計算機数学I (2019)
第13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

除算の計算量

多項式の剰余つき除算

- $a(x), b(x) \in \mathbb{Z}[x]$, $b(x)$: モニック
- $\deg(a) = k \geqq m = \deg(b)$

に対し、多項式の剰余つき除算を行う

多項式の剰余つき除算:アルゴリズム

Algorithm 多項式の剰余つき除算 (a, b)

入力: $a(x) = (a_0, a_1, \ldots, a_k) \in \mathrm{Z}[x], \quad a_k \neq 0, \quad a_j \in \mathrm{Z}$

$b(x) = (b_0, b_1, \ldots, b_{m-1}, 1) \in \mathrm{Z}[x], \quad b_j \in \mathrm{Z},$

$m \leqq k$

出力: $q(x) = (q_0, q_1, \ldots, q_{k-m}) \in \mathrm{Z}[x],$

$r(x) = (r_0, r_1, \ldots, r_{m-1}) \in \mathrm{Z}[x]$

s.t. $a(x) = q(x) \cdot b(x) + r(x)$

多項式の剰余つき除算:アルゴリズム

Algorithm 多項式の剰余つき除算 (a, b)

1. for $i \in$ [k..0] do $r_i \leftarrow a_i$;
2. for $i \in$ [k-m..0] do
   a. $q_i \leftarrow r_{i+m}$;$r_{i+m} \leftarrow 0$;
   b. for $j \in$ [m-1..0] do
      $r_{i+j} \leftarrow r_{i+j} - q_i b_j$;
3. return $(q_0, q_1, \ldots, q_{k-m}), (r_0, r_1, \ldots, r_{m-1})$

多項式の剰余つき除算：計算量の見積もり

Algorithm 多項式の剰余つき除算 (a, b)

1. for $i \in$ [k..0] do $r_i \leftarrow a_i$;
2. for $i \in$ [k-m..0] do
   a. $q_i \leftarrow r_{i+m}$; $r_{i+m} \leftarrow 0$;
   b. for $j \in$ [m-1..0] do
      $r_{i+j} \leftarrow r_{i+j} - q_i b_j$;
3. return $(q_0, q_1, \ldots, q_{k-m}), (r_0, r_1, \ldots, r_{m-1})$

代入のみ（影響なし）

ループ k-m+1 回

ループ m 回

乗算1回、加算1回

合計: $O(m\,(k-m))$

多項式の剰余つき除算：計算量の見積もり

- 係数間の加減乗除の回数は、除多項式 (divisor) と商 (quotient) の次数の積に比例

整数どうしの除算

単精度整数どうしの乗算

- $x, y \in \mathbb{Z}$

x	×	y	=	xy
(n桁)		(n桁)		(2n桁)
1ワード		1ワード		2ワード

分のメモリが必要

逆に、除算は(高々)2ワード÷ 1ワード

- a_0, a_1, b: 単精度整数

$$[a_0, a_1] = (a_1 \times 2^n + a_0) \quad \div \quad b \quad = \quad (q, r)$$

(2n桁) (n桁) (n桁, n桁)

2ワード 1ワード 1ワード, 1ワード

- q: 商、r: 剰余
- これを $(q, r) \leftarrow [a_0, a_1] \div b$ で表す
- 計算量は上の演算を1単位とする

単精度整数による多倍長整数の除算:
アルゴリズム

Algorithm 単精度整数による多倍長整数の除算 (a, b)

入力: $a = [a_0, a_1, \ldots, a_{k-1}]$, $a_{k-1} \neq 0, b \in [0..2^n\text{-}1]$

出力: $c = [q_0, q_1, \ldots, q_{k-1}], d \in [0..2^n\text{-}1]$

s.t. $a = c \cdot b + d$, $|d| < b$

単精度整数による多倍長整数の除算：アルゴリズム

Algorithm 単精度整数による多倍長整数の除算

1. for $i \in [0..k\text{-}1]$ do $r_i \leftarrow a_i$; $r_k \leftarrow 0$;
2. for $i \in [k\text{-}1..0]$ do
   a. $(q, r) \leftarrow [r_{i+1}, r_i] \div b$;
   b. $q_i \leftarrow q$; $r_i \leftarrow r$;
3. $c \leftarrow [q_0, q_1, \ldots, q_{k\text{-}1}]$; $d \leftarrow r$;
4. return c, d;

単精度整数による多倍長整数の除算：計算量

Algorithm 単精度整数による多倍長整数の除算

1. for $i \in [0..k\text{-}1]$ do $r_i \leftarrow a_i$; $r_k \leftarrow 0$;
2. for $i \in [k\text{-}1..0]$ do
   a. $(q, r) \leftarrow [r_{i+1}, r_i] \div b$;
   b. $q_i \leftarrow q$; $r_i \leftarrow r$;
3. $c \leftarrow [q_0, q_1, \ldots, q_{k-1}]$; $d \leftarrow r$;
4. return c, d;

代入のみ（影響なし）

ループ k 回

除算 1 回

合計: $O(k) = O(\mathrm{len}(a))$

多倍長数どうしの除算の計算量

- 多倍長数どうしの除算のアルゴリズムはより複雑なので今回の授業では省略

  (see e.g. V. Shoup: A Computational Introduction to Number Theory and Algebra, Cambridge University Press)

- a, b: 多倍長数、$\mathrm{len}(a) \geqq \mathrm{len}(b)$ のとき
- a を b で割る除算の計算量は $O(\mathrm{len}(a)\,\mathrm{len}(b))$

(拡張) Euclid互除法の計算量

- a, b: 多倍長数
- Euclid互除法、拡張Euclid互除法とも、計算量は $O(\mathrm{len}(a)\,\mathrm{len}(b))$
- 証明は各自の演習課題

中国剰余算法の計算量の見積もり

Chinese Remainder Algorithm

- $n_1, \ldots, n_k \in \mathrm{R}$: 互いに素
- $a_1, \ldots, a_k \in \mathrm{R}$

のとき

$$a \equiv a_i \pmod{n_i} \qquad (i = 1, \ldots, k)$$

を満たす a を求める

数の長さに関する仮定

- $n_1, \ldots, n_k$: 単精度, $\mathrm{len}(n_i) = \mathrm{O}(1)$
- $a_1, \ldots, a_k$: 単精度, $\mathrm{len}(a_i) = \mathrm{O}(1)$
- $n := n_1 \ldots n_k$: $\mathrm{len}(n) = \mathrm{O}(L)$, ゆえに $k = \mathrm{O}(L)$
- 計算量を L の関数として求める

Chinese Remainder Algorithm の手順

1. for $i \in [1..k]$ (e_i を求める)
   a. $n_i^* \leftarrow n / n_i$;
   b. t_i: n_i を法とする n_i^* の逆元
      EEA(n_i^*, n_i) から求める
   c. $e_i \leftarrow t_i n_i^*$
2. $a \leftarrow a_1 e_1 + \ldots + a_k e_k$
3. $a \leftarrow a \bmod n$

Chinese Remainder Alg

1. for $i \in [1..k]$ (e_iを求める
   a. $n_i^* \leftarrow n / n_i$;
   b. $t_i : n_i$を法とするn_i^*の逆元
      EEA(n_i^*, n_i) から求め
   c. $e_i \leftarrow t_i n_i^*$
2. $a \leftarrow a_1 e_1 + \ldots + a_k e_k$
3. $a \leftarrow a \bmod n$

len(n) = O(L), len(n_i) = O(1)
除算の計算量は O(L)

len(n_i^*) = O(L), len(n_i) = O(1)
EEAの計算量は O(L)

len(n_i^*) = O(L), len(t_i) = O(1)
乗算の計算量は O(L)

len(e_i) = O(L), len(a_i) = O(1)
乗算と加算の計算量の合計=O(L)

len(a) = O(L), len(n) = O(L)
除算の計算量は O(L^2)

Chinese Remainder Algorithm の計算量

- $n_1, \ldots, n_k$: 単精度, $\mathrm{len}(n_i) = O(1)$
- $a_1, \ldots, a_k$: 単精度, $\mathrm{len}(a_i) = O(1)$
- $n := n_1 \ldots n_k$: $\mathrm{len}(n) = O(L)$, ゆえに $k = O(L)$
- 計算量を L の関数として求めると $O(L^2)$

第13回のまとめ

- 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

第14回の内容

- 高速乗算法:

  Karatsuba（カラツバ）の乗算アルゴリズム

  - 整数の乗算アルゴリズム
  - 1変数多項式の乗算アルゴリズム