5/11 cont'd

計算量　Computational complexity

算法の効率を測る上での理論的指標の一つ.

(この授業で扱う)
→ ◎ 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.

5/18

▨ 計算量の表し方 … 「漸近記法」 (§1.2)

5/25

$$T : \mathbb{N} = \{0,1,2,\dots\} \longrightarrow \mathbb{R}_{\geq 0}$$
$$n \longmapsto T(n)$$

n ↑ 入力データの「サイズ」 … 数の桁数, ワード数, 多項式の次数, … .

例　$f(n)$　n: 多倍長整数のワード数.

$f(n) = an$, $g(n) = bn^2$, $a>0$, $b>0$

⟶ いずれは g が f を「追い抜く」

⟶ f の方が, 計算量が「漸近的に」より小さい.

他の例 : $h(n) = c^n$ $(c>0)$, $p(n) = d\log(n)$ $(d>0)$ 等. ▨

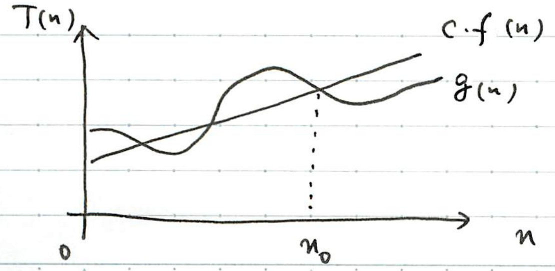
▨ 計算量の記法 (notations)

Def 1.1 (漸近記法). $f(n), g(n) : \mathbb{N} \longrightarrow \mathbb{R}$.

① O-notation : 漸近的上界.

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

↑ テキストで ∈ となっているものと本質的に変わらない. 以下同様.



(数学の O 記法と同じ)

② o-notation

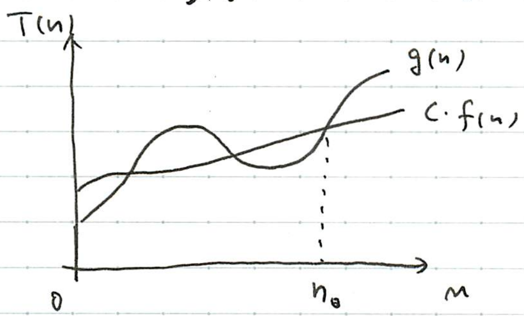
(数学の o 表記と同じ)

$g(n) = o(f(n)) \overset{\text{def}}{\Longleftrightarrow} \forall c > 0,\ ^{\exists} n_0 > 0$
s.t. $n \geq n_0 \Rightarrow 0 \leq g(n) \leq c \cdot f(n)$

$\left(\Leftrightarrow \lim_{n \to \infty} \frac{g(n)}{f(n)} = 0. \right)$

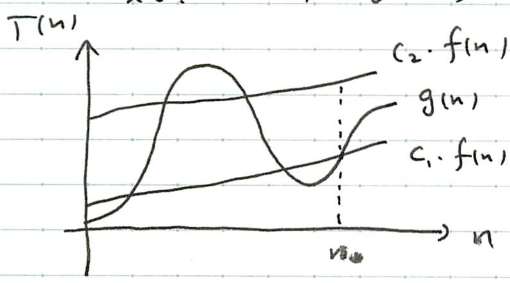
③ Ω-notation : 漸近的下界

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



④ Θ-notation

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



⑤ ω-notation

$g(n) = \omega(f(n)) \overset{\text{def}}{\Longleftrightarrow} \forall c > 0,\ ^{\exists} n_0 > 0$
s.t. $n \geq n_0 \Rightarrow 0 \leq c \cdot f(n) \leq g(n).$

$\left(\Leftrightarrow \lim_{n \to \infty} \frac{g(n)}{f(n)} = +\infty. \right)$

・有効性：有効な（意味のある）計算が行われること.

（例）有効でない例：
整数上のEuclidの互除法に、有理数や無限桁の小数を与えたりすること.

▨ アルゴリズム（多倍長整数の加算）

入力：多倍長整数 $a = [a_0\ a_1\ \cdots\ a_{k-1}]$, $b = [b_0\ b_1\ \cdots\ b_{k-1}]$

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

(1) $r_0 \leftarrow 0$;
(2) for $i \in [0..k-1]$ do
　　$[c_i\ r_{i+1}] \leftarrow a_i + b_i + r_i$;
(3) return $[c_0\ c_1\ \cdots\ c_{k-1}]$;

5/18

← ここで $\boxed{\text{計算量}}$ を導入 (p.3-2)

▨ 多倍長整数の加算の計算量

5/25 つづき

☆ 計算量はワードごとの四則演算の回数を単位とする (p.4)

上のアルゴリズムの計算量（演算回数）を調べると…

(1) $r_0 \leftarrow 0$;　　代入は今回は無視
(2) for $i \in [0..k-1]$ do　　ループ k回
　　$[c_i\ r_{i+1}] \leftarrow a_i + b_i + r_i$;　　加算2回
(3) return $[c_0\ c_1\ \cdots\ c_{k-1}]$;　　return文は無視

より、全体で行われる演算は加算2k回.

よって、このアルゴリズムの計算量は $2k = O(k) = \Theta(k)$.

すなわち、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.

$$v = [a_0 \cdots a_{k-1}] = \sum_{i=0}^{k-1} a_i 2^{ni}$$

但し、最上位ワード a_{k-1} の最上位ビットは符号を表す. すなわち

$$a_{k-1}$$

$$v' = \begin{array}{|c|c|c|c|c|} \hline m_{n-1} & m_{n-2} & \qquad\qquad\qquad & m_1 & m_0 \\ \hline \end{array}$$

$$\begin{array}{ccccc} -2^{n-1} & 2^{n-2} & \qquad\qquad\qquad & 2^1 & 2^0 \end{array}$$

より、 $$v' = \sum_{j=0}^{n-2} m_j 2^j + (-1) m_{n-1} \cdot 2^{n-1}.$$

ゆえに $$-2^{n-1} \le v' < 2^{n-1}.$$

よって、v 全体としては $-2^{nk-1} \le v < 2^{nk-1}$ の値をとる.

$\boxed{\text{問題2.7}}$ ・符号つき多倍長整数の減算を、1ワード=nビットの整数の減算の組み合わせで表す.

・基本的な方法は多倍長整数の加算とほぼ同様だが、最上位ワードの最上位ビットを符号に用いているので、この部分に対応した処理(の吟味)が必要.

2.2 多項式の表現

2.2.1 多項式の表現と加算

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

  ($R[x, y, \dots, z]$ 上へ自然に加減乗算が定義でき, $R[x, y, \dots, z]$ は可換環となる).

- 当面は 変数を x : 1変数.
  係数を $\mathbb{Z}$: $\mathbb{Z}[x]$.

Def (主項, 主係数, 次数, モニック)

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

$a_n x^n$: $a(x)$ の主項 (the leading term), $lt(a(x))$.
a_n : ―― の主係数 (the leading coefficient), $lc(a(x))$.
n : ―― の次数 (degree), $\deg(a(x))$.
$a(x)$ がモニック (monic) $\iff$ $lc(a(x)) = 1$.

☒

Def (1変数多項式の表現)

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

を

$$(a_0, \dots, a_n) \in \mathbb{Z}^{n+1}$$

で表す. ☒

5/25

◎ 1変数多項式の加算.

6/1

$$a(x) = \sum_{i=0}^{k} a_i x^i = (a_0, a_1, \dots, a_k),$$

$$b(x) = \sum_{i=0}^{k} b_i x^i = (b_0, b_1, \dots, b_k)$$

に対し,

$$c(x) = a(x) + b(x) = \sum_{i=0}^{k} c_i x^i = (c_0, c_1, \dots, c_k)$$

とおく.