Coppersmith Seminar

2019/10/07 (학기 중 세미나)
rbtree

Coppersmith’s Method

- Coppersmith가 처음 제안한 method가 일반적인 방법론을 제안했기 때문에 이러한 명칭으로 불림
- 미지수가 들어가 있는 정수론 방정식이 있고, 해당 식에 대해서 Partial Information을 알고 있을 때 전체를 얻어내는 강력한 기법

기본 아이디어

- 다음과 같은 RSA와 관련된 식을 생각해보자.

$$c = m^e \pmod N$$

- 우리가 알고 있는 정보가 (e, c, N)이라면, N의 소인수분해 정보를 알지 않는 한 m을 계산해내는 것은 어렵다.

기본 아이디어

- 만약 $m < N^{\frac{1}{e}}$라면?

$$c = m^e \pmod N \Leftrightarrow c = m^e$$

- 조건에 따라서, 해가 충분히 작다면 $f(x) = 0 \pmod N$ 을 푸는 것은 정수 집합 $\mathbb{Z}$ 위에서 $f(x) = 0$을 푸는 것과 같다.

  그리고 $f(x) = 0$을 푸는 것은 $f(x) = 0 \pmod N$보다 훨씬 쉽다!

조건 찾기

- 우리가 풀고 싶은 식:

$$f(x) = 0 \pmod{a}$$

- 우리가 알고 싶은 해: x_0
- 해가 충분히 작다 → $|x_0| < X$ (X는 어떤 양수)
- 어떤 조건 → $f(x_0)$???

조건 찾기

- $f(x) = 0 (\mathrm{mod}\ a)$의 의미는, 해 x에 대해 $f(x) = ak$ 를 만족하는 k가 존재한다는 뜻
- 만약 $|f(x_0)| < a$라면? → $f(x_0) = 0 (\mathrm{mod}\ a)$ 이라면 $f(x_0) = 0$ 이여야 함!
- *이것을 어떻게* $|x_0| < X$ *라는 조건과 엮어서 쓰지?*

Notations

- $f(x) := \sum_i a_i x^i \ (a_i \in \mathbb{Z})$ 인 일변수 다항식 $f(x)$가 있다고 하자.
- a_i는 **계수(coefficient)**라고 하며,

  각각의 0이 아닌 계수를 가진 항들 $(a_i x^i \ with \ a_i \neq 0)$을 **monomial**이라고 한다.
- f의 **차수(degree)**는 $\max\{i | a_i \neq 0\}$으로 정의한다.

Notations

- f의 차수가 i일 때 a_i를 f의 **leading coefficient**라고 하며, leading coefficient가 1인 f를 **monic**이라고 한다.
- f의 계수들을 vector로 나타낸 것을 **coefficient vector**라고 한다.

$$[a_0, a_1, a_2, \dots]$$

- f의 **norm**은 coefficient vector의 Euclidean (L2) norm으로 정의한다. 즉, $||f(x)||^2 \coloneqq \sum_i a_i^2$ 로 정의한다.

Howgrave-Graham Theorem

$f(x)$는 n개의 항을 가지는 일변수 다항식이라고 하자.

① $f(x_0) = 0 \pmod{a}$ where $|x_0| < X$

② $||f(xX)|| < \frac{a}{\sqrt{n}}$

위의 조건을 만족할 때, 정수 집합 $\mathbb{Z}$ 위에서 $f(x_0) = 0$ 이다.

Howgrave-Graham Theorem

Cauchy-Schwarz Inequation (껴라위키 링크):

$$\left(\sum_i a_i b_i\right)^2 \le \left(\sum_i a_i^2\right)\left(\sum_i b_i^2\right)$$

Lemma 1) $\sum_i |c_i| X^i \le \sqrt{n} ||f(xX)||$

Cauchy-Schwarz Inequation에 따라서, $a_i = |c_i| X^i, b_i = 1$로 정의하면,

$$\left(\sum_i |c_i| X^i \cdot i\right)^2 \le \left(\sum_i \left(c_i X^i\right)^2\right)\left(\sum_i 1\right) = ||f(xX)|| \cdot n$$

$\sum_i |c_i| X^i \cdot i$와 $||f(xX)||, n$ 모두 양수이므로 성립한다.

Howgrave-Graham Theorem

Proof)

$$\begin{aligned} |f(x_0)| &= \sum_i c_i x_0^i \le \sum_i \left|c_i x_0^i\right| \\ &\le \sum_i |c_i| X^i \le \sqrt{n}||f(xX)|| < a \ (\because \text{Lemma } 1, ②) \end{aligned}$$

인데, $f(x_0) = 0 \pmod{a}$ 이므로 $f(x_0)$는 a의 배수여야 한다.

그러므로 $f(x_0) = 0$이 성립한다.

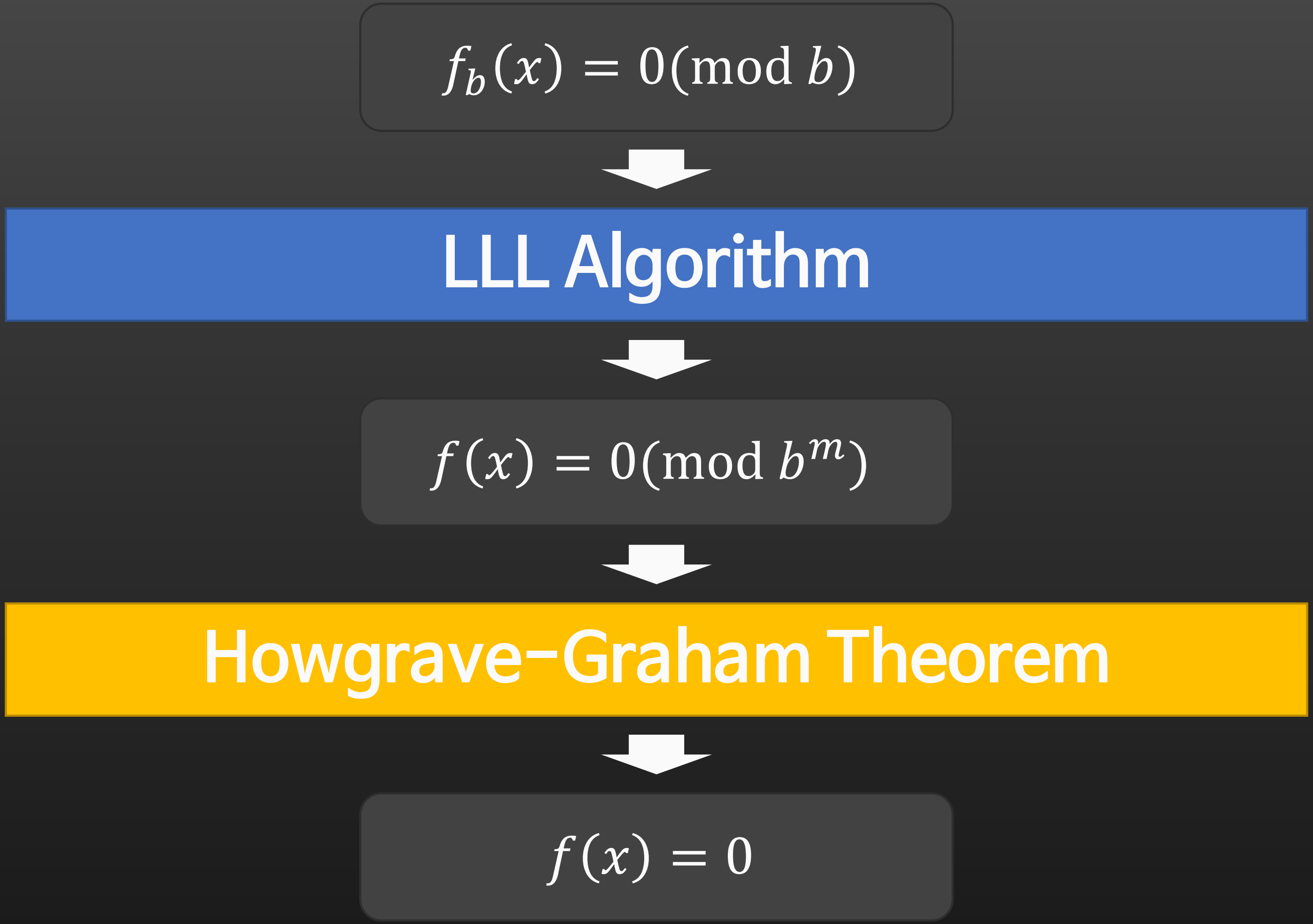
How Coppersmith's method works

1. 풀고 싶은 식 $f_b(x) = 0(\mathrm{mod}\ b)$ 가 있다.
   b가 어떤 수인지는 모르지만, N의 약수라는 사실은 알고 있다.
   또한, $f_b(x) = 0(\mathrm{mod}\ b)$ 는 작은 해인 $|x_0| < X$를 갖고 있다는 사실을 알고 있다.
2. LLL Algorithm을 통해서 $f_b(x) = 0(\mathrm{mod}\ b)$를 변환해 Howgrave-Graham Theorem의 조건을 만족하는 $f(x) = 0(\mathrm{mod}\ b^m)$로 변환한다.

How Coppersmith's method works

3. 주어진 $f(x) = 0 (\mathrm{mod}\ b^m)$는 Howgrave-Graham Theorem의 조건을 만족한다. 곧, $f(x) = 0 (\mathrm{mod}\ b^m)$의 작은 해 x_0에 대해서, $f(x_0) = 0$ over $\mathbb{Z}$가 성립한다.
4. 이제 x_0을 구하기 위해서 $f(x) = 0$을 풀면 된다.

기본 개요



$f_b(x) = 0(\mathrm{mod}\ b)$

LLL Algorithm

$f(x) = 0(\mathrm{mod}\ b^m)$

Howgrave-Graham Theorem

$f(x) = 0$

Lattices

- $v_1, v_2, \ldots, v_n \in \mathbb{Z}^m$ $(n \leq m)$ 는 서로 linearly independent
- $\{v_1, v_2, \ldots, v_n\}$ 에 의해서 span 되는 Lattice L은

$$L = \{v \in \mathbb{Z}^m | v = \sum_{i=1}^{n} a_i v_i \ \ with\ a_i \in \mathbb{Z}\}$$

- 이 때 $B = \{v_1, v_2, \ldots, v_n\}$은 L의 **Basis**라고 한다.
- $n = m$ 일 때, 해당 Lattice를 **full rank**라고 한다.
- Lattice L이 full rank일 경우, $\mathbf{det}(\boldsymbol{L})$ 은 각 row가 $v_1, v_2, \ldots, v_n$인 $(n \times n)$-matrix의 determinant이다.

Lattice Examples

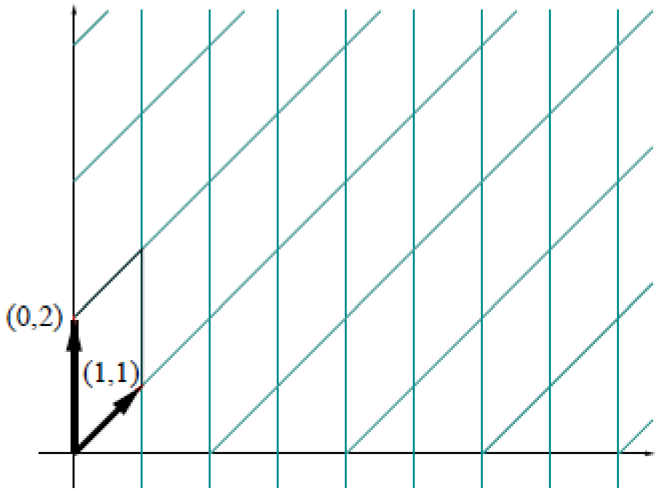
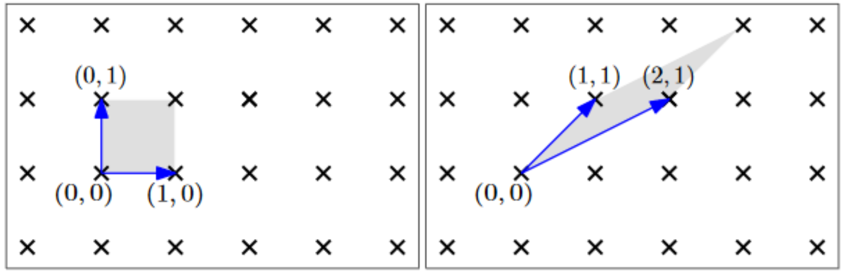


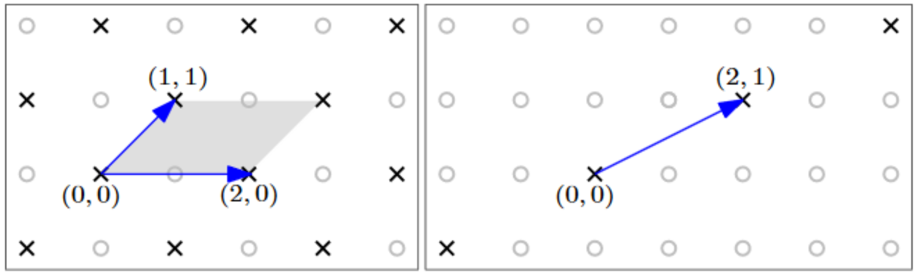
Figure 2.1: Lattice with basis $B = \{(0,2),(1,1)\}$

$$\det(L) = \det\left(\begin{bmatrix} 0 & 2 \\ 1 & 1 \end{bmatrix}\right) = 1$$



(a) A basis of $\mathbb{Z}^2$

(b) Another basis of $\mathbb{Z}^2$



(c) Not a basis of $\mathbb{Z}^2$

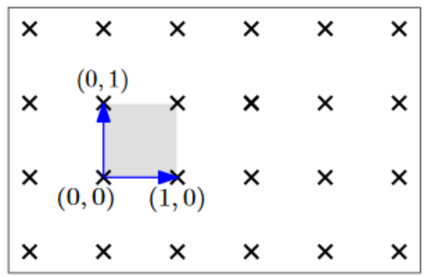
(d) Not a full-rank lattice

Figure 2: Some lattice bases

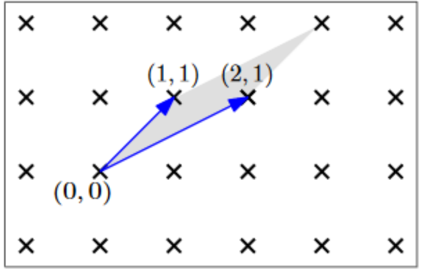
Lattice

- 한 Lattice에 대해서는 다양한 basis가 존재할 수 있음
  - 예시) $B_a = \{(0,1),(1,0)\}$과 $B_b = \{(1,1),(2,1)\}$
- 그런 Basis들 중에서 각 vector의 size가 작은 basis 가 있지 않을까? 어떻게 그걸 구하지?

  → LLL-algorithm!



(a) A basis of $\mathbb{Z}^2$



(b) Another basis of $\mathbb{Z}^2$

LLL-Algorithm

Lattice $L \in \mathbb{Z}^n$ 이 $B = \{b_1, b_2, \dots, b_n\}$ 에 의해 span 된다고 하자.

L^3-algorithm은 아래의 성질을 만족하는 **reduced lattice basis** $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_n}\}$ 을 B의 entry들의 bit size와 n에 대한 다항 시간 안에 구할 수 있다.

$$||v_i|| \leq 2^{\frac{n(n-1)}{4(n-i+1)}} \det(L)^{\frac{1}{n-i+1}}$$

(증명 생략)

LLL Algorithm

- 다음 조건을 생각해보자.

$$||v_i|| \leq 2^{\frac{n(n-1)}{4(n-i+1)}} \det(L)^{\frac{1}{n-i+1}}$$

- v_1의 norm이 가장 작고, 그 값은

$$||v_1|| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}}$$

- 즉, LLL Algorithm은 위의 조건을 만족하는 Lattice 상의 vector v를 구하는 알고리즘으로 볼 수 있다!

어떻게 쓰지?

- 우리가 원하는 것: LLL Algorithm을 통해서 $f_b(x) = 0(\mathrm{mod}\ b)$를 변환해 Howgrave-Graham Theorem의 조건을 만족하는 $f(x) = 0(\mathrm{mod}\ b^m)$로 변환한다.
- Howgrave-Graham Theorem의 조건? : $||f(xX)|| < \frac{a}{\sqrt{n}}$
  - $f(x) = 0\left(\mathrm{mod}\ b^k\right)$에 대해서는, $||f(xX)|| < \frac{b^m}{\sqrt{n}}$ 라고 할 수 있음.
  - $\mathrm{mod}\ b$인 식을 $\mathrm{mod}\ b^m$로 확장해 조건을 완화하려는 목적

LLL Algorithm to Coppersmith Method

- 풀고 싶은 식 $f_b(x) = 0 (\mathrm{mod}\ b)$ 가 있다.

  b가 어떤 수인지는 모르지만, N의 약수라는 사실은 알고 있다.

  또한, $f_b(x) = 0 (\mathrm{mod}\ b)$ 는 작은 해인 $|x_0| < X$를 갖고 있다는 사실을 알고 있다.

- 다음과 같은 $g_{i,j}(x)$를 생각해보자.

$$g_{i,j}(x) = N^{m-i} x^j f_b^i(x) \text{ for } i = 0, \ldots, m-1$$

LLL Algorithm to Coppersmith Method

- 다음과 같은 $g_{i,j}(x)$를 생각해보자.

$$g_{i,j}(x) = N^{m-i} x^j f_b^i(x) \text{ for } i = 0, \ldots, m-1$$

- $f_b(x_0) = 0 (\text{mod } b)$인 x_0에 대해, $g_{i,j}(x_0) = 0 (\text{mod } b^m)$ 이 성립한다.
- 그리고 $g_{i,j}$의 integer linear combination f를 생각해보자.

$$f(x) = \sum_{i,j} a_{i,j} g_{i,j}(x), a_{i,j} \in \mathbb{Z}$$

- $f(x_0) = 0 (\text{mod } b^m)$ 이 성립한다.

LLL Algorithm to Coppersmith Method

- 다음과 같은 $g_{i,j}(x)$를 생각해보자.

$$g_{i,j}(x) = N^{m-i}x^j f_b^i(x) \text{ for } i = 0, \dots, m-1$$

- $f_b(x_0) = 0 (\bmod b)$인 x_0에 대해, $g_{i,j}(x_0) = 0 (\bmod b^m)$ 이 성립한다.
- 그리고 $g_{i,j}$의 integer linear combination f를 생각해보자.

$$f(x) = \sum_{i,j} a_{i,j} g_{i,j}(x), a_{i,j} \in \mathbb{Z}$$

- $f(x_0) = 0 (\bmod b^m)$ 이 성립한다.

Lattice???

Coppersmith's Method

어떻게 소인수분해가 되는지 모르는 N이 있고, N의 약수 $b \geq N^{\beta}$가 있다.

$f_b(x)$는 monic인 일변수 다항식이고, 차수 δ를 가진다.

우리는 다음 조건을 만족하는 $f_b(x) = 0 (\mathrm{mod}\ b)$의 해 x_0을 $\left(\log N, \delta, \frac{1}{\epsilon}\right)$에 대한 다항 시간 안에 구할 수 있다.

$$|x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$$

Proof to Coppersmith’s Method

$$g_{i,j}(x) = N^{m-i} x^j f_b^i(x)$$

- $f_b(x)$의 차수가 δ일 때, $i = 0, \ldots, m-1$ 과 $j = 0, \ldots, \delta - 1$ 에 대해서 $g_{i,j}(xX)$의 coefficient vector들로 만든 Lattice를 생각하자. ($n = \delta m$)

Proof to Coppersmith’s Method

$$\boxed{g_{i,j}(x) = N^{m-i} x^j f_b^i(x)}$$

차수:

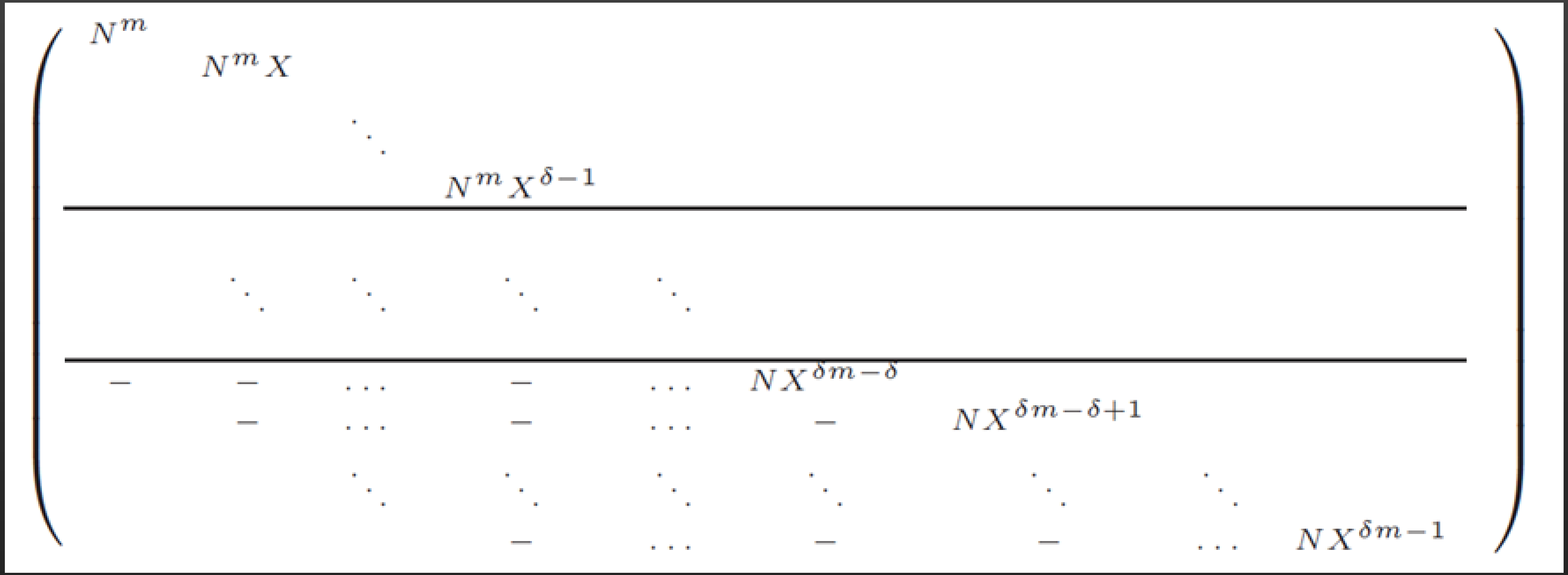
$$\delta \cdot i + j$$

$g(xX)$의 Leading Coefficient:

$$N^{m-i} \cdot X^{\delta i + j}$$

- $f_b(x)$의 차수가 δ일 때, $i = 0, \ldots, m-1$ 과 $j = 0, \ldots, \delta - 1$ 에 대해서 $g_{i,j}(xX)$의 coefficient vector들로 만든 Lattice를 생각하자. $(n = \delta m)$

Proof to Coppersmith’s Method



$$\left(\begin{array}{ccccccccc}
N^m &&&&&&&& \\
& N^m X &&&&&&& \\
&& \ddots &&&&&& \\
&&& N^m X^{\delta-1} &&&&& \\
\hline
& \ddots & \ddots & \ddots & \ddots &&&& \\
\hline
- & - & \cdots & - & \cdots & N X^{\delta m-\delta} &&& \\
& - & \cdots & - & \cdots & - & N X^{\delta m-\delta+1} && \\
&& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
&&& - & \cdots & - & - & \cdots & N X^{\delta m-1}
\end{array}\right)$$

그림판으로 대충 수정해서 발퀄입니다 죄송

Proof to Coppersmith’s Method

- 삼각행렬의 determinant는 대각선 상에 있는 값들의 곱과 같다.

$$\det(L) = \prod_{i,j} N^{m-i} \cdot X^{\delta i+j} = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}n(n-1)}$$

- 우리가 바라는 것은 LLL Algorithm으로부터 나온 v에 대해

$$||v|| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}} < \frac{N^{\beta m}}{\sqrt{n}} \leq \frac{b^m}{\sqrt{n}}$$

Proof to Coppersmith’s Method

- N은 일반적으로 Lattice의 order n보다 매우 크므로, 다음과 같이 근사할 수 있다.

$$2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}} < \frac{N^{\beta m}}{\sqrt{n}} \iff \det(L) < N^{\beta m n}$$

- 혹시 $g_{m-1,\delta-1}(x)$ 뒤로 δm차 이상의 다항식을 계속 하나씩 이어 붙여서 $\det(L)$ 을 상대적으로 $N^{\beta m n}$ 보다 더 작게 할 수 있지 않을까?

Proof to Coppersmith’s Method

$$\det(L) < N^{\beta mn}$$

- 혹시 $g_{m-1,\delta-1}(x)$ 뒤로 δm차 이상의 다항식을 계속 하나씩 이어 붙여서 $\det(L)$ 을 상대적으로 $N^{\beta mn}$ 보다 더 작게 할 수 있지 않을까?
- 다항식을 하나 추가할 때 n이 하나 늘어나므로, 새로 추가한 다항식이 $\det(L)$에 끼치는 영향이 $N^{\beta m}$ 보다 작다면, 긍정적인 영향을 줄 것이다!

Proof to Coppersmith’s Method

$$\boxed{g_{i,j}(x) = N^{m-i} x^{j} f_b^{i}(x)}$$

차수:

$\delta \cdot i + j$

$g(xX)$의 Leading Coefficient:

$N^{m-i} \cdot X^{\delta i + j}$

- $f_b(x)$의 차수가 δ일 때, $i = 0, \ldots, m-1$ 과 $j = 0, \ldots, \delta - 1$ 에 대해서 $g_{i,j}(xX)$의 coefficient vector들과,

$$h_i(x) = x^{i} f_b^{m}(x)$$

- 어떤 정수 t가 있어 $i = 0, \ldots, t-1$ 에 대해서 $h_i(xX)$의 coefficient vector들로 만든 Lattice를 생각하자. $(n = \delta m + t)$

Proof to Coppersmith's Method

$$g_{i,j}(x) = N^{m-i}x^j f_b^i(x)$$

차수:

$\delta \cdot i + j$

$g(xX)$의 Leading Coefficient:

$N^{m-i} \cdot X^{\delta i+j}$

- $f_b(x)$의 차수가 δ일 때, $i = 0, \dots, m-1$ 과 $j = 0, \dots, \delta - 1$ 에 대해서 $g_{i,j}(xX)$의 coefficient vector들과,

$$h_i(x) = x^i f_b^m(x)$$

차수:

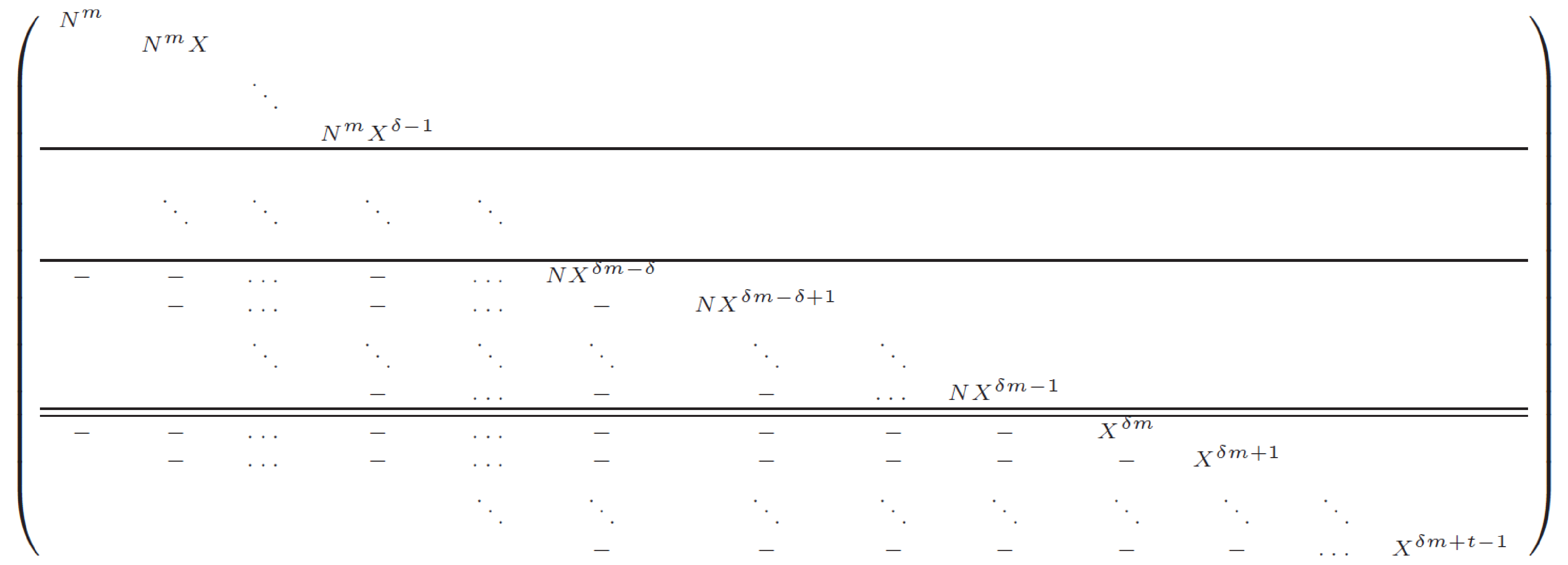
$\delta \cdot m + i$

$h(xX)$의 Leading Coefficient:

$X^{\delta m+i}$

- 어떤 정수 t가 있어 $i = 0, \dots, t-1$ 에 대해서 $h_i(xX)$의 coefficient vector들로 만든 Lattice를 생각하자. $(n = \delta m + t)$

Proof to Coppersmith’s Method



$$\left(\begin{array}{ccccccccccccc}
N^m & & & & & & & & & & & & \\
& N^m X & & & & & & & & & & & \\
& & \ddots & & & & & & & & & & \\
& & & N^m X^{\delta-1} & & & & & & & & & \\
\hline
& \ddots & \ddots & \ddots & \ddots & & & & & & & & \\
\hline
- & - & \dots & - & \dots & N X^{\delta m-\delta} & & & & & & & \\
& - & \dots & - & \dots & - & N X^{\delta m-\delta+1} & & & & & & \\
& & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & \\
& & & - & \dots & - & - & \dots & N X^{\delta m-1} & & & & \\
\hline\hline
- & - & \dots & - & \dots & - & - & - & - & X^{\delta m} & & & \\
& - & \dots & - & \dots & - & - & - & - & - & X^{\delta m+1} & & \\
& & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
& & & & & - & - & - & - & - & - & \dots & X^{\delta m+t-1}
\end{array}\right)$$

Proof to Coppersmith’s Method

- 삼각행렬의 determinant는 대각선 상에 있는 값들의 곱과 같다.

$$\det(L) = \left(\prod_{i,j} N^{m-i} \cdot X^{\delta i+j}\right)\left(\prod_{i} X^{\delta m+i}\right) = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}n(n-1)}$$

- $h_i(x)$가 $\det(L)$을 $X^{\delta m+i}$만큼 늘리고 있고 우리는 이 증가폭이 $N^{\beta m}$보다 작기를 바란다. 즉,

$$X^{n-1} < b^m$$

Proof of Coppersmith's Method

- Coppersmith는 $X = \frac{1}{2}N^{\frac{\beta^2}{\delta}-\epsilon}$로 정하고, m은 $m = \max\left\{\frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta}\right\}$로 잡았다.
- 앞서 $h_i(x)$를 덧붙이기 위한 조건 $X^{n-1} < b^m$를 전개하면,

$$X^{n-1} < N^{\left(\frac{\beta^2}{\delta}-\epsilon\right)(n-1)} < N^{\frac{\beta^2}{\delta}n}$$

- 이고, $b \geq N^{\beta}$이므로 $n = \frac{\delta}{\beta}m$으로 잡으면 $X^{n-1} < b^m$을 만족한다.

Proof of Coppersmith’s Method

- $n = \frac{\delta}{\beta} m$이고, $m = \max\left\{\frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta}\right\}$ 이므로 $n = \max\left\{\frac{\beta}{\epsilon}, 7\right\}$이다.
- 앞서 이야기했던 Lattice의 Basis B의 entry들의 bit size를 생각해보면, $(\delta + m)\log N \le (\delta + n)\log N$에 bound된다.
- 즉, 우리가 만든 Lattice에 대한 LLL Algorithm의 소요 시간은 **$\log N, \delta, \frac{1}{\epsilon}$의 다항 시간에 동작한다.**

Proof of Coppersmith's Method

- 앞서 $\det(L)$ 이 $N^{\frac{1}{2}\delta m(m+1)}X^{\frac{1}{2}n(n-1)}$ 라는 것을 구했다.
- 우리가 바라는 것은 $2^{\frac{n-1}{4}}\det(L)^{\frac{1}{n}} < \frac{b^m}{\sqrt{n}}$이다.
- 이를 합쳐서 정리하면,

$$N^{\frac{\delta m(m+1)}{2n}}X^{\frac{n-1}{2}} \leq 2^{-\frac{n-1}{4}}n^{-\frac{1}{2}}N^{\beta m}$$

$$X \leq 2^{-\frac{1}{2}}n^{-\frac{1}{n-1}}N^{\frac{2\beta m}{n-1}-\frac{\delta m(m+1)}{n(n-1)}}$$

Proof of Coppersmith’s Method

- 앞서 $n = \max\left\{\frac{\beta}{\epsilon}, 7\right\}$으로 잡았었다. 즉 $n \geq 7$인데, 이 경우

$$n^{-\frac{1}{n-1}} = 2^{-\frac{\log n}{n-1}} \geq 2^{-\frac{1}{2}}$$

가 성립한다.

- 즉,

$$X \leq \frac{1}{2} N^{\frac{2\beta m}{n-1} - \frac{\delta m(m+1)}{n(n-1)}}$$

이면 충분하다.

Proof of Coppersmith’s Method

- $X = \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$ 로 정했으므로, 앞서 나온 식을 다음과 같이 쓸 수 있다.

$$\frac{2\beta m}{n-1} - \frac{\delta m^2 \left(1 + \frac{1}{m}\right)}{n(n-1)} \geq \frac{\beta^2}{\delta} - \epsilon$$

- 좌변에 $\frac{n-1}{n}$을 곱하고, $n = \frac{\delta}{\beta} m$을 대입하면

$$2\frac{\beta^2}{\delta} - \frac{\beta^2}{\delta}\left(1 + \frac{1}{m}\right) \geq \frac{\beta^2}{\delta} - \epsilon$$

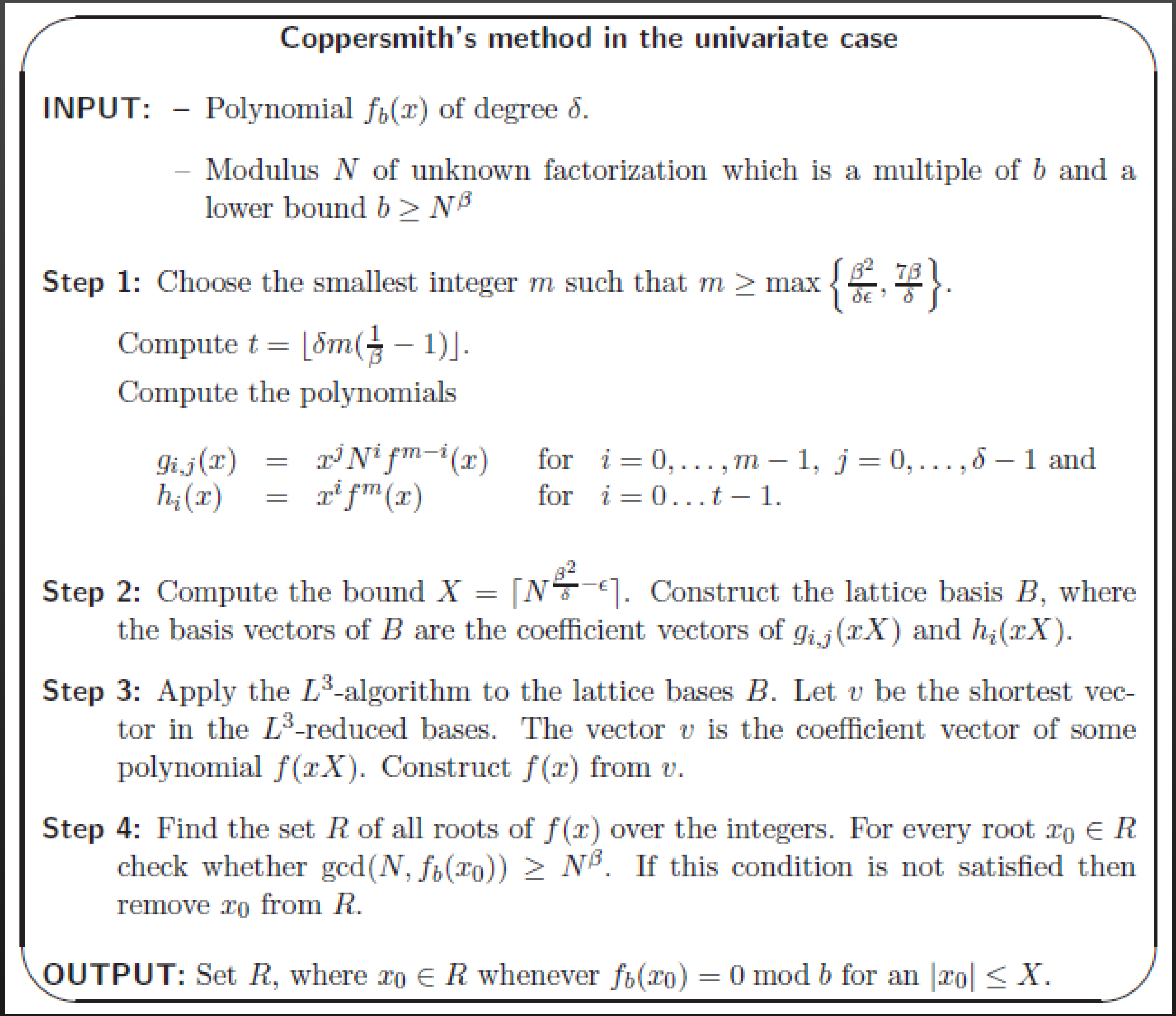
Proof of Coppersmith’s Method

- 이를 정리하면,

$$-\frac{\beta^2}{\delta}\cdot\frac{1}{m}\geq-\epsilon \iff m\geq\frac{\beta^2}{\delta\epsilon}$$

- $m=\max\left\{\frac{\beta^2}{\delta\epsilon},\frac{7\beta}{\delta}\right\}$이므로 조건을 만족한다. 즉, LLL algorithm이 구하는 v에 따라 생성한 다항식 $f(x)$는 $f(x_0)=0$ 을 만족한다.

요약



Coppersmith's method in the univariate case

INPUT:
- Polynomial $f_b(x)$ of degree δ.
- Modulus N of unknown factorization which is a multiple of b and a lower bound $b \geq N^{\beta}$

Step 1: Choose the smallest integer m such that $m \geq \max\left\{\frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta}\right\}$.

Compute $t = \lfloor \delta m(\frac{1}{\beta} - 1) \rfloor$.

Compute the polynomials

$$\begin{array}{rcll} g_{i,j}(x) & = & x^j N^i f^{m-i}(x) & \text{for} \quad i = 0, \ldots, m-1, \; j = 0, \ldots, \delta - 1 \text{ and} \\ h_i(x) & = & x^i f^m(x) & \text{for} \quad i = 0 \ldots t-1. \end{array}$$

Step 2: Compute the bound $X = \lceil N^{\frac{\beta^2}{\delta} - \epsilon} \rceil$. Construct the lattice basis B, where the basis vectors of B are the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$.

Step 3: Apply the L^3-algorithm to the lattice bases B. Let v be the shortest vector in the L^3-reduced bases. The vector v is the coefficient vector of some polynomial $f(xX)$. Construct $f(x)$ from v.

Step 4: Find the set R of all roots of $f(x)$ over the integers. For every root $x_0 \in R$ check whether $\gcd(N, f_b(x_0)) \geq N^{\beta}$. If this condition is not satisfied then remove x_0 from R.

OUTPUT: Set R, where $x_0 \in R$ whenever $f_b(x_0) = 0 \bmod b$ for an $|x_0| \leq X$.

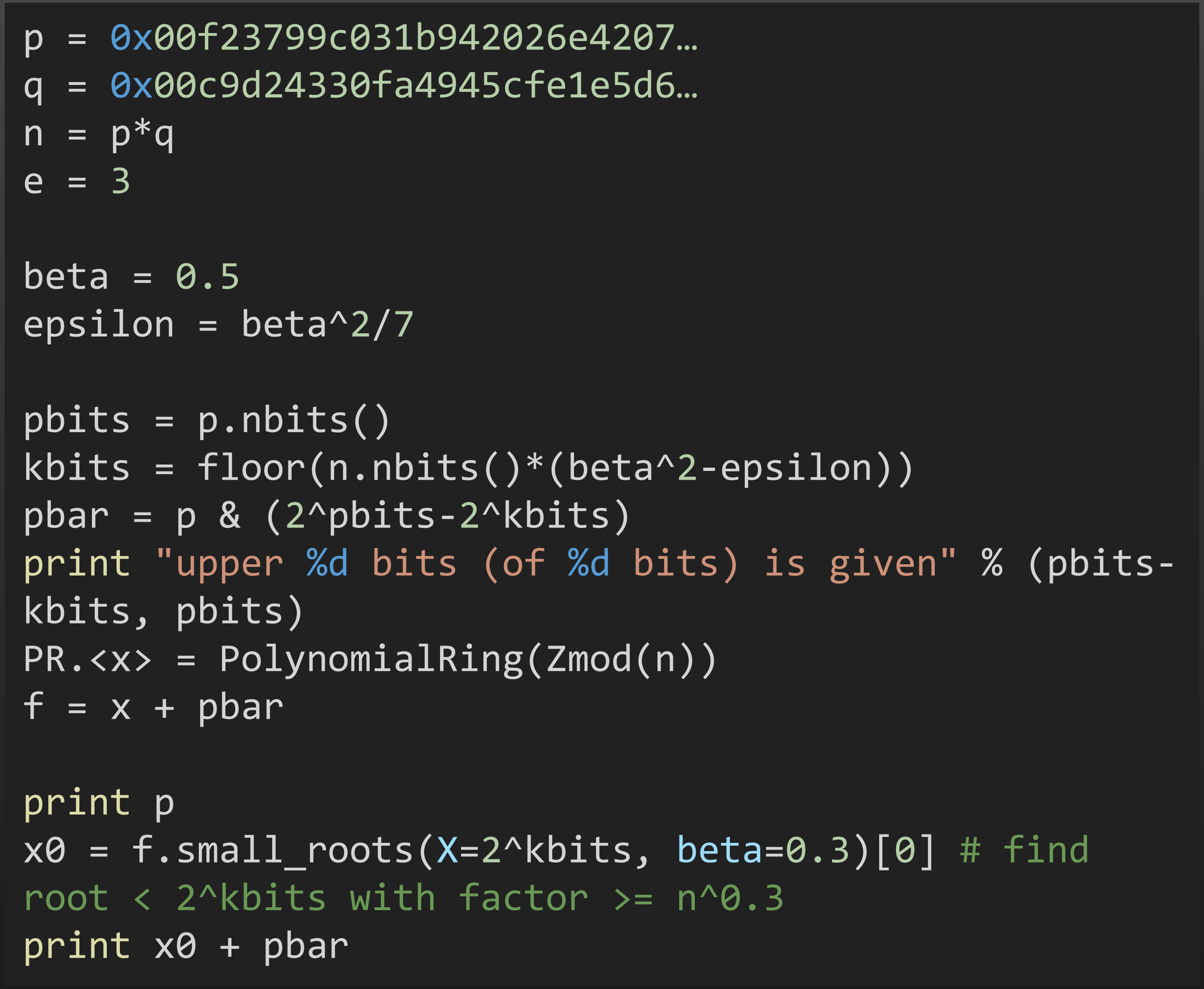
Sage에서 사용하기

- Univariate case 에 대해서는 small_roots function으로 정의되어있음.
  `def small_roots(self, X=None, beta=1.0, epsilon=None, **kwds):`
  https://github.com/sagemath/sage/blob/e8633b09919542a65e7e990c8369fee30c7edefd/src/sage/rings/polynomial/polynomial_modn_dense_ntl.pyx#L400
- 들어가서 구현 살펴보기

Sage에서 사용하기

- http://inaz2.hatenablog.com/entry/2016/01/20/022936 라는 블로그 글이 있음
- 일본어를 몰라도 대충 한자를 읽고 코드를 복붙하기 좋은 사이트임
- 그러나…

코드의 문제점



```
p = 0x00f23799c031b942026e4207…
q = 0x00c9d24330fa4945cfe1e5d6…
n = p*q
e = 3

beta = 0.5
epsilon = beta^2/7

pbits = p.nbits()
kbits = floor(n.nbits()*(beta^2-epsilon))
pbar = p & (2^pbits-2^kbits)
print "upper %d bits (of %d bits) is given" % (pbits-kbits, pbits)
PR.<x> = PolynomialRing(Zmod(n))
f = x + pbar

print p
x0 = f.small_roots(X=2^kbits, beta=0.3)[0] # find root < 2^kbits with factor >= n^0.3
print x0 + pbar
```

코드의 문제점

- kbits를 구할 때의 조건:

  $$\beta = 0.5, \epsilon = \beta^2/7$$

  - 해당 코드는 $p > q$이기 때문에 $p \geq N^\beta$가 성립.
  - 만약 $p < q$라면 어쩌려고?
- small_roots에 넣은 조건:

  $$X = 2^{kbits}, \beta = 0.3$$

  - 도대체 왜 kbits를 만들 때의 조건과 다른 β를 넣는가?
  - Sage에서 ϵ의 기본 값은 $\frac{\beta^2}{8}$

```
p = 0x00f23799c031b942026e4207…
q = 0x00c9d24330fa4945cfe1e5d6…
n = p*q
e = 3

beta = 0.5
epsilon = beta^2/7

pbits = p.nbits()
kbits = floor(n.nbits()*(beta^2-epsilon))
pbar = p & (2^pbits-2^kbits)
print "upper %d bits (of %d bits) is given" % (pbits-
kbits, pbits)
PR.<x> = PolynomialRing(Zmod(n))
f = x + pbar

print p
x0 = f.small_roots(X=2^kbits, beta=0.3)[0] # find
root < 2^kbits with factor >= n^0.3
print x0 + pbar
```

옳게 된 코드

```
p = 0x00f23799c031b942026e4207…
q = 0x00c9d24330fa4945cfe1e5d6…
n = p*q
e = 3

beta = 0.4
epsilon = beta^2/7

pbits = p.nbits()
kbits = floor(n.nbits()*(beta^2-epsilon))
pbar = p & (2^pbits-2^kbits)
print "upper %d bits (of %d bits) is given" % (pbits-
kbits, pbits)
PR.<x> = PolynomialRing(Zmod(n))
f = x + pbar

print p
x0 = f.small_roots(beta=beta, epsilon=epsilon)[0]
print x0 + pbar
```

Sage에서 사용할 때 주의점

- 수많은 CTF solver 코드들이 small_roots의 인자로 X랑 β를 넣고 있음
- 하지만 이러면 small_roots가 β에 따라 알아서 ϵ 값을 정할 뿐더러, X가 $\frac{1}{2}N^{\frac{\beta^2}{\delta}-\epsilon}$ 보다 더 큰 경우가 허다함
- X가 무엇인지 정의하고 β값을 적당히 때려박는 것이 아니라, $X = \frac{1}{2}N^{\frac{\beta^2}{\delta}-\epsilon}$ 에 따라서 적절한 δ와 ϵ을 잡는 것이 **수학적으로 무조건 답이 나오는 solver**를 만드는 지름길

기본적인 방법들

- p의 상위 비트 $\tilde{p}$를 알고 있는 경우:

$$x + \tilde{p} = 0 \pmod{p}$$

- p의 하위 비트 $\tilde{p}$를 알고 있는 경우:

$$2^k x + \tilde{p} = 0 \pmod{p}$$

- m의 상위 비트 $\widetilde{m}$을 알고 있는 경우:

$$(x + \widetilde{m})^e - c = 0(\mathrm{mod}\ N)$$

- m의 하위 비트 케이스는 p 랑 동일하게 $2^k x$로 치환

기본적인 방법들

- d의 하위 l 비트 $\tilde{d}$를 알고 있는 경우:

$$ed = k(N - p - q + 1) + 1$$

$$= k\left(N - p - \frac{N}{p} + 1\right) + 1$$

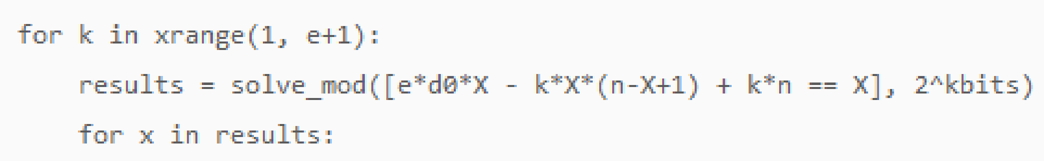
- $k \leq e$이므로, $k = 0 \ldots e$에 대해서 순회하면서

$$kp^2 + \left(e\tilde{d} - kN - k - 1\right)p + kN = 0 \pmod{2^l}$$

- 을 풀어 $p \bmod 2^l$을 구함.

기본적인 방법들

$$kp^2 + \left(e\tilde{d} - kN - k - 1\right)p + kN = 0 (\mathrm{mod}\ 2^l)$$



```
for k in xrange(1, e+1):
    results = solve_mod([e*d0*X - k*X*(n-X+1) + k*n == X], 2^kbits)
    for x in results:
```

- 을 풀어 $p \bmod 2^l$을 구함.
- 이 때, Sage의 solve_mod는 느릴 수 있음. (블로그 글에서 사용) $p \bmod 2^i$의 후보를 구하면, 후보들 앞에 0 또는 1을 붙여 식에 대입해 성립하는지 확인해 $p \bmod 2^{i+1}$를 구할 수 있음. 이런 식으로 iterative하게 $p \bmod 2$부터 시작해 $p \bmod 2^l$을 구할 수 있음.
- 이 이후는 p의 하위 비트 $\tilde{p}$를 알고 있는 경우와 동일.

TWCTF 2019 – Happy!

- 이것을 못 푼 것은 천추의 한
- 문제에서 주어진 것: $N = pq^2, e,$ 그리고 $c_f = p^{-1} \bmod q^2$
- $c_f p - 1 = 0 \pmod{q^2}$ 이니 이를 Coppersmith's method에 넣고 p를 구하는 상상을 해볼 수 있음.
  p와 q의 동일하게 765비트로 주어진 문제이기 때문에

$$N^{\beta} \le q^2 = N^{\frac{\log q^2}{\log N}}$$

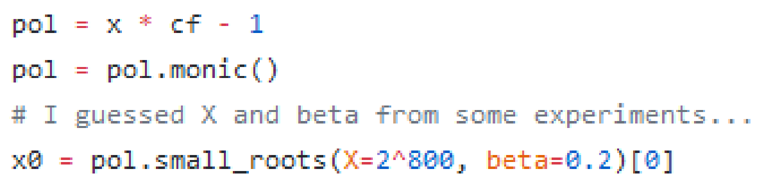
$$\frac{\log q^2}{\log N} = \frac{2 \log q}{\log N} \ge \frac{2 \cdot 764}{765 \cdot 3} \approx 0.6657952069716776$$

TWCTF 2019 – Happy!

- $\beta = 0.66$으로 설정하고, ϵ은 Sage 기본 설정으로 두어 X의 크기를 구해보면

$$X = \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon} = N^{0.66^2\left(\frac{1}{1} - \frac{1}{8}\right) - \frac{1}{\log_2 N}} \approx 0.3807142701525055$$

- p는 당연히 X 범위 안에 들어감
- small_roots(beta=0.65) 를 실행하면 당연히 나와야 함
- 그런데 Solver는?
  - https://gist.github.com/elliptic-shiho/3ea53bc0829bd1fb37236cb35e73d532
  - 제발 이러지 말자



```
pol = x * cf - 1
pol = pol.monic()
# I guessed X and beta from some experiments...
x0 = pol.small_roots(X=2^800, beta=0.2)[0]
```

다음으로 공부해 볼 것은?

- Boneh-Durfee Attack ($d \leq N^{0.292}$)
  - https://github.com/mimoo/RSA-and-LLL-attacks
  - ebmoon의 블로그를 읽으면 조금 더 도움이 될 수도? https://eyebrowmoon.github.io/hacking/crypto/rsa/2019/05/23/RSA_Attack_Using_LLL.html
  - 그 외의 Bivariate case에 대해서도 고민해볼 수 있음
- 더 나아가서 Alexander May의 글도 읽어보길 권장.
  - https://www.math.uni-frankfurt.de/~dmst/teaching/WS2015/Vorlesung/Alex.May.pdf
- Nadia Heninger의 PPT에는 Coppersmith method로 공격 가능한 모든 방법들에 대해서 간략하게 소개하고 있음
  - https://www.kangacrypt.info/files/NH.pdf

참고 문헌

- APA 스타일로 정리하기 귀찮습니다
- Alexander May May (2003), New RSA Vulnerabilities Using Lattice Reduction Methods, https://www.math.uni-frankfurt.de/~dmst/teaching/WS2015/Vorlesung/Alex.May.pdf
- https://github.com/sagemath/sage/blob/e8633b09919542a65e7e990c8369fee30c7edefd/src/sage/rings/polynomial/polynomial_modn_dense_ntl.pyx
- http://inaz2.hatenablog.com/entry/2016/01/20/022936

END