Multiplication 비교

https://youtu.be/yxvtwG-R_Qs

정보컴퓨터공학과 송경주

Karatsuba Multiplication

- 큰 수의 곱셈을 빠르게 수행할 수 있음
- $O(n^2)$ → $O(n^{1.585})$
- Divide and conquer 방식을 사용함

1. X, Y에 대해 절반으로 나눔

$$X = a \cdot 10^m + b \quad Y = c \cdot 10^m + d$$

1. 나눈 값에 대해 분할 곱셈 수행

$$X \cdot Y = (a \cdot 10^m + b)(c \cdot 10^m + d) = ac \cdot 10^{2m} + (ad + bc) \cdot 10^m + bd$$

$$(ad + bc) = (a + b)(c + d) - ac - bd$$

곱셈 두 번 곱셈 한 번

Montgomery Multiplication

- 모듈로 곱셈을 빠르게 수행할 수 있음
- 나눗셈을 곱셈 및 덧셈으로 대체하여 사용
- 특정 전처리 필요 (Montgomery form)

1. 입력에 A, B 대해 Montgomery form 변환 (R은 N보다 큰 거듭제곱 ex. N=7, R = 8)

   $\tilde{A} = A \cdot R \mod N \qquad \tilde{B} = B \cdot R \mod N$

2. 곱셈 수행

   $T = \tilde{A} \cdot \tilde{B}$

3. 몽고메리 리덕션(REDC) 수행
   - $R^{-1} \mod N$: R의 모듈러 역원
   - N': $-N^{-1} \mod R$

   $$\mathrm{REDC}(T) = \frac{T + (T \cdot N' \mod R) \cdot N}{R} = T \cdot R^{-1} \mod N$$

4. 결과에 대해 normal form 변환

   $C = \mathrm{REDC}(\tilde{C}) = \tilde{C} \cdot R^{-1} \mod N$

Montgomery in Cryptography

- 암호에서는 모듈러 연산이 매우 많아 Montgomery multiplication이 많이 사용됨
- Montgomery in RSA

$$C = M^d \mod N$$

M: 평문 (메시지), d: 개인키 (private exponent), $N = p * q$ (두 소수의 곱)

- Montgomery form 변환 $M' = M \cdot R \mod N$
- 그 결과 Montgomery form에서의 제곱이 수행됨 $C' = M'^d \mod N$

1. 초기값 설정 (값을 누적하는 용도)

   $C = 1$(여기에 값을 쌓음), $* C' = R$ mod N (몽고메리 형식에서 값을 쌓을 곳)

2. Square-and-Multiply 알고리즘 수행

- d(제곱 값)를 이진수로 변환하여 한 비트씩 처리
- 제곱(Square)연산 수행 (항상): $C' = MontMul(C', C')$
- 곱셈(Multiply)연산 수행 (비트가 1일 때): $C' = MontMul(C', M')$

Chinese Remainder Theorem (CRT)

- 모듈러 연산(덧셈, 뺄셈, 곱셈, 거듭제곱)에 효율적임
- 병렬처리가 가능함
- 큰 수를 작은 모듈러 값으로 분해하고 각 모듈러 연산 후 다시 합침

Ex) X가 서로소인 n개($m_1, m_2, \ldots, m_n$)의 모듈러 값에 대해 각각 다른 나머지를 가질 때, 이 값을 통해 X를 다시 복원할 수 있음 (즉, 작은 모듈러 값들에 대한 나머지 정보로 원래 값을 복원)

$$X \equiv a_1 \mod m_1 \qquad X \equiv a_2 \mod m_2 \qquad X \equiv a_3 \mod m_3$$

1. 전체 모듈러스 $M = m_1 \cdot m_2 \cdot m_3$ 일 때, 각 모듈러에 대해 partial modulus M_1을 구할 수 있음

$$M_i = \frac{M}{m_i} = \frac{m_1 m_2 m_3}{m_i} \qquad M_1 = \frac{M}{m_1}, \quad M_2 = \frac{M}{m_2}, \quad M_3 = \frac{M}{m_3}$$

2. 각 M_i에 대해 모듈러 역원을 구함 $M_i^{-1} \equiv M_i^{-1} \mod m_i$

3. 다음과 같이 최종 X가 구해짐 $X = \sum_{i=1}^{k} a_i \cdot M_i \cdot M_i^{-1} \mod M$

CRT in Cryptography

- CRT in RSA
  - 거듭제곱의 속도를 약 4배 가속화함
  - RSA 복호화 과정에서 다음과 같은 거듭제곱을 수행

    $$C^d \mod N$$ (C: 암호문, d: RSA 개인키, $N = p * q$)

  - C^d를 두개의 작은 모듈러 연산으로 변환 C^{dp}, C^{dq}

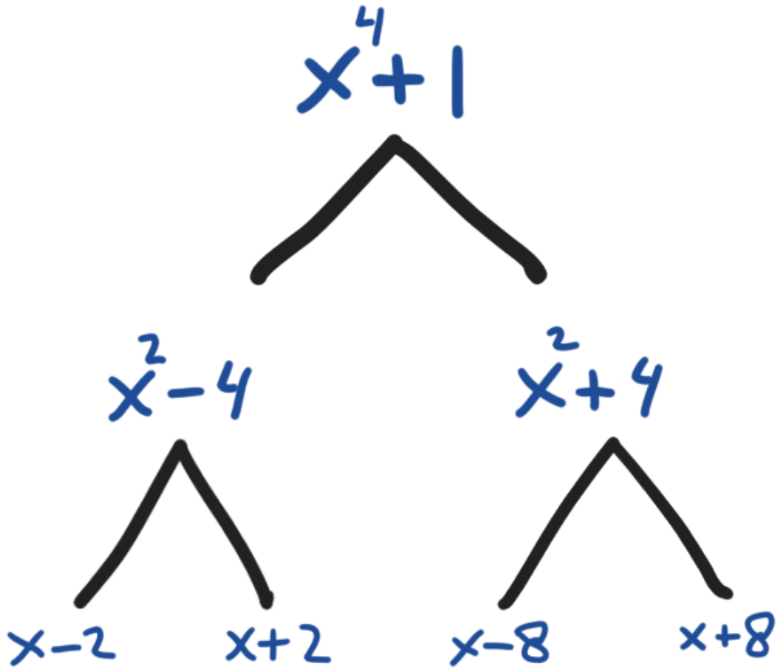
    $$m_1 = C^{d_p} \mod p$$
    $$m_2 = C^{d_q} \mod q$$
    ($d_p = d$ mod (p – 1), $d_q = d$ mod (q – 1))

  - 두개의 모듈러 연산 결과 m_1, m_2을 사용하여 원래 값 복원
    1. 전체 모듈러스 M계산 $M = p \times q$
    2. 각각의 Partial Modulus M_i 계산 (즉, $M_1 = q,\ M_1 = p$) $M_1 = \frac{M}{p} = q, \quad M_2 = \frac{M}{q} = p$
    3. 각 M_i의 모듈러 역원 계산 $M_1^{-1} = q^{-1} \mod p, \quad M_2^{-1} = p^{-1} \mod q$
    4. 최종적 X 계산 $X = (m_1 \cdot M_1 \cdot M_1^{-1}) + (m_2 \cdot M_2 \cdot M_2^{-1}) \mod N$

Number Theoretic Transform (NTT)

- 다항식 곱셈을 빠르게 수행
- $O(n^2) \rightarrow O(n\log n)$
- 다항식을 작은 차수의 다항식으로 분해하여 Point-Value Representation 으로 표현

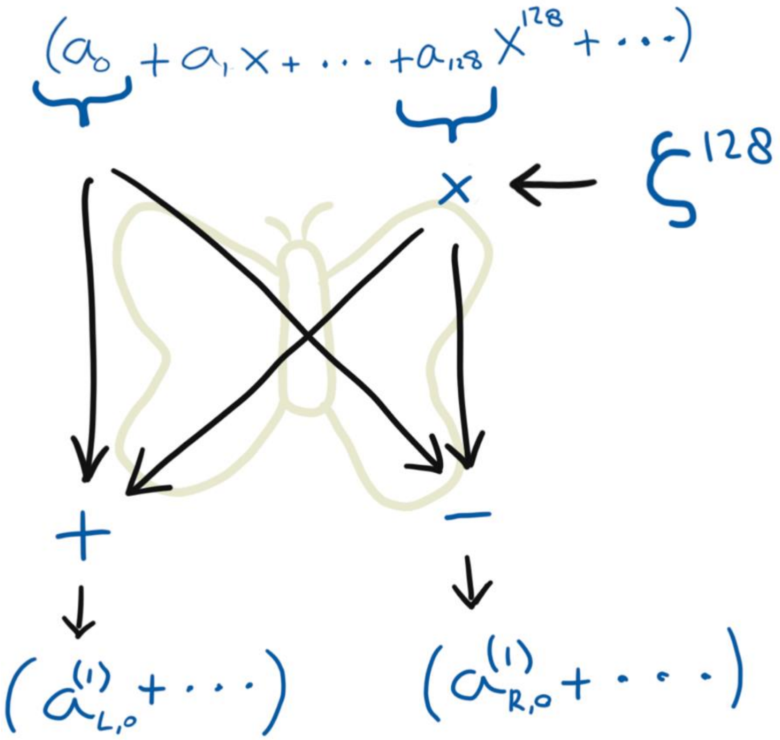
  Ex. $x^4 + 1$ 이 작은 차수의 다항식으로 분해됨 $(x^2-4)(x^2+4) = (x-2)(x+2)(x-8)(x+8)$



Number Theoretic Transform (NTT)

- 다항식의 계수들이 두 그룹으로 나뉨

$$\mathbf{a} \in \mathbb{Z}_q[X]/(X^{256}+1) \longrightarrow \quad \mathbf{a}_L^{(1)} \in \mathbb{Z}_q[X]/(X^{128}-\zeta^{128}) \quad \mathbf{a}_R^{(1)} \in \mathbb{Z}_q[X]/(X^{128}+\zeta^{128})$$



$$\mathbf{a}_L^{(1)} = (a_0 + \zeta^{128}a_{128}) + (a_1 + \zeta^{128}a_{129})X + (a_2 + \zeta^{128}a_{130})X^2 + \dots$$

$$\mathbf{a}_R^{(1)} = (a_0 - \zeta^{128}a_{128}) + (a_1 - \zeta^{128}a_{129})X + (a_2 - \zeta^{128}a_{130})X^2 + \dots$$

Number Theoretic Transform (NTT)

$$A(X) = a_0 + a_1 X + a_2 X^2 + \dots + a_{255} X^{255}$$

$$B(X) = b_0 + b_1 X + b_2 X^2 + \dots + b_{255} X^{255}$$

NTT

$$\tilde{A} = (\tilde{A}_0, \tilde{A}_1, \dots, \tilde{A}_{255})$$

$$\tilde{B} = (\tilde{B}_0, \tilde{B}_1, \dots, \tilde{B}_{255})$$

Pointwise Multiplication $\tilde{C}_i = \tilde{A}_i \cdot \tilde{B}_i \mod p$

$$\tilde{C} = (\tilde{C}_0, \tilde{C}_1, \dots, \tilde{C}_{255})$$

Inverse NTT $C(X) = \mathrm{iNTT}(\tilde{C})$

$$C(X) = A(X) \cdot B(X) \mod p$$

NTT in Cryptography

- NTT in Lattice-based Cryptography (Kyber, Dilithium)
  - Kyber
  - Key Generation: 공개키 및 비밀키를 생성하는 과정에서 격자기반 다항식 곱셈 수행
  - Encryption: 다항식 연산을 통한 암호문 생성 $(a \cdot s + e) \mod q$
  - Decryption: 복호화 과정에서의 다항식 연산 수행

  - Dilithium
  - Key Generation: 비밀키를 통해 격자기반 다항식 연산 수행 $H(m) + a \cdot s \mod q$
  - Signature Verification: 격자 기반 다항식 곱셈 수행

곱셈 비교

	Karatsuba	Montgomery	CRT	NTT
용도	큰 정수 곱셈	모듈러 곱셈	큰 수 모듈러 곱셈 분할 (병렬화)	다항식 곱셈 최적화
복잡도	$O(n^{1.585})$	$O(n^2)$	$O(n)$ - 병렬화	$O(\mathrm{nlog}\, n)$
활용분야	정수 연산, 다항식 연산	RSA, ECC	RSA, ECC	격자기반암호, FHE
특징	Divide and Conquer 방식	나눗셈 없이 모듈러 수행	큰 정수를 작게 나눠서 계산	FFT와 유사 (정수기반)

Q & A