計算機数学I (2019)
第4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3回のまとめ

- 符号なし多倍長整数の表現
- CPUのレジスタ

  （汎用レジスタ，状態レジスタ）
- 多倍長整数の加算の原理
- アルゴリズム，疑似コード，制御構造
- 符号なし多倍長整数の加算のアルゴリズム

第4回の内容

- 計算量の概念
- 多倍長整数の加算の計算量
- 1変数多項式の表現
- 1変数多項式の加算のアルゴリズムと計算量

計算量

計算量 (Computational complexity)

- アルゴリズムの効率を測る上での指標の一つ
  - 時間計算量 (Time complexity)

    必要な計算のステップ数
  - 空間計算量 (Space complexity)

    計算に必要な記憶容量

計算量の表し方: 漸近記法 (p. 10)

$$T : \mathbb{N} \longrightarrow \mathbb{R}_{\geq 0}$$
$$n \mapsto T(n)$$

n: 入力の「サイズ」
（数値の大きさ、桁数、ワード数、多項式の次数など）

漸近記法

- 例: アルゴリズム 1 と 2 の時間計算量の比較
  - T_1: アルゴリズム1の時間計算量
  - T_2: アルゴリズム2の時間計算量
  - $T_1(n) = a\,n\ (a > 0), T_2(n) = b\,n^2\ (b > 0)$

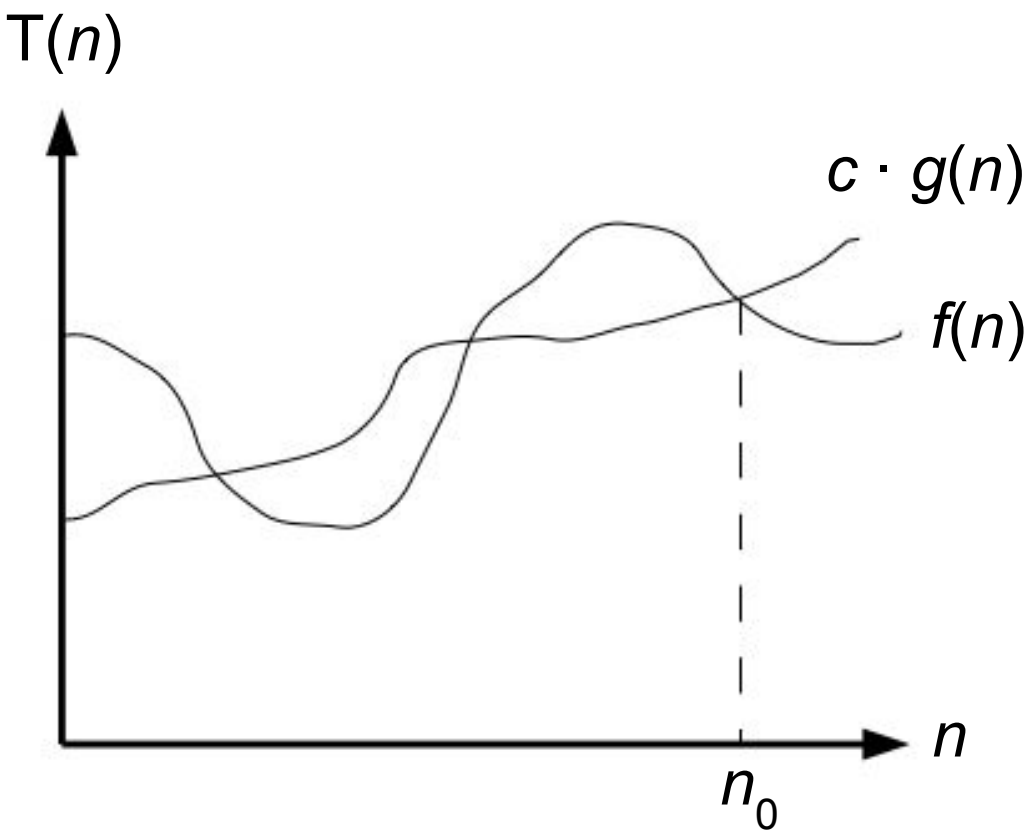
    ⇒ あるnの時点で $T_2(n)$ が $T_1(n)$ を追い越す
  - 他の例:
    - $T_3(n) = c^n\ (c > 0), T_4(n) = d\log(n)\ (d > 0)$

時間計算量の表示: 漸近記法 (定義 1.1)

- O-記法: 漸近的上界
- o-記法
- Ω-記法: 漸近的下界
- Θ-記法
- ω-記法

O-記法（漸近的上界）

$$f(n) = O(g(n)) \overset{\text{def}}{\iff} \exists c > 0,\ \exists n_0 > 0$$
$$\text{s.t. } n \geq n_0 \implies 0 \leq f(n) \leq c \cdot g(n)$$

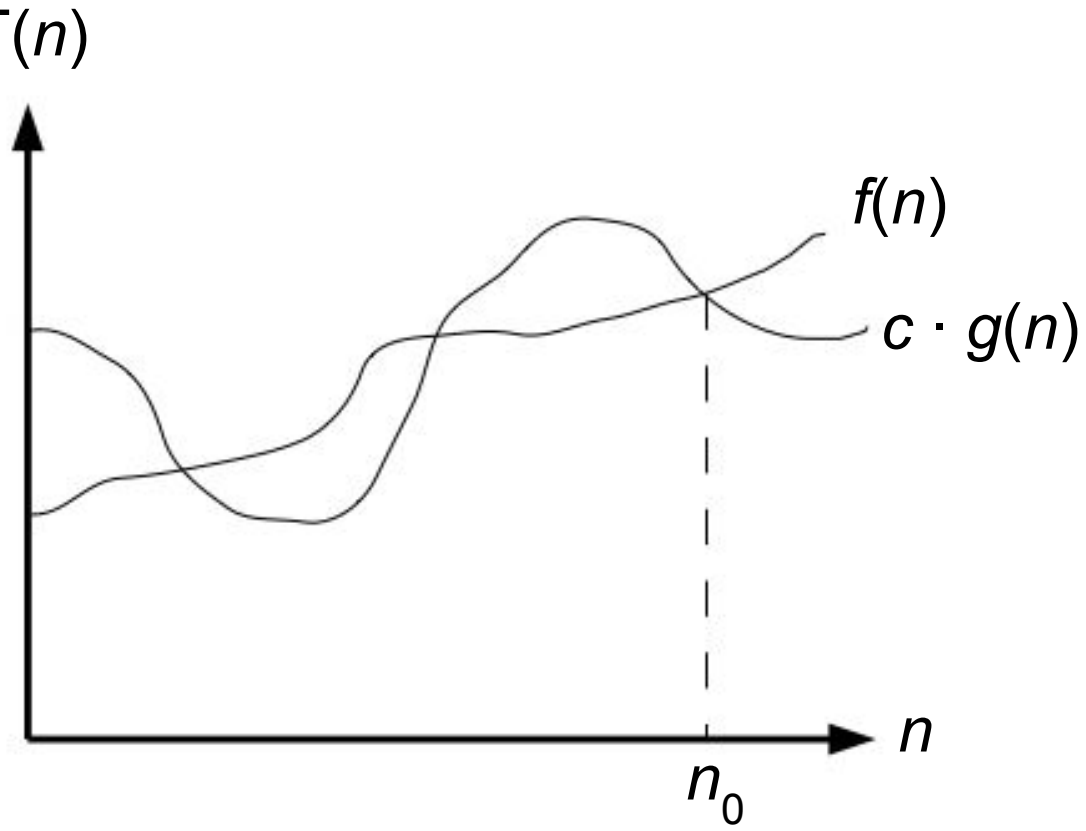


o-記法

$$f(n) = o(g(n)) \overset{\text{def}}{\iff} \forall c > 0,\ \exists n_0 > 0$$
$$\text{s.t. } n \geq n_0 \implies 0 \leq f(n) \leq c \cdot g(n)$$
$$\iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

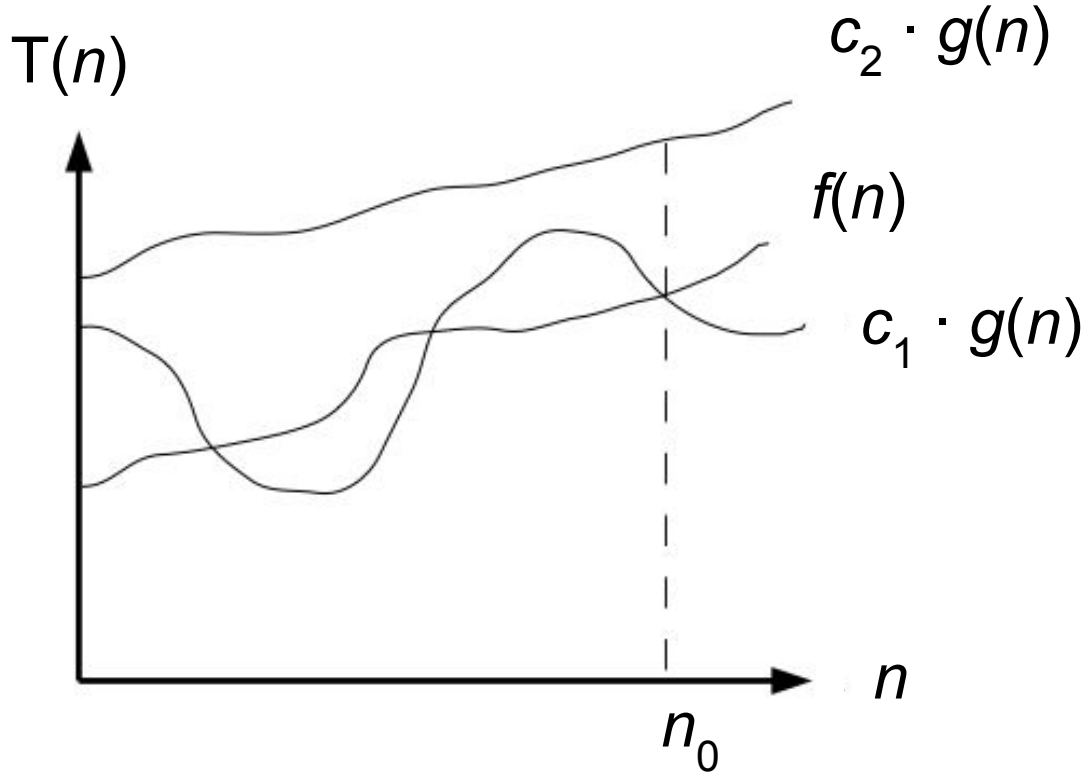
Ω-記法（漸近的下界）

$$f(n) = \Omega(g(n)) \overset{\text{def}}{\iff} \exists c > 0,\ \exists n_0 > 0$$
$$\text{s.t. } n \geq n_0 \Longrightarrow 0 \leq c \cdot g(n) \leq f(n)$$



Θ-記法

$$f(n) = \Theta(g(n)) \overset{\text{def}}{\iff} \exists c_1 > 0,\ \exists c_2 > 0,\ \exists n_0 > 0$$
$$\text{s.t. } n \geq n_0 \Longrightarrow 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$



ω-記法

$$f(n) = \omega(g(n)) \overset{\text{def}}{\iff} \forall c \geq 0,\ \exists n_0 \geq 0$$
$$\text{s.t. } n \geq n_0 \implies 0 \leq c \cdot g(n) \leq f(n)$$
$$\iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = +\infty$$

多倍長整数の加算の計算量 (p. 16)

多倍長整数の加算の計算量

- 計算量はワード毎の四則演算を単位とする

多倍長整数の加算のアルゴリズム

Algorithm 多倍長整数の加算 (a, b)

入力: $a = [a_0, a_1, \ldots, a_{k\text{-}1}], b = [b_0, b_1, \ldots, b_{k\text{-}1}]$

出力: $c = [c_0, c_1, \ldots, c_{k\text{-}1}]$ s.t. $c = a + b$

1. $\gamma_0 \leftarrow 0$;
2. for $i \in [0..k\text{-}1]$ do $[c_i, \gamma_{i+1}] \leftarrow a_i + b_i + \gamma_i$;
3. return $[c_0, c_1, \ldots, c_{k\text{-}1}]$;

多倍長整数の加算のアルゴリズム

Algorithm 多倍長整数の加算 (a, b)

入力: $a = [a_0, a_1, \ldots, a_{k-1}]$, $b = [b_0, b_1, \ldots, b_{k-1}]$

出力: $c = [c_0, c_1, \ldots, c_{k-1}]$ s.t. $c = a + b$

ループ k 回

加算 2 回

1. $\gamma_0 \leftarrow 0$;
2. for $i \in [0..k\text{-}1]$ do $[c_i, \gamma_{i+1}] \leftarrow a_i + b_i + \gamma_i$;
3. return $[c_0, c_1, \ldots, c_{k-1}]$;

多倍長整数の加算の計算量

- アルゴリズム全体で行われる計算 : 加算 $2k$ 回
- アルゴリズム全体の計算量 :

  $2k = O(k) = \Theta(k)$
- 多倍長整数の加算の計算量は、整数の長さ（ワード長 ）k に比例する

符号つき多倍長整数

- 2の補数を用いた表現 $v = [a_0, a_1, \ldots, a_{k-1}]$
- ただし、最上位ワード a_{k-1} の最上位ビットは符号を表す

$$a_{k-1} = \sum_{j=0}^{n-2} m_j 2^j + (-1) m_{n-1} 2^{n-1}$$

符号つき多倍長整数

$$a_{k-1} = \sum_{j=0}^{n-2} m_j 2^j + (-1) m_{n-1} 2^{n-1}$$

a_{k-1} の値の範囲は $-2^{n-1} \leqq a_{k-1} \leqq 2^{n-1}-1$

よって v の値の範囲は $-2^{nk-1} \leqq v \leqq 2^{nk-1}-1$

2.2 多項式の表現

2.2.1 多項式の表現と加算（p. 17）

多項式の表現と加算

- R: 可換環
- $R[x, y, \ldots, z]$: 変数 $x, y, \ldots, z$ をもつR上の多項式全体

  （$R[x, y, \ldots, z]$ に自然に加減乗算が定義され，$R[x, y, \ldots, z]$ は可換環をなす）
- 当面は変数を x: 1変数、係数を整数環 Z とする

主項，主係数，次数，モニック

- 定義（主項，主係数，次数，モニック）

  $a(x) = a_n x^n + \ldots + a_1 x + a_0 \in \mathrm{Z}[x],\ \ a_n \neq 0$

  - $a_n x^n$: $a(x)$**の主項** (the leading term), $\mathrm{lt}(a(x))$
  - a_n : $a(x)$**の主係数** (the leading coefficient), $\mathrm{lc}(a(\mathrm{x}))$
  - n: $a(x)$**の次数** (degree), $\deg(a(x))$
  - $a(x)$**がモニック** (monic) $\Leftrightarrow \mathrm{lc}(a(\mathrm{x})) = 1$

1変数多項式の表現

- 定義（1変数多項式の表現）
  - $a(x) = a_n x^n + \ldots + a_1 x + a_0 \in \mathrm{Z}[x], \qquad a_n \neq 0$
  - これを $(a_0, a_1, \ldots, a_n) \in \mathrm{Z}^{n+1}$ で表す

1変数多項式の加算

$$a(x) = a_k x^k + \cdots + a_1 x + a_0$$

$$\leftrightarrow (a_0, \ldots, a_k) \in \mathbb{Z}^{k+1},\ a_k \neq 0,$$

$$b(x) = b_k x^k + \cdots + b_1 x + b_0$$

$$\leftrightarrow (b_0, \ldots, b_k), \in \mathbb{Z}^{k+1},\ b_k \neq 0$$

に対し

$$c(x) = a(x) + b(x) = c_k x^k + \cdots + c_1 x + c_0$$

$$\leftrightarrow (c_0, \ldots, c_k) \in \mathbb{Z}^{k+1},\ c_k \neq 0$$

を求める

1変数多項式の加算のアルゴリズム

Algorithm 1変数多項式の加算 $(a(x), b(x))$

入力: $a(x) = (a_0, a_1, \ldots, a_k), b(x) = (b_0, b_1, \ldots, b_k)$

出力: $c(x) = (c_0, c_1, \ldots, c_k)$ s.t. $c(x) = a(x) + b(x)$

1. for $i \in [0..k]$ do $c_i \leftarrow a_i + b_i$;
2. return $(c_0, c_1, \ldots, c_k)$;

1変数多項式の加算のアルゴリズム

- 注意
  - 入力多項式 a, b が$\deg(a) \neq \deg(b)$ の場合にはアルゴリズムでの対応（拡張）が必要
  - 入力多項式の各係数 a_i, b_i は一般に多倍長整数
    - 計算量を見積もる際は，多倍長整数の加算の計算量を考慮する必要がある

1変数多項式の加算の計算量

$a(x) = (a_0, a_1, \ldots, a_k)$

＋　＋＋　＋

$b(x) = (b_0, b_1, \ldots, b_k)$

↓　↓↓　↓

$c(x) = (c_0, c_1, \ldots, c_k)$

加算 k+1 回 = O(k)

各項の係数の計算量は
各項の多倍長数の長さに比例
O(max{len(a_i), len(b_i)})

1変数多項式の加算の計算量

- 全体では各項の係数の（多倍長数の）長さの和に比例

$$O\left(\sum_{i=0}^{k} \max\{\mathrm{len}(a_i), \mathrm{len}(b_i)\}\right)$$

- 係数がすべて単精度（1ワード）ならば O(k)（多項式の次数に比例）
- 係数がすべて長さ m ワードならば O(mk)

第4回のまとめ

- 計算量の概念
- 符号なし多倍長整数の加算の計算量
- 符号つき多倍長整数の表現
- 1変数多項式の加算のアルゴリズム
- 1変数多項式の加算の計算量

第5回の内容

- 1変数多項式のホーナー (Horner) 法
  - 非負整数の2進・10進変換
  - 小数，分数の2進・10進変換