Scribe Notes for *Algorithmic Number Theory*
Class 24—June 19, 1998

Scribes: Wen Wang, Yizhong Wang, and Jeremy Rotter

1 Abstract

In this section, we continue to discuss the L^3 algorithm. First we will complete the proof that it terminates and we will state the time complexity. Then we will show three applications of the algorithm, namely polynomial factorization, the subset-sum problem, and Merkle-Hellman knapsack encryption.

2 Termination of the L^3 Algorithm

Define

$$\begin{aligned} d_i &= |\det(\langle b_j, b_k\rangle)_{1\le j,k\le i}| \\ &= \left|\det(\langle b_j^*, b_k^*\rangle)_{1\le j,k\le i}\right|. \end{aligned} \tag{1}$$

The equality (1) holds, because, from the previous class, we know

$$\underset{i\times n}{B_i} = \underset{i\times i}{M_i}\ \underset{i\times n}{B_i^*}$$

The j,k-entry in $B_iB_i^T$ is precisely $\langle b_j, b_k\rangle$. Also,

$$\begin{aligned} B_iB_i^T &= (M_iB_i^*)(M_iB_i^*)^T \\ &= M_i(B_i^*B_i^*)^TM_i^*. \end{aligned}$$

Therefore,

$$\begin{aligned} \det(\langle b_j, b_k\rangle) &= \det(B_iB_i^T) \\ &= \det(M_i)\det(B_i^*B_i^{*T})\det(M_i)^T \\ &= 1\cdot\det(\langle b_j^*, b_k^*\rangle)\cdot 1 \\ &= \det(\langle b_j^*, b_k^*\rangle). \end{aligned}$$

We can rewrite d_i as

$$d_i = \prod_{j=1}^{i}\|b_j^*\|^2,$$

for $0 \leq i \leq n$. Each $d_i > 0$, $d_0 = 1$, and $d_n = (d(L))^2$.

Define

$$D = \prod_{i=1}^{n-1} d_i.$$

When we swap b_{k-1} and b_k, d_{k-1} is reduced by a factor less than $\frac{3}{4}$. We need a lower bound on d_i. Define

$$m(L) = \min\left\{\|x\|^2 \ : \ x \in L - \{0\}\right\}.$$

We know that $m(L) \leq n(d(L))^{\frac{2}{n}}$. This bound is true for each i, i.e.,

$$m(L) \leq i d_i^{2/i}.$$

Hence,

$$\left(\frac{m(L)}{i}\right)^{\frac{i}{2}} \leq d_i,$$

and therefore L^3 terminates.

3 Time Complexity of the L^3 Algorithm

Let

$$B = \max_{1 \leq i \leq n} \|b_i\|^2.$$

Then L^3 requires $O(n^4 \lg B)$ arithmetic operations on integers of size $O(n \lg B)$. The bit complexity of L^3 is $O(n^6 (\lg B)^3)$.

4 Applications of the L^3 Algorithm

4.1 Factorization of Polynomials with Integer Coefficients

Theorem 4.1. *Let $f \in \mathbb{Z}[X]$ be such that the greatest common divisor[1] of its coefficients is 1. Then L^3 can be used to factor f in $O(n^{12} + n^9 (\lg f)^3)$ bit operations, where $n = \deg f$, and $\lg f$ is the number of bits necessary to represent the polynomial f.*

4.2 The Subset-Sum Problem

Definition 4.2. The Subset-Sum Problem is defined as follows:

[1]The greatest common divisor here is the greatest integer that divides each of the coefficients. This statement does not imply that the coefficients are pairwise relatively prime.

Instance: A set $S = \{a_1, a_2, \ldots, a_n\}$ of positive integers and a positive integer s.
Solution: A subset of S that sums to s. This can also be thought of as a sequence $x_1, x_2, \ldots, x_n \in \{0, 1\}$ such that

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

Theorem 4.3. *The decision version of* Subset-Sum *is NP-complete.*

Theorem 4.4. *Using dynamic programming,* Subset-Sum *can be solved in $O(ns)$ time.*

The hard case for this algorithm is when $s \gg n$.
Define the density of S as

$$density(S) = \frac{n}{\max\limits_{1 \le i \le n} \lg a_i}.$$

Now we can state the L^3 Subset-Sum algorithm:

L^3-Subset-Sum $(a_1, a_2, \ldots, a_n, s)$
$m \leftarrow \left\lceil \frac{1}{2}\sqrt{n} \right\rceil$
Let $b_1, b_2, \ldots, b_{n+1}$ be the rows of the following matrix:

$$\mathrm{B} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & ma_1 \\ 0 & 1 & \cdots & 0 & 0 & ma_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & ma_n \\ \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & \frac{1}{2} & ms \end{pmatrix}$$

$(b_1, b_2, \ldots, b_{n+1}) \leftarrow L^3(b_1, b_2, \ldots, b_{n+1})$
for $i \leftarrow 1$ to $n+1$ **do**
  Let $b_i = (Y_1, Y_2, \ldots, Y_{n+1})$
  if $Y_{n+1} = 0$ **and** $Y_i \in \{-\frac{1}{2}, \frac{1}{2}\}$, $1 \le i \le n$, **then**
    $sum \leftarrow 0$
    for $j \leftarrow 1$ to n **do**
      $X_j \leftarrow Y_j + \frac{1}{2}$
      $sum \leftarrow sum + X_j a_j$
    if $sum = s$, **then return** $(X_1, X_2, \ldots, X_n)$
    $sum \leftarrow 0$
    for $j \leftarrow 1$ to n **do**
      $X_j \leftarrow -Y_j + \frac{1}{2}$
      $sum \leftarrow sum + X_j a_j$
    if $sum = s$, **then return** $(X_1, X_2, \ldots, X_n)$
return Failure {no solution or bad luck}

If $(X_1, X_2, \ldots, X_n)$ is a solution, then

$$Y = \left(\sum_{i=1}^{n} X_i b_i \right) - b_{n+1}$$

has the form $(\pm\frac{1}{2}, \ldots, \pm\frac{1}{2}, 0)$ and has length $\|Y\| = \frac{1}{2}\sqrt{n}$.

Coster, et.al. show this algorithm will succeed with high probability when

$$density(S) < 0.9408.$$

4.3 Merkle-Hellman Knapsack Encryption

We wish to encrypt a message $m = m_1 m_2 \cdots m_n \in \{0,1\}^n$. A super-increasing sequence is a sequence $X_1, X_2, \ldots, X_n$ of positive integers such that

$$X_i > \sum_{j=1}^{i-1} X_j,$$

for $1 \leq j \leq n$.

Example 4.5. Let $m = 101101$. Then we can encode m with the super-increasing sequence 3, 5, 14, 33, 70, 197.

To encode a message m, we compute the sum

$$s = \sum X_i m_i.$$

In the example above, the encoding of m would be

$$\begin{aligned} s &= (1)3 + (0)5 + (1)14 + (1)33 + (0)70 + (1)197 \\ &= 3 + 14 + 33 + 197 \\ &= 247. \end{aligned}$$

Because we have a super-increasing sequence, decoding is easy. We start with X_n and compare it to s. If $X_n \geq s$, then $m_n = 1$. Otherwise, $m_n = 0$. Then we can simply subtract $m_n X_n$ from s, decrement n, and repeat until we have all the m_ns.

Example 4.6. Here's a decoding of the example above:

$$\begin{array}{ll} m_6 = 1 & 247 \geq 197 = X_6 \\ \quad 247 - 197 = 50 & \\ m_5 = 0 & 50 < 70 = X_5 \\ m_4 = 1 & 50 \geq 33 = X_4 \\ \quad 50 - 33 = 17 & \\ m_3 = 1 & 17 \geq 14 = m_3 \\ \quad 17 - 14 = 3 & \\ m_2 = 0 & 3 < 5 = X_2 \\ m_1 = 1 & 3 = 3 = X_1 \end{array}$$

Choose p to be a prime greater than $\sum_{i=1}^{n} X_i$. Note that, because we have a super-increasing sum, $s = s \bmod p$. Choose a random integer c between 1 and $p-1$, and compute the inverse d modulo p, that is,

$$cd \equiv 1 \pmod{p}.$$

Now, define

$$Y_i = cX_i \pmod{p}.$$

In general, the sequence $Y_1, \ldots, Y_n$ is not super-increasing. To encrypt a message m, all we must do is create the message t, where

$$t = \sum_{i=1}^{n} Y_i m_i \pmod{p}.$$

To decrypt t, all we have to do is compute $s = dt \bmod p$, and then we can just decode s as before.

We can use this scheme for public-key encryption by making the following values public and secret:

$$\begin{array}{ll} \textit{public:} & Y_1, \ldots, Y_n, p \\ \textit{secret:} & c, d, X_1, \ldots, X_n \end{array}$$

Example 4.7. Continuing with the example above, let $p = 509$ and let $c = 428$. We can compute $d = 465$. Now, we get the following values for Y_i:

$$\begin{aligned} Y_1 &= 266 \\ Y_2 &= 104 \\ Y_3 &= 393 \\ Y_4 &= 381 \\ Y_5 &= 438 \\ Y_6 &= 331. \end{aligned}$$

With these values, the encryption of m is

$$\begin{aligned} t &= (1)266 + (0)104 + (1)393 + (1)381 + (0)438 + (1)331 \bmod p \\ &= 353. \end{aligned}$$

To decrypt this message, we just compute

$$\begin{aligned} dt \bmod p &= (465)(353) \bmod p \\ &= 247. \end{aligned}$$

Definition 4.8. We can now define the Subset-Sum problem for Merkle-Hellman:

Instance: Encrypted message t, public keys consisting of a prime p and a sequence of integers $Y_1, \ldots, Y_n$, satisfying $1 \leq Y_i \leq p - 1$.

Solution: A sequence of bits $m_1, \ldots, m_n$ such that $t = \sum_{i=1}^{n} m_i Y_i$.

4.4 Simultaneous Diophantine Approximation

Definition 4.9. We define the Simultaneous Diophantine Approximation problem as

Instance: $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$, a bound $Q \in \mathbb{Z}^+$ and a real error bound ϵ.
Solution: $q, p_1, \ldots, p_d \in \mathbb{Z},\ 1 \leq q \leq Q$ such that

$$\left|\alpha_r - \frac{p_r}{q}\right| \leq \frac{\epsilon}{q}$$

for all r satisfying $1 \leq r \leq d$.

Theorem 4.10. *(Dirichlet) Simultaneous Diophantine Approximation has a solution when* $\epsilon \geq Q^{-1/d}$.

Theorem 4.11. L^3 *can solve simultaneous Diophantine Approximation in polynomial time when* $\epsilon > 2^{(d+1)/4}Q^{-1/d}$ *or, equivalently,* $\epsilon^d \geq 2^{d(d+1)/4}Q^{-1}$.

Proof. Let

$$B = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ -\alpha_1 & -\alpha_2 & \cdots & -\alpha_{d-1} & -\alpha_d & \epsilon/Q \end{pmatrix},$$

and let $b_1, \ldots, b_{d+1}$ be the reduced basis from L^3. We know

$$||b_1^*||^2 \leq 2^{i-1}||b_i^*||^2,$$

for $1 \leq i \leq d+1$.

Taking the product

$$\begin{aligned} ||b_1||^{2(d+1)} &\leq 2^{(0+1+\cdots+d)} \prod_{i=1}^{d+1} ||b_1^*||^2 \\ &= 2^{\frac{d(d+1)}{2}} (d(L))^2, \end{aligned}$$

we obtain

$$||b_1|| \leq 2^{\frac{d}{4}} (d(L))^{1/(d+1)}.$$

Now

$$\epsilon^{d+1} \geq \frac{\epsilon}{Q} 2^{d(d+1)/4},$$

so

$$d(L) = \frac{\epsilon}{Q} \leq \frac{\epsilon^{d+1}}{2^{d(d+1)/4}}$$

and

$$||b_1|| \leq \epsilon.$$

Now, we note that $b_1 = (p_1, p_2, \ldots, p_d, q)$ is an integer combination of the rows of B. Hence, we can write

$$\left|\alpha_r - \frac{p_r}{q}\right| \leq \frac{\epsilon}{q},$$

and therefore,

$$|p_r - \alpha_r q| \leq \epsilon.$$

□

5 Next Time

In the next class, we will begin to discuss primality testing and RSA Encryption.