計算機数学I (2019)
第12回

照井 章（筑波大学 数理物質系 数学域）
Akira Terui (Institute of Mathematics, University of Tsukuba)

第11回のまとめ

- 有理数を剰余環に埋めこんでの計算
- 有理数の（循環）小数表記からの復元

第12回の内容

- 中国剰余算法の応用
  - モジュラ算法
    - 行列積の法計算 (§ 4.9)

モジュラ算法 (Modular algorithms)

モジュラ算法 (Modular algorithms)

1. 主に多倍長数を扱うアルゴリズムが対象
2. アルゴリズムで扱う数の絶対値の上界 M に対し、$2M < n$ を満たす n を算出
3. $n = n_1 \dots n_k$ かつ $\forall\, i \neq j\, (n_i \perp n_j)$ をみたす $n_1 \dots n_k$ を求める
4. $(\mathrm{mod}\, n_1), \dots, (\mathrm{mod}\, n_k)$ でアルゴリズムを実行
5. 中国剰余定理を用いて $(\mathrm{mod}\, n)$ での解を復元⇒2. より求める整数解を得る

行列積の法計算 (§ 4.9) (p. 64)

多倍長数を成分に持つ行列の積の計算

- $A, B \in Z^{m \times m}$
- $C = A\,B$ を計算する

$$C[r,t] = \sum_{s=1}^{m} A[r,s]\,B[s,t]$$

- 計算量を見積もる

多倍長数を成分に持つ行列の積の計算量

$$C[r,t] = \sum_{s=1}^{m} A[r,s]\, B[s,t]$$

- $|A[r,s]|, |B[s,t]| \leqq M$, $L := \mathrm{len}(M)$
- このとき $|A[r,s]\, B[s,t]| \leqq M^2$, $|C[r,t]| \leqq M^2 m =: M'$
- $m \leqq M$ を仮定する: ゆえに $\mathrm{len}(m) \leqq \mathrm{len}(M) = L$

多倍長数を成分に持つ行列の積の計算量

$$C[r,t] = \sum_{s=1}^{m} A[r,s]\, B[s,t]$$

- 長さ L の多倍長数どうしの乗算: m^3 回
  → $O(m^3 L^2)$
- 長さ len(M') の多倍長数の加算: m^3 回
  len(M') ≦ $2L$ + len(m) ≦ $3L$ = $O(L)$ → $O(m^3 L)$
- 合計 $O(m^3 L^2)$ ← 中国剰余定理で改善する

中国剰余定理を用いる際の法の決定

- $$C[r,t] \equiv \sum_{s=1}^{m} A[r,s]\,B[s,t] \pmod{n}$$

- $$C'[r,t] \equiv \sum_{s=1}^{m} A[r,s]\,B[s,t] \pmod{n}$$

  かつ $-n/2 \leqq C'[r,t] < n/2, n > 2M'$ ならば

  $C[r,t] = C'[r,t]$

中国剰余定理を用いる際の法の決定

- $\mathrm{C}[r, t] = C'[r, t]$ が成り立つ理由
  - $\mathrm{C}[r, t] \equiv C'[r, t] \pmod{n}$ より $n \mid (\mathrm{C}[r, t] - C'[r, t])$
  - 一方で、$|\mathrm{C}[r, t]| < M'$ より

    $|\mathrm{C}[r, t] - C'[r, t]| \leqq |\mathrm{C}[r, t]| + |C'[r, t]|$

    $\leqq M' + n/2 < n/2 + n/2 = n$

中国剰余定理を用いる際の法の決定

- $C[r, t] = C'[r, t]$ が成り立つ理由
  - $(C[r, t] - C'[r, t])$ が n の倍数、かつ

    $(C[r, t] - C'[r, t])$ は n よりも真に小さい

    $\Rightarrow (C[r, t] - C'[r, t]) = 0$, よって $C[r, t] = C'[r, t]$
- 法 n は $n > 2M' = 2M^2m$ を満たすようにとる
- 剰余環の元は $[-n/2, n/2)$ の範囲で求める

中国剰余定理を用いる際の法の決定

- $n = n_1 \ldots n_k$ かつ $\forall\ i \neq j\ (n_i \perp n_j)$ をみたす $n_1 \ldots n_k$ を求める
  - 実際は、$n_1 \ldots n_k$ にはあらかじめ素数を用意しておく
- 法 n は $n > 2M' = 2M^2m$ を満たすようにとる
- 剰余環の元は $[-n/2, n/2)$ の範囲で求める

中国剰余定理を用いる際の法の決定

- k の値 ($n_1, \ldots, n_k$ の個数):
  - $n_1, \ldots, n_k$ は単精度の素数とする
  - $n = n_1 \ldots n_k > 2M' = 2M^2 m$ を満たし、かつ必要最小限の大きさになるように k をとると

    $\mathrm{len}(n) = \mathrm{len}(n_1 \ldots n_k) = \mathrm{len}(n_1) + \ldots + \mathrm{len}(n_k)$

    $= 1 + \ldots + 1 = k = \mathrm{O}(\mathrm{len}(M')) = \mathrm{O}(L)$

アルゴリズム（中国剰余算法による行列積）

Algorithm (中国剰余算法による行列積)

入力: $A, B \in \mathbb{Z}^{m \times m}$ with $|A[r, t]|, |B[r, t]| \leqq M$

出力: $C = A\,B$

1. $n = n_1 \ldots n_k, n > 2\,M' = 2\,M^2 m, \forall\ i \neq j\ (n_i \perp n_j)$ を満たす $n_1 \ldots n_k$ を求める

アルゴリズム（中国剰余算法による行列積）

2. for $i \in [1..k]$ do

    for $r \in [1..m]$ do

        for $t \in [1..m]$ do

            $A_i[r, t] \leftarrow A[r, t] \bmod n_i$;

            $B_i[r, t] \leftarrow B[r, t] \bmod n_i$;

アルゴリズム（中国剰余算法による行列積）

3. for $i \in [1..k]$ do

    for $r \in [1..m]$ do

        for $t \in [1..m]$ do

$$C_i[r,t] \leftarrow \left(\sum_{s=1}^{m} A_i[r,s]\, B_i[s,t] \right) \pmod{n_i}$$

アルゴリズム（中国剰余算法による行列積）

4. for $r \in [1..m]$ do

    for $t \in [1..m]$ do

        $C[r, t] \leftarrow \mathrm{ChineseRemainder}(C_1[r, t] \bmod n_1,$

        $\ldots, C_k[r, t] \bmod n_k)$

5. return C;

計算量の見積もり

アルゴリズム（中国剰余算法による行列積）

Algorithm (中国剰余算法による行列積)

入力: $A, B \in \mathbb{Z}^{m \times m}$ with $|A[r, t]|, |B[r, t]| \leqq M$

出力: $C = A\,B$

1. $n = n_1 \ldots n_k, n > 2\,M' = 2\,M^2 m, \ \forall\ i \neq j\ (n_i \perp n_j)$ を満たす $n_1 \ldots n_k$ を求める

$n_1, \ldots, n_k$ にはあらかじめ求めておいた素数を充てるので、新たな計算はなし

アルゴリズム（中国剰余算法による行列積）

2. for $i \in [1..k]$ do
   for $r \in [1..m]$ do
      for $t \in [1..m]$ do
         $A_i[r, t] \leftarrow A[r, t] \bmod n_i$;
         $B_i[r, t] \leftarrow B[r, t] \bmod n_i$;

ループ k 回 = O(L)

ループ m^2 回

長さ L の多倍長数を単精度数で割る: O(L)
（詳しくは次回）

Step 2 の合計: O($m^2 L^2$)

アルゴリズム（中国剰余算法による行列積）

3. for $i \in [1..k]$ do
  for $r \in [1..m]$ do
    for $t \in [1..m]$ do

$$C_i[r,t] \leftarrow \left(\sum_{s=1}^{m} A_i[r,s]\, B_i[s,t] \right) \pmod{n_i}$$

ループ k 回 = O(L)

ループ m^2 回

単精度 乗算 m 回 + 加算 $m - 1$ 回 = O(m)

Step 3 の合計: O($m^3 L$)

アルゴリズム（中国剰余算法による行列積）

4. for $r \in [1..m]$ do
   for $t \in [1..m]$ do
   $C[r, t] \leftarrow \mathrm{ChineseRemainder}(C_1[r, t] \bmod n_1, \ldots, C_k[r, t] \bmod n_k)$
5. return C;

ループ m^2 回

中国剰余算法: $O(L^2)$
（詳しくは次回）

Step 4 の合計: $O(m^2 L^2)$

中国剰余算法による行列積: 計算量の合計

Algorithm (中国剰余算法による行列積)

Step 1: なし

Step 2: $O(m^2 L^2)$

Step 3: $O(m^3 L)$

Step 4: $O(m^2 L^2)$

合計: $O(m^3 L + m^2 L^2)$ $\leftrightarrow O(m^3 L^2)$ (従来の算法)

例 4.70（中国剰余算法による行列積）

$$A = \begin{pmatrix} 9 & 7 & -6 \\ -8 & -3 & 10 \\ 8 & -10 & 7 \end{pmatrix}, \quad B = \begin{pmatrix} -7 & 3 & 11 \\ -12 & 9 & -6 \\ 8 & -3 & 9 \end{pmatrix}$$

$M = 12, m = 3 \Rightarrow 2M^2 m = 2 \times 12^2 \times 3 = 864 < n,$

$n = n_1 \ldots n_k, \ \forall\, i \neq j \, (n_i \perp n_j)$

を満たす $n_1 \ldots n_k$ を求めると

$k = 3, n_1 = 9, n_2 = 10, n_3 = 11, n = 9 \times 10 \times 11 = 990$

例 4.70（中国剰余算法による行列積）

$$A \bmod 9 = \begin{pmatrix} 0 & -2 & 3 \\ 1 & -3 & 1 \\ -1 & -1 & -2 \end{pmatrix}, \quad B \bmod 9 = \begin{pmatrix} 2 & 3 & 2 \\ -3 & 0 & 3 \\ -1 & -3 & 0 \end{pmatrix}$$

より（各成分を [–4..4] の範囲で表していることに注意）

$$AB \bmod 9 = (A \bmod 9)(B \bmod 9) \bmod 9$$

$$= \begin{pmatrix} 3 & 0 & 3 \\ 1 & 0 & 2 \\ 3 & 3 & 4 \end{pmatrix}$$

例 4.70（中国剰余算法による行列積）

同様にして

$$AB \bmod 10 = (A \bmod 10)(B \bmod 10) \bmod 10$$

$$= \begin{pmatrix} 5 & -2 & 3 \\ 2 & -1 & 0 \\ 0 & 3 & 1 \end{pmatrix}$$

$$AB \bmod 11 = (A \bmod 11)(B \bmod 11) \bmod 11$$

$$= \begin{pmatrix} 3 & -2 & 3 \\ -4 & -4 & -2 \\ -1 & 1 & 2 \end{pmatrix}$$

例 4.70 (中国剰余算法による行列積)

$AB \bmod 9$, $AB \bmod 10$, および $AB \bmod 11$ より、

中国剰余算法を用いて各成分を復元することにより

を得る

$$AB = \begin{pmatrix} -195 & 108 & 3 \\ 172 & -81 & 20 \\ 120 & -87 & 211 \end{pmatrix}$$

第12回のまとめ

- モジュラ算法 (modular algorithms)
- 中国剰余算法による行列積の計算の効率化

第13回の内容

- 除算の計算量
- （拡張）Euclid互除法の計算量
- 中国剰余算法の計算量