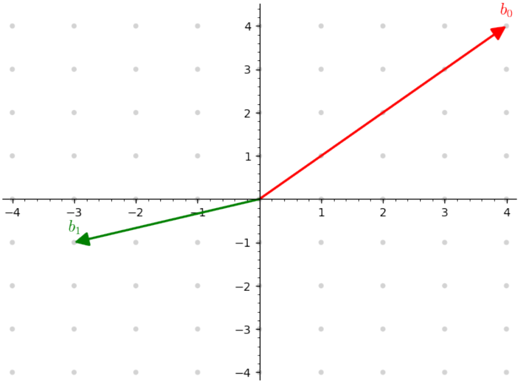
Consider



$$b_0 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \; b_1 = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

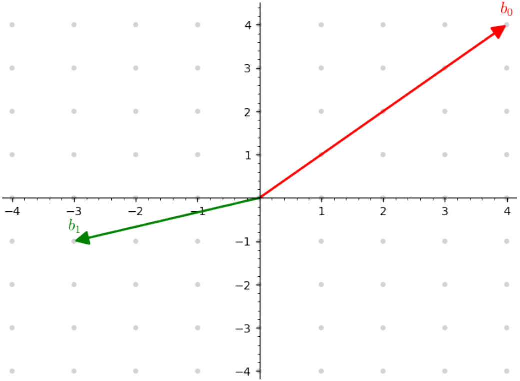
The Problem

Consider



$$b_0' = \begin{bmatrix} 4 \\ 4 \end{bmatrix},\ b_1' = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

Consider



$$b_0' = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \; b_1' = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

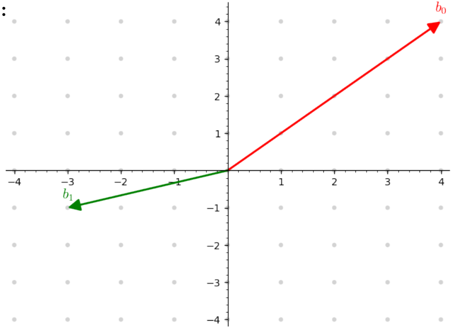
Insight: Order Matters

Langrange Gauss Reduction

```
def lagrange_gauss(L: Matrix) -> Matrix:
    assert L.nrows() == 2
    R = copy(L)

    while True:
        if R[1].norm() < R[0].norm():
            R[0], R[1] = R[1], R[0]

        mu = round((R[1] * R[0]) / R[0].norm()^2)
        if (mu == 0):
            return R
        R[1] -= mu*R[0]
```

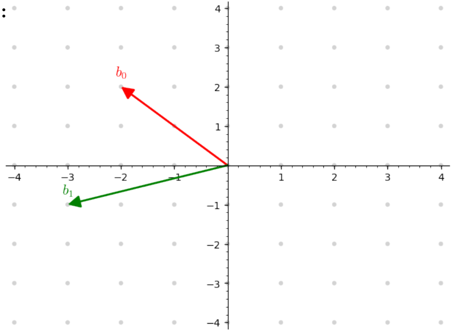


Langrange Gauss Reduction

```python
def lagrange_gauss(L: Matrix) -> Matrix:
    assert L.nrows() == 2
    R = copy(L)

    while True:
        if R[1].norm() < R[0].norm():
            R[0], R[1] = R[1], R[0]

        mu = round((R[1] * R[0]) / R[0].norm()^2)
        if (mu == 0):
            return R
        R[1] -= mu*R[0]
```



Key Idea

We know two things about our final vectors b'_0, b'_1:

- $\left\|b'_0\right\| \leq \left\|b'_1\right\|$
- $\lfloor \frac{\langle b'_0, b'_1 \rangle}{\langle b'_0, b'_0 \rangle} \rceil = 0 \implies \left| \frac{\langle b'_0, b'_1 \rangle}{\langle b'_0, b'_0 \rangle} \right| \leq \frac{1}{2}$

Using this, we can see that b'_0 and b'_1 are "nearly" orthogonal!

Key Idea

We know two things about our final vectors b_0', b_1':

- $\left\| b_0' \right\| \leq \left\| b_1' \right\|$
- $\lfloor \frac{\langle b_0', b_1' \rangle}{\langle b_0', b_0' \rangle} \rceil = 0 \implies \left| \frac{\langle b_0', b_1' \rangle}{\langle b_0', b_0' \rangle} \right| \leq \frac{1}{2}$

Using this, we can see that b_0' and b_1' are "nearly" orthogonal! Let θ denote the angle between b_0' and b_1'. Then, since

$$\cos(\theta) = \frac{\langle b_0', b_1' \rangle}{\left\| b_0' \right\| \left\| b_1' \right\|} = \frac{\langle b_0', b_1' \rangle}{\langle b_0', b_0' \rangle} \cdot \frac{\left\| b_0' \right\|}{\left\| b_1' \right\|}$$

we get that

$$\left| \cos(\theta) \right| \leq \left| \frac{\langle b_0', b_1' \rangle}{\langle b_0', b_0' \rangle} \right| \cdot \frac{\left\| b_0' \right\|}{\left\| b_1' \right\|} \leq \frac{1}{2} \cdot 1 = \frac{1}{2}$$

Hence,

$$60^\circ \leq \theta \leq 120^\circ$$

Outline

Key Ideas

1. We iteratively reduce the first k vectors, $k = 0, 1, \cdots, n$

Key Ideas

1. We iteratively reduce the first k vectors, $k = 0, 1, \cdots, n$
2. During reduction, we project each vector to the nearest point of its ideal orthogonalization. (along the line of travel)

Key Ideas

1. We iteratively reduce the first k vectors, $k = 0, 1, \cdots, n$
2. During reduction, we project each vector to the nearest point of its ideal orthogonalization. (along the line of travel)
3. During swap, we use a different equivalent criteria to stop. Notice that in the subbasis $\left\{b'_i, b'_{i+1}, \cdots, b'_n\right\}$

$$b'_i = b^*_i \text{ and } b_{i+1} = b^*_{i+1} + \mu_{i+1,i} b^*_i$$

Giving us

$$\begin{aligned} \left\|b'_i\right\|^2 \leq \left\|b'_{i+1}\right\|^2 \iff \left\|b^*_i\right\|^2 &\leq \left\|b^*_{i+1} + \mu_{i+1,i} b^*_i\right\|^2 \\ &= \left\|b^*_{i+1}\right\| + \mu^2_{i+i,i} \left\|b^*_i\right\| \\ \therefore \left\|b'_i\right\| \leq \left\|b'_{i+1}\right\| \iff (1- \ \mu^2_{i+1,i}) \left\|b^*_i\right\|^2 &\leq \left\|b^*_{i+1}\right\|^2 \end{aligned}$$

Hermite's Algorithm

```
def hermite(L: Matrix) -> Matrix:
    n = L.nrows()
    Q,_ = L.gram_schmidt()
    def proj(i, j):
        return (L[i] * Q[j]) / Q[j].norm()^2

    k = 1
    while k < n:
        # Reduction Step
        for j in range(0, k):
            mu = proj(k, j)
            if abs(mu) > 0.5:
                L[k] -= mu.round()*L[j]
                # Q,_ = L.gram_schmidt() <-- Q is unaffected

        # Conditional Swap Step
        if Q[k].norm()^2 >= (1 - proj(k, k-1)^2) * Q[k-1].norm()^2:
            k += 1
        else:
            L[k-1], L[k] = L[k], L[k-1]
            Q, _ = L.gram_schmidt()
            k = max(k-1, 1)

    return L
```

Outline

Volume of a Parellepiped

Question: How can we find the volume of a k-parallelepiped whose vectors live in n-dimensional space?

Notation: $V(\cdot)$ is the function that does this, which takes as input matrix $B = \begin{bmatrix} b_0 & b_1 & \cdots & b_k \end{bmatrix}$

If we orthogonalize our vectors, Gram-Schmidt gaurantees volume would not change. After orthogonalization, our k-parallelepiped is simply a k-rectangle, whose volume wn can compute!

$$V(B) = \prod_{i=1}^{k} \|b_i^*\|$$

Alternatively, if $k = n$, we also know the formula $V(B) = |\det(B)|$
A similar formula exists for $k \leq n$ as well.

$$V(B) = \sqrt{\det(B^\top B)}$$

Our Loss Function

Recall that each subbbasis $B_k = \begin{bmatrix} b_0 & b_1 & \cdots & b_k \end{bmatrix}$ is lattice reduced.

We define our loss function as

$$S(B) := \prod_{k=1}^{n} V(B_k) = \prod_{k=1}^{n} \prod_{i=1}^{k} \|b_i^*\|$$

Note: $S(\cdot)$ is always a positive integer

Termination of Hermite's Algorithm

Lets see how our Reduction and Swap Step affects $S(B)$

1. Reduction Step:

```
for j in range(0, k):
    mu = proj(k, j)
    if abs(mu) > 0.5:
        L[k] -= mu.round()*L[j]
        # Q,_ = L.gram_schmidt() <-- Q is unaffected
```

Since $S(\cdot)$ can be written only in terms on the Gram-Schmidt orthogonalized vectors, $S(B)$ is unaffected!

Termination of Hermite's Algorithm

Lets see how our Reduction and Swap Step affects $S(B)$

1. Reduction Step:
   Since $S(\cdot)$ can be written only in terms on the Gram-Schmidt orthogonalized vectors, $S(B)$ is unaffected!
2. Swap Step:

```
if Q[k].norm()^2 >= (1 - proj(k, k-1)^2) * Q[k-1].norm()^2:
    k += 1
else:
    L[k-1], L[k] = L[k], L[k-1]
    Q, _ = L.gram_schmidt()
    k = max(k-1, 1)
```

$V(B_j)$ is the same for all B_j except B_{k-1}

$$\frac{S(B^{\text{new}})}{S(B^{\text{old}})} = \frac{V(B^{\text{new}}_{k-1})}{V(B^{\text{old}}_{k-1})} = \frac{\left\|b^*_k + \mu_{k,k-1} b^*_{k-1}\right\|}{\left\|b^*_{k-1}\right\|} < \sqrt{1} = 1$$

Thus, $S(B^{\text{new}}) < S(B^{\text{old}})$

Termination of Hermite's Algorithm

Lets see how our Reduction and Swap Step affects $S(B)$

1. Reduction Step:
   Since $S(\cdot)$ can be written only in terms on the Gram-Schmidt orthogonalized vectors, $S(B)$ is unaffected!
2. Swap Step:
   $V(B_j)$ is the same for all B_j except B_{k-1}

$$\frac{S(B^{\text{new}})}{S(B^{\text{old}})} = \frac{V(B_{k-1}^{\text{new}})}{V(B_{k-1}^{\text{old}})} = \frac{\left\| b_k^* + \mu_{k,k-1} b_{k-1}^* \right\|}{\left\| b_{k-1}^* \right\|} < \sqrt{1} = 1$$

   Thus, $S(B^{\text{new}}) < S(B^{\text{old}})$

Termination of Hermite's Algorithm

Lets see how our Reduction and Swap Step affects $S(B)$

1. Reduction Step:
   Since $S(\cdot)$ can be written only in terms on the Gram-Schmidt orthogonalized vectors, $S(B)$ is unaffected!
2. Swap Step:
   $V(B_j)$ is the same for all B_j except B_{k-1}

$$\frac{S(B^{\text{new}})}{S(B^{\text{old}})} = \frac{V(B_{k-1}^{\text{new}})}{V(B_{k-1}^{\text{old}})} = \frac{\left\| b_k^* + \mu_{k,k-1} b_{k-1}^* \right\|}{\left\| b_{k-1}^* \right\|} < \sqrt{1} = 1$$

   Thus, $S(B^{\text{new}}) < S(B^{\text{old}})$

Since, $S(\cdot) \subseteq \mathbb{N}$, by Well Ordering, S must terminate!

Termination of Hermite's Algorithm

Lets see how our Reduction and Swap Step affects $S(B)$

1. Reduction Step:
   Since $S(\cdot)$ can be written only in terms on the Gram-Schmidt orthogonalized vectors, $S(B)$ is unaffected!
2. Swap Step:
   $V(B_j)$ is the same for all B_j except B_{k-1}

$$\frac{S(B^{\text{new}})}{S(B^{\text{old}})} = \frac{V(B_{k-1}^{\text{new}})}{V(B_{k-1}^{\text{old}})} = \frac{\left\|b_k^* + \mu_{k,k-1} b_{k-1}^*\right\|}{\left\|b_{k-1}^*\right\|} < \sqrt{1} = 1$$

   Thus, $S(B^{\text{new}}) < S(B^{\text{old}})$

Since, $S(\cdot) \subseteq \mathbb{N}$, by Well Ordering, S must terminate!
Open Problem: Proving Hermite's algorithm is polynomially bound
If we fix the dimension, then we do know it is polynomially bound!

Outline

LLL Algorithm

```python
def lll(L: Matrix, delta: float) -> Matrix:
    assert 0.25 <= delta < 1
    n = L.nrows()
    Q,_ = L.gram_schmidt()
    def proj(i, j):
        return (L[i] * Q[j]) / (Q[j] * Q[j])

    k = 1
    while k < n:
        # Reduction Step
        for j in range(0, k):
            mu = proj(k, j)
            if abs(mu) >= 0.5:
                L[k] -= mu.round()*L[j]
                # Q,_ = L.gram_schmidt() <-- Q is unaffected

        # Conditional Swap Step
        if Q[k].norm()^2 >= (delta - proj(k, k-1)^2) * Q[k-1].norm()^2:
            k += 1
        else:
            L[k-1], L[k] = L[k], L[k-1]
            Q, _ = L.gram_schmidt()
            k = max(k-1, 1)

    return L
```

Termination of LLL Algorithm

Lets see how our Reduction and Swap Step affects $S(B)$

1. Reduction Step:
   Since $S(\cdot)$ can be written only in terms on the Gram-Schmidt orthogonalized vectors, $S(B)$ is unaffected!
2. Swap Step:

```
if Q[k].norm()^2 >= (delta - proj(k, k-1)^2) * Q[k-1].norm()^2:
    k += 1
else:
    L[k-1], L[k] = L[k], L[k-1]
    Q, _ = L.gram_schmidt()
    k = max(k-1, 1)
```

$V(B_j)$ is the same for all B_j except B_k

$$\frac{S(B^{\text{new}})}{S(B^{\text{old}})} = \frac{V(B_k^{\text{new}})}{V(B_k^{\text{old}})} = \frac{\left\| b_{k+1}^* + \mu_{k+1,k} b_k^* \right\|}{\left\| b_k^* \right\|} < \sqrt{\delta}$$

Thus, $S(B^{\text{new}}) < \sqrt{\delta} S(B^{\text{old}})$

When $\delta < 1$, we are gauranteed LLL is polynomially bound!

Outline

Shortest Vector Problem

Let $\lambda(B)$ denote the shortest vector in $\Lambda(B)$

Lemma

There exists $b_i^* \in B^*$ *such that* $\left\|b_i^*\right\| \leq \left\|\lambda(B)\right\|$

Proof.

Since $\lambda(B)$ is generated by B, $\lambda(B) = \sum_{i=1}^{n} c_i b_i$ where $c_i \in \mathbb{Z}$
Let $m \in \{1, \cdots, n\}$ be the largest integer such that $c_m \neq 0$. Thus,

$$\lambda(B) = \sum_{i=1}^{m} c_i b_i = \sum_{i=1}^{m} c_i \sum_{j=1}^{i} \mu_{i,j} b_j^* = c_m b_m^* + \sum_{j=1}^{m-1} \alpha_i b_j^*$$

Thus,

$$\left\|\lambda(B)\right\|^2 = c_m^2 \left\|b_m^*\right\|^2 + \sum_{j=1}^{m-1} \left\|\alpha_i b_j^*\right\|^2 \geq c_m^2 \left\|b_m^*\right\|^2$$

Since $c_m \in \mathbb{Z}$, we have $\left\|\lambda(B)\right\|^2 \geq c_m^2 \left\|b_m^*\right\|^2 \geq \left\|b_m^*\right\|^2$
Hence, $\left\|b_m^*\right\| \leq \left\|\lambda(B)\right\|$ □

Shortest Vector Problem

Theorem

$$\|b'_0\| \leq \left(\frac{4}{4\delta - 1}\right)^{n/2} \|\lambda(B)\|$$

Proof.

Due to the ordering condition, we know

$$\|b'_0\|^2 = \|b^*_0\|^2 \leq \frac{\|b^*_1\|^2}{\delta - \mu^2_{1,0}} \leq \cdots \leq \frac{\|b^*_i\|^2}{\prod_{k=1}^{i}(\delta - \mu^2_{k,k-1})}$$

Since $1 \leq \frac{1}{\delta - \mu^2_{k,k-1}} \leq \frac{1}{\delta - \left(\frac{1}{2}\right)^2} = \frac{4}{4\delta - 1}$,

$$\|b'_0\|^2 \leq \frac{\|b^*_i\|^2}{\prod_{k=1}^{n}(\delta - \mu^2_{k,k-1})} \leq \left(\frac{4}{4\delta - 1}\right)^n \|b^*_i\|^2 \;\; \forall i \in \{1, \cdots, n\}$$

Hence, by the previous lemma, $\|b'_0\| \leq \left(\frac{4}{4\delta - 1}\right)^{n/2} \|\lambda(B)\|$ □

Thank you

Generously used facts, examples, and code from:

- *Factoring Polynomials with Rational Coefficients* by Arjen Lenstra, Hendrik Lenstra and László Lovász
- *LLL Algorithm for Lattice Basis Reduction* by Alex Kalbach, Ted Chinburg
- *The optimal LLL algorithm is still polynomial in fixed dimension* by Ali Akhavi
- Understanding the LLL Lattice Basis Reduction Algorithm by Julian D'Costa
- Building Lattice Reduction (LLL) Intuition by Kelby Ludwig

Thank you for listening to me ramble.