5/16

$\boxed{\text{計算量}}$ 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.

▨ 計算量の表し方 … 「漸近記法」(§1.2)

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

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. ▨

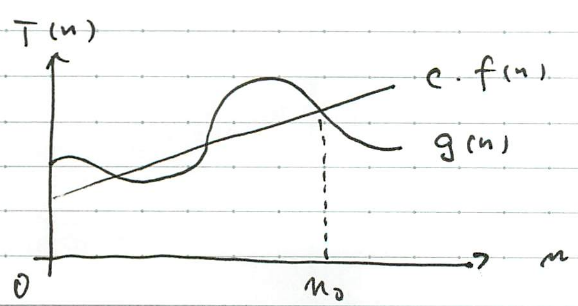
▨ 計算量の記法 (notations)

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

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 $\in$ となっているものと本質的に同じ. 以下同様.)



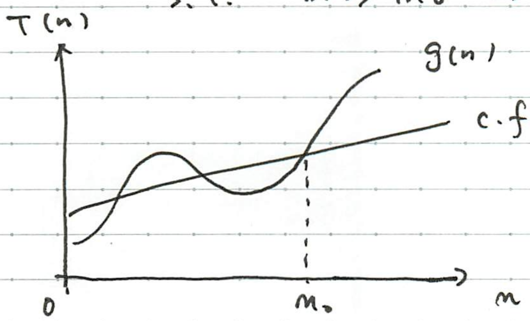
(数学の O 表記と同じ)

② o - notation　　　　（数学の o 表記と同じ）

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

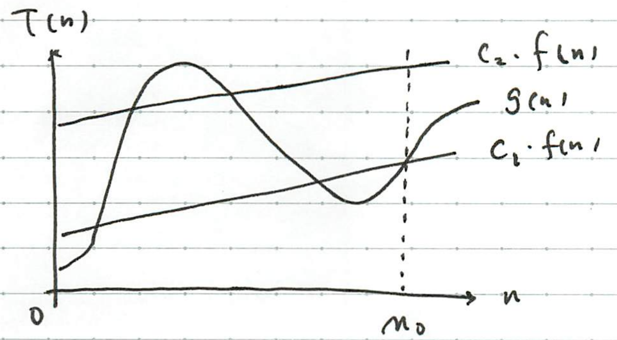
③ Ω - notation　（漸近的下界）

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



④ Θ - notation

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



⑤ ω - notation

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

⑩ 多倍長整数の加算の計算量.

☆ 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)

p.10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. アルゴリズム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.

⑪ 符号つき多倍長整数.

・「2の補数」を用いた表現.

$$v = [a_0 \ a_1 \cdots a_{k-1}] = \sum_{i=0}^{k-1} a_i 2^{ni}$$

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} =

m_{n-1}	m_{n-2}	$\cdots$	m_1	m_0
-2^{n-1}	2^{n-2}		2^1	2^0

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

ゆえに $-2^{n-1} \le a_{k-1} < 2^{n-1} - 1$

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

となる.