第2章　基礎理論

1．計算複雑性理論 (CCT : Computational Complexity Theory)

計算複雑性理論アルゴリズムのスケーラビリティや、特定の計算問題の解法の複雑性(計算問題の困難さ)などを数学的に扱う理論である。計算量理論、計算の複雑さの理論、計算複雑度の理論ともいう。

あるアルゴリズムへの入力データの長さを増やしたとき、実行時間や必要な記憶量はどのように増えるか？

⇩

アルゴリズムが入力データ長の増大にうまく対応できるか否かで、計算機が現実的な問題を解決するのに役に立つか否かが決まる

⇩

計算複雑性理論では、計算問題やそれを解くアルゴリズムを、NP や P といった複雑性クラスに分類

Definition 1.1（P）

P(polynomial time) とは多項式時間で解ける判定問題である。
判定問題のうち、ある決定性チューリング機械によって多項式時間で解かれるものの全体をいう。

Definition 1.2（NP）

NP(Non-deterministic Polynomial time(非決定性多項式時間)) の定義は次の 2 つである、ただしこれらはお互い同値である

1. 非決定性チューリングマシンによって多項式時間で解くことができる問題。
2. yes となる証拠が与えられたとき、その証拠が本当に正しいかどうかを多項式時間で判定できる問題。

Definition 1.3（チューリングマシン）

チューリングマシン (Turing Machine) は計算機を数学的に議論するための単純化・理想化された仮想機械である。

Definition 1.4（形式的なチューリングマシン）

形式的にチューリング機械とは次の 7 つ組 $M = \langle Q, \Gamma, b, \Sigma, \delta, q_{\text{init}}, q_{\text{acc}} \rangle$ である。

- Q は有限集合であり、その元を状態という。
- Γ は Q に交わらない有限集合であり、字母とよばれる。その元を記号という。
- b は Γ の元であり、空白記号とよばれる。
- Σ は Γ - $\{b\}$ の部分集合であり、入力字母とよばれる。その元を入力記号という。
- δ は $Q \times \Gamma$ から $Q \times \Gamma \times \{left, right\}$ への写像であり、遷移函数とよばれる。$\delta(q, a) = (q', a', m)$ は、「現在の状態が q であり、着目位置にある記号が a であれば、状態を q' に移し、着目位置に記号 a' を書き込んでから、着目位置を m 方向に 1 つずらす」と読む。
- q_{init} は Q の元であり、初期状態とよばれる。
- q_{acc} は Q の元であり、受理状態とよばれる。

Definition 1.5

計算量 t_C をもつ複雑性クラス **C** に 或る計算問題 **X** が属するとは、あるアルゴリズム **A** が存在して、問題 **X** が **A** により t_C 以下で解決されることを意味し、問題 **X** の複雑性の上界を与えることである。

Definition 1.6

問題 **X** の複雑性のよりよい上界を求めることは、問題 **X** をより少ない計算資源で解くアルゴリズムを発見する (あるいはその存在を示す) ことである。

Definition 1.7

ある計算問題 $\mathbf{X}$ が、ある複雑性クラス $\mathbf{C}$ に属しないとは、問題 $\mathbf{X}$ は、いかなるアルゴリズムをもってしても、計算量 t_C 以下では解決できないことを意味し、問題 $\mathbf{X}$ の複雑性の下界を与えることである。

Definition 1.8

問題 $\mathbf{X}$ の複雑性の下界を示すことは、理論的意義を有するだけではなくて、暗号理論においては、ある暗号方式が計算量的に解読不能であることを示すことである。

最も重要な課題

$\mathbf{P} \neq \mathbf{NP}$ 予想の証明である。

NP 完全問題は、解法が正しいかどうかは簡単に確かめられるが、正確な解を探す方法はスケーラブル ではない問題である。NP クラスが P クラスより範囲が広いことが確定すると、それらの問題にはスケーラブ ルな解が存在しないことが確定する。このため、P≠ NP 予想の証明は重要とされているのである。

1.1 計算問題と計算量・複雑性

計算複雑性理論で扱う問題とは、ある一連の問いの集合があり、各問いは有限長の文字列で表され、与えられた 問いに対して解を求めて出力する計算問題である。
アルゴリズムの計算量とは、計算機がそのアルゴリズムの実行に要する計算資源の量をいい、アルゴリズムの スケーラビリティを示す。形式的には計算機をチューリング機械や即時呼出機械(random access machine)な どの計算モデルとして定式化したうえで、アルゴリズムの使用する資源の量を入力データ長などに対する関数 として表す。
時間計算量は、あるアルゴリズムを使ったときに問題のインスタンスを解くのに要するステップ数を意味し、 入力データの長さ(ビット数などで表現できる)の関数として表される。
空間計算量は、同様の概念であり、アルゴリズムが必要とする記憶容量を意味する。
計算問題の複雑性とは、それがどの計算モデルにおいて、どれほどの計算量のアルゴリズム によって解けるかで測られる。

1.2 決定問題

- 決定問題とは、答えが「はい」か「いいえ」になる問 題を指す。
- 答えが「はい」かどうかを確認する問題と、答えが「いいえ」かどうかを確認する問題 を区別する。「はい」と「いいえ」を逆転させた問題は、元の問題の補問題と呼ばれる。
- ある問題の解を求める計算量とその補問題の解を求める計算量は同じであるが、問題のあるインスタンスに ついて「はい」となる証拠を与えられて、その証拠が正しいかを判定する計算量は同じとは限らない。

1.3 計算資源

計算問題の難しさを時間、空間、無作為性、反復性、その 他のより直観的でない尺度などで必要とする様々な計算資源の観点で分析することである

計算資源は決定性のある計算機で問題を解くのに必要な

- 決定性時間 (DTIME) :「計算時間(演算回数)」
- 決定性空間 (DSPACE) :「記憶装置」の量

1.4 複雑性クラス

複雑性クラス P は、チューリング機械で多項式時間で解ける決定問題の集合。直感的に言えば最悪の場合でも効率的に解くことができる問題

複雑性クラス NP は、非決定性チューリング機械で多項式時間で解ける決定問題の集合

困難 (hard)

クラス C に対して、問題 P が C 困難であるとは、C に属する任意の問題を P に帰着(多くの場 合多項式時間帰着を考えるが、そうでない場合もある)できるということである。すなわち P は C に属 するいかなる問題よりも、同等かそれ以上に難しいということである。ただし、C 完全と異なり、P 自 身は C に属するとは限らない。

完全 (complete)

クラス C に対して、問題 P が C 完全であるとは、P が C に属しかつ C 困難ということで ある。すなわち P は C に属する問題の中で、本質的に最も難しい問題であるという

1.5 未解決の問題

1.5.1 P=NP 問題（P = NP が証明されていない）

NP が P と同じかどうかという疑問、つまり、非決定的な多項式時間で解くことのできる問題は決定的な多項式時間でも解くことができるかは、理論計算機科学における最重要問題の 1 つである

暗号理論の多くが NP の困難さに依存してい るため、P と同じであることが判明すると使い物にならなくなる

P = NP が証明されていないため、ある問題をNP完全と判明している問題に還元できるということは、その問題の多項式時間の解法が未知である

⇕

すべてのNP問題はNP完全問題に還元できるため、NP完全問題の多項式時間の解法を発見すれば、P = NPが証明される

たとえP = NPが成立しても、NP困難な問題は多項式時間で解けるとは限らない

1.5.2 NP における不完全問題

NPクラスに属する問題でPクラスには属しないがNP完全でもない問題は存在するか?という問題

⇓

P = NP であることが示されれば、そのような問題が存在する

1.5.3 NP = co-NP (NP ≠ co-NPと考えられているが証明されていない)

co-NP クラスはNP問題の補問題の集合

NP ≠ co-NP

⇩

NP 完全問題はco-NP には含まれない

⇩

co-NP 完全問題はNP には含まれない

1.5.4 Intractability

理論上計算可能な問題であっても、実際に解くことができない問題をintractableであるという

例

EXPTIME

計算量理論において、チューリング機械で $\mathrm{O}(2^{p(n)})$ の時間で解ける全ての決定問題の集合である。ここで、$p(n)$ は n の多項式関数である。DTIME では次のように表される。

$$\mathrm{EXPTIME} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}\left(2^{n^k}\right)$$

そして、以下の関係が知られている。

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq PSPACE \subseteq EXPTIME \subseteq NEXPTIME \subseteq EXPSPACE$$

また、時間階層定理と領域階層定理により、次のようになる。

$$\mathbf{P} \subsetneq EXPTIME \text{ かつ } \mathbf{NP} \subsetneq NEXPTIME \text{ かつ } PSPACE \subsetneq EXPSPACE$$

Definition 1.9 (DTIME)

DTIME という資源は複雑性クラスの定義に使われ、複雑性クラスとは、ある特定の計算時間量で解ける全ての決定問題の集合である。入力長 n の問題を解くのに $f(n)$ の計算時間がかかる場合、その複雑性クラスは DTIME($f(n)$)（または TIME($f(n)$)）となる。このとき使用するメモリ空間量に制限はないが、他の複雑性尺度は制限されることもある。

指数関数時間の解法がなぜ実際には使えないか？

2^n回の操作を必要とする問題では、n =100とすると、$2^{100} \fallingdotseq 10^{30}$

1 秒間に10^{12} (1テラ)回命令を実行できる計算機を想定すると、

その問題を解くには約4×10^{10} 年かかる。

2 ランダウの記号（Landau symbol）

ポール・バッハマンによって発明され、エドムント・ランダウが広めた。ランダウの漸近記法(asymptotic notation)、ランダウ記法(Landau notation) あるいは主要な記号としてOを用いることから(ランダウの)O-記法などともいう。

関数極限における漸近挙動、すなわち値の変動のおおよその評価を与えるための記法である。

記号Oは通常、関数の大きさを上からの漸近的な上界を別のわかりやすい関数によって記述するために用いる。

上界や下界あるいはその両側からの評価を様々に与えることで、O,o, Ω , ω , Θ
といった類似記法が 定義される。
Oは上界Ω は下界のみをそれぞれ定義します。o とωもそれぞれ上界と下界を定義するが、全て定数cについて成り立つ必要がある。

Definition 2.1

$n, n_0, c_1, c_2 \in R$

$$\Theta(g(n)) = \left\{ f(n) \middle| {}^{\exists}c_1, c_2, {}^{\forall}n, {}^{\exists}n_0 \leq n,\ s.t.\ 0 \leq n_0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \right\}$$

Definition 2.2

$f(x)$ と $g(x)$ は実数からなるある集合上で定義されているとするとき、「$f(x)$ が $x \to a$ のとき $\mathrm{O}(g(x))$ 程度である」ということを

$$^{\exists}x_0, {}^{\exists}M > 0 \quad \text{s.t.} \quad x > x_0 \Rightarrow |f(x)| < M|g(x)|$$

となることとして定める。また、ある実数 a の付近での挙動を記述するならば「$f(x)$ が $x \to \infty$ のとき $\mathrm{O}(g(x))$ のオーダーである」ということを

$$^{\exists}\delta > 0, {}^{\exists}M > 0 \quad \text{s.t.} \quad |x - a| < \delta \Rightarrow |f(x)| < M|g(x)|$$

であることと定める。a の十分近くで $g(x)$ が 0 の値をとらないならこれらの定義を上極限を使って統一的に記述できる。つまり「$f(x)$ が $x \to a$ のとき $\mathrm{O}(g(x))$ の程度である」とは

$$\overline{\lim_{x \to a}} \left| \frac{f(x)}{g(x)} \right| < \infty$$

が満たされることである。

準指数関数　　超多項式

$O(n^c)$ と $O(c^n)$ は全く異なる。　（発散：$O(n^c) < O(c^n)$）

超多項式（*superpolynomial*）：どのような定数 c に対しても n^cより速く発散する関数
準指数関数（*subexponential*）：どのような定数 c に対しても c^n よりも遅く発散する関数

$O(\log n)$ と $O(\log(n^c))$ は全く等価である。

$\log(n^c) = c \log n$ より2つの指数関数は定数係数のみが異なり、これは big O−記法では無視される

- 同様に異なる底を持つ対数関数も等価であるが、一方、異なる底を持つ指数関数は等価ではない。（例：2^n と 3^n は同じオーダーではない）
- アルゴリズムが n^2 のオーダーで動くとき、n を cn で置き換えれば計算量は c^2n^2 となり、O−記法では c^2 は無視されるので計算量は変化しない（$c^2n^2 \in O(n^2)$）。しかし例えば 2^n のオーダーで動くアルゴリズムでは、n を cn で置き換えると計算量は $2^{cn} = (2^c)^n$ となる。これは 2^n とは等しくない（もちろん、$c = 1$ の場合を除く）。

積

$$f_1(n)f_2(n) = O(g_1(n)g_2(n))$$

和

$$O(f(n)) + O(g(n)) = O(\max\{|f(n)|, |g(n)|\})$$

定数倍

$$O(kg(n)) = O(g(n)), \quad k \neq 0$$

記法	意味	定義	ε-δ記法による言い換え
$f(n) \in O(g(n))$	漸近的に上に有界	$\overline{\lim_{n\to\infty}} \left\vert\frac{f(n)}{g(n)}\right\vert < \infty$	$\exists M > 0$, $\exists x_0$ such that $\vert f(x)\vert \leq M\vert g(x)\vert$ for all $x > x_0$.
$f(n) \in \Omega(g(n))$	漸近的に下に有界	$\varliminf_{n\to\infty} \left\vert\frac{f(n)}{g(n)}\right\vert > 0$	$\exists M > 0$, $\exists x_0$ such that $\vert f(x)\vert \geq M\vert g(x)\vert$ for all $x > x_0$.
$f(n) \in \Theta(g(n))$	漸近的に両側に有界で、**上下界が一致**	$0 < \varliminf_{n\to\infty} \left\vert\frac{f(n)}{g(n)}\right\vert \leq \overline{\lim_{n\to\infty}} \left\vert\frac{f(n)}{g(n)}\right\vert < \infty$	$f(n) \in O(g(n))$ かつ $f(n) \in \Omega(g(n))$
$f(n) \in o(g(n))$	漸近的に消える	$\lim_{n\to\infty} \left\vert\frac{f(n)}{g(n)}\right\vert = 0$	$\forall M > 0$, $\exists x_0$ such that $\vert f(x)\vert < M\vert g(x)\vert$ for all $x > x_0$.
$f(n) \in \omega(g(n))$	漸近支配的	$\lim_{n\to\infty} \left\vert\frac{f(n)}{g(n)}\right\vert = \infty$	$\forall M > 0$, $\exists x_0$ such that $\vert f(x)\vert > M\vert g(x)\vert$ for all $x > x_0$.

演習問題

演習問題 (1)

(1) $f(n)$ と $g(n)$ を漸近的に非負の関数とする。 Θ 記法の基本的な定義を用いて,$\max(f(n), g(n)) = \Theta(f(n) + g(n))$ であることを証明せよ

(2) 任意の実定数 a と b、ただし $b > 0$ にたいして $(n + a)^b = \theta(n^b)$ であることを示せ

(3) $2^{n+1} = O(2^n)$ は成り立つか？ また、$2^{2n} = \mathrm{O}(2^n)$ はどうか？

(4) $o(g(n)) \cap \omega(g(n))$ は空集合であることを証明せよ

Definition 2.3（無限小）

$f(x)$ を $a \in \mathbf{R} \cap \{\infty\} \cap \{\infty\}$ の近傍で定義された関数とする

$$\lim_{x \to a} f(x) = 0$$

であるとき、$f(x)$ は $x \to a$ のとき、無限小であるという

Definition 2.4（高位の無限小と o 記号）

$f(x)$, $g(x)$ は $x \to a$ のとき、無限小であると仮定する。

$$\lim_{x \to a} \frac{f(x)}{g(x)} = 0$$

であるとき、$f(x)$ は $g(x)$ より高位の無限小であるという。それを $f(x) = \mathrm{o}(g(x))$ と書く

$$f(x) = g(x) + \mathrm{o}(h(x))$$
$$\Updownarrow$$
$$f(x) - g(x) = \mathrm{o}(h(x))$$

Definition 2.5（O-記号）

$x = a$ の近傍で

$$|f(x)| \leq C|g(x)|$$

となる $C > 0$ が存在するとき、$f(x) = \mathrm{O}(g(x))$ 書く。ただし、$(x \to a)$

(注) $f(x) = \mathrm{o}(g(x)) \Rightarrow f(x) = \mathrm{O}(g(x))$　　ただし、逆は成立しない。

Definition 2.6（同位の無限小）

$f(x)$, $g(x)$ は $x \to a$ のとき、無限小であると仮定する。

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \ell \neq 0$$

であるとき、 $f(x)$ と $g(x)$ は同位の無限小であるという。
また、$f(x)$ が $(x-a)^{\alpha}$ $(\alpha > 0)$ と同位の無限小であるとき、$f(x)$ は α 位の無限小であるという

Definition 2.6（同位の無限小）

$f(x)$, $g(x)$ は $x \to a$ のとき、無限小であると仮定する。

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \ell \neq 0$$

であるとき、 $f(x)$ と $g(x)$ は同位の無限小であるという。
また、$f(x)$ が $(x-a)^{\alpha}$ $(\alpha > 0)$ と同位の無限小であるとき、$f(x)$ は α 位の無限小であるという

$$x^3 + 2x^2 + x = \mathrm{O}(x)$$
$$x^3 + 2x^2 = \mathrm{o}(x)$$

Definition 2.7（ 無限大 ）

$$\lim_{x \to a} f(x) = \infty$$

であるとき、$f(x)$ は $x \to a$ のとき無限大であるとう。

Definition 2.8（ 低位の無限大と o 記号 ）

$f(x)$, $g(x)$ は $x \to a$ のとき、無限大であると仮定する。

$$\lim_{x \to a} \frac{f(x)}{g(x)} = 0$$

であるとき、$f(x)$ は $g(x)$ より低位の無限大であるという。それを $f(x) = \mathrm{o}(g(x))$ $(x \to a)$ と書く

Definition 2.9（ O 記号 ）

$f(x)$, $g(x)$ は $x \to a$ のとき、無限大であると仮定する。$x = a$ の近傍で

$$|f(x)| \leq C|g(x)|$$

となる $C > 0$ が存在するとき、$f(x) = \mathrm{O}(g(x))$ $(x \to a)$ と書く

Definition 2.10（ 同位の無限小 ）

$f(x)$, $g(x)$ は $x \to a$ のとき、無限大であると仮定する。

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \ell \neq 0$$

であるとき、 $f(x)$ と $g(x)$ は同位の無限大であるという。
また、$f(x)$ が x^{α} $(\alpha > 0)$ と同位の無限大であるとき、$f(x)$ は α 位の無限大であるという

演習問題

演習問題 (2)

$x \to 0$ とするとき、次の問題を示しなさい。

(1) $p \geq q > 0$ ならば $\mathrm{O}(x^p) + \mathrm{O}(x^q) = \mathrm{O}(x^p)$

(2) $p > 0$ ならば $\mathrm{O}(x^p) + \mathrm{o}(x^p) = \mathrm{O}(x^p)$

(3) $p, q \geq 0$ ならば $\mathrm{O}(x^p)\mathrm{O}(x^q) = \mathrm{O}(x^{p+q})$

(4) $p \geq q > 0$ ならば $\mathrm{O}(x^p)/x^q = \mathrm{O}(x^{p-q})$

3 全順序 (total order)

線型順序(linear order)または単純順序(simple order)と呼ばれ、推移的、反対称かつ完全な 二項関係のことである

Definition 3.1 (全順序)

集合 X が関係 $\leq$ によって全順序付けられるとき、X の任意の元 a, b, c に対して、以下の条件を満足する

反対称律 $a \leq b$ かつ $b \leq a$ ならば $a = b$ ⇨ 反対称性によって a < b でも b < a でもあるような不確定な状態は排除

推移律 $a \leq b$ かつ $b \leq c$ ならば $a \leq c$

完全律 (比較可能) $a \leq b$ または $b \leq a$ の何れかが必ず成り立つ

⇩

完全性を持つ関係は、その集 合の任意の二元がその関係で比較可能

Definition 3.2 (狭義全順序)

任意の (広義) 全順序関係 $\leq$ に対し、それに付随する非対称 (従って非反射的) な狭義全順序 (strict total order) と呼ばれる関係 $<$ が存在する。次の互いに同値な二種類の仕方が成立する。

- $a < b \Leftrightarrow a \leq b$ かつ $a \neq b$
- $a < b \Leftrightarrow b \leq a$ でない

ただし、関係 $<$ が $\leq$ の補関係の逆関係である。

性質

- **推移律**：$a < b$ かつ $b < c$ ならば $a < c$.
- **三分律**：$a < b$ または $b < a$ または $a = b$ の何れか一つのみが成立する.
- 恒等性を付随する同値関係とする**狭義弱順序**である。

Definition 3.3（全順序）

推移的かつ三分的な二項関係 $<$ が最初に与えられたとき、そこから (広義の) 全順序 $\leq$ を定めることも、次の同値な二種類の方法が成立する

- $a \leq b \Leftrightarrow a < b$ または $a = b$
- $a \leq b \Leftrightarrow b < a$ でない

鎖　全順序集合のことを**鎖**と呼ぶこともある

Definition 3.4（束）

任意の a, b に対して

$$\{a \vee b, a \wedge b\} = \{a, b\}$$

が成り立つものとして、$a \leq b \Leftrightarrow a = a \wedge b$ と定義するのである。

これにより、全順序集合は分配束になる。

Definition 3.5（順序位相）

任意の全順序集合 $\mathbf{X}$ に対して、開区間は以下で定義する

$$\begin{aligned}(a,b) &= \{x : a < x \text{ かつ } x < b\}\\ (-\infty, b) &= \{x : x < b\},\\ (a,\infty) &= \{x : a < x\},\\ (-\infty,\infty) &= \mathbf{X}\end{aligned}$$

(集合 X が完備となるような順序位相の性質)

- X 上の順序位相が連結ならば X は完備である。
- X が順序位相に関して連結となる必要十分条件は、それが完備かつ X に「ギャップ」がないことである。
  (X の適当な二点 $a, b(a < b)$ に対して $a < c < b$ を満たす点 c が存在しない)
- X が完備となる必要十分条件は、その順序位相に関する任意の閉有界集合がコンパクトとなることである。

Definition 3.6（被覆）

まず集合 X に対し X の部分集合の族 $\{A_\lambda\}_\lambda$ が

$$X = \bigcup_\lambda A_\lambda$$

を満たすとき、$\{A_\lambda\}_\lambda$ は X を被覆しているいい、特に A_λ が全て開集合であれば、$\{A_\lambda\}_\lambda$ を X の開被覆という。

Definition 3.7（コンパクト）

位相空間 X がコンパクトであるとは次の性質を満たす事を言う：X の任意の開被覆 $\{O_\lambda\}_\lambda$ に対し、$\{O_\lambda\}_\lambda$ の有限な部分族 $\{O_{\lambda_i}\}_{i=1,\dots,n}$ が存在し、$\varnothing_{\lambda_i}\}_{i=1,\dots,n}$ も X を被覆する。

順序の和

二つの順序 $(A_1, \leq_1)$ と $(A_2, \leq_2)$ の非交和と呼ばれる自然な順序 $\leq_+$ が和集合 $A_1 \cup A_2$ 上に定義される。しばしばこれを順序集合の和と呼び、単に $A_1 + A_2$ で表す。

性質

$x, y \in A_1 \cup A_2$ に対し $x \leq_+ y$ は以下の何れかひとつを満足することと定められる

1. $x, y \in A_1$ かつ $x \leq_1 y$
2. $x, y \in A_2$ かつ $x \leq_2 y$
3. $x \in A_1$ かつ $y \in A_2$

性質

全順序付けられた添字集合 $(I, \leq)$ の各元 $i \in I$ に対して全順序集合 $(A_i, \leq_i)$ が対応して、各集合は対ごとに交わらないものとするとき、$\bigcup_i A_i$ 上の自然な全順序が $x, y \in \bigcup_{i \in I} A_i$ に対して $x \leq y$ であるとは、

1. 適当な $i \in I$ について $x \leq_i y$ となるか
2. I 上で $i < j$ なる添字について $x \in A_i$ かつ $y \in A_j$ となること

(順序集合の例)

- 自然数全体の集合 $\mathbb{N}$、整数全体の集合 $\mathbf{Z}$、有理数全体の集合 $\mathbf{Q}$、実数全体の集合 $\mathbf{R}$ は通常の代数的な大小関係によって全順序集合になるが、複素数はそうではない。複素数全体の集合 $\mathbf{C}$ に $\mathbf{R} \times \mathbf{R}$ としての辞書式順序を定めたものは全順序集合であるが、この順序は複素数の乗法とは両立しない。
- 自然数全体の成す集合は整除関係を順序として半順序集合である。
- ある集合の冪集合は部分集合の包含関係を順序として半順序集合となる。これは一般には全順序ではない。

  (例) $\{1,2,3\}$ の冪集合 $\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$ について、$\{1,2\}$ と $\{2,3\}$ は比較不能 ($\{1,2\} \leq \{2,3\}$ でも $\{1,2\} \geq \{2,3\}$ でもない)
- ベクトル空間の部分空間全体は包含関係で順序付けられた半順序集合である。
- 半順序集合 P に対し、P の元の列全体の成す集合は、列 a, b に対し、

$$a = (a_n)_{n \in \mathbb{N}} \leq b = (b_n)_{n \in \mathbb{N}} \iff a_n \leq b_n (\forall n \in \mathbb{N})$$

  と定めると半順序集合となる。
- 集合 X と半順序集合 P に対し、X から P への写像全体の成す写像空間は、ふたつの写像 f, g に対して、$f \leq g$ を X の任意の元 x に対して $f(x) \leq g(x)$ となることとして定義すると、半順序集合になる。
- 非循環有向グラフの頂点集合は、到達不可能性によって順序付けられる。
- 半順序集合における順序関係の向きが $a < b > c < d...$ というように交互に入れ替わる列をフェンスと呼ぶ。

4 リスト (List)

リストは、順序つきのデータコンテナとして定義される。
型のないミュータブルなリストはコンストラクタと 4 つの操作によって特徴付けられる。

- 空のリストを作るコンストラクタ
- リストが空かどうかを確かめる操作
- リストの先頭に要素を追加する操作（Lisp の cons）
- リストの先頭要素 (”head”) を求める操作（Lisp の car）
- リストの先頭を除く部分リスト (”tail”) を求める操作（Lisp の cdr）

リストのプロパティ

リストは以下の性質を有する

- リストのサイズ
- リストの等価性 (2 つのリストが等しいとは、2 つが同じオブジェクトであるということでありかつその場合に限る)
- リストの型付け (リストの要素はリストの型と互換の型を持たなければならない。リストが配列を用いて実装されているときは型付けされている)
- リスト中のすべての要素はインデックスを持つ (最初の要素はインデックス 0 を持つ。続く要素は前の要素より 1 大きいインデックスを持つ。最後の要素は (頭のインデックス) + (リストのサイズ) - 1 というインデックスを持つ)

4.1 線形リスト

片方向リスト (singly-linked list)

最も単純な連結リストであり、ノード毎に 1 つのリンクを持つ。リンクはリスト上の次のノードを指し、リストの最後尾なら Null 値を格納するか、空のリストを指す。

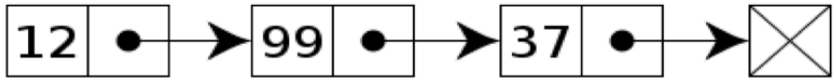

図 4.1　3 つの整数値を格納した片方向リスト

双方向リスト (doubly-linked list)

各ノードには 2 つのリンクがあり、1 つが次のノード（前方リンク）、もう 1 つが後ろのノード（後方リンク）を指す。リストの先頭のノードには後ろのノードがないので、後方リンクにはヌル値を格納するか、空のリストを指す。リストの最後尾のノードには次のノードがないので、前方リンクにはヌル値を格納するか、空のリストを指す。

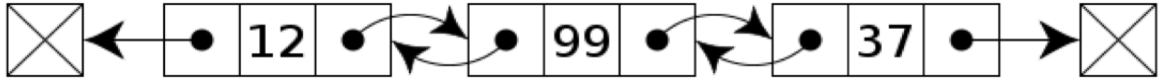

図 4.2　3 つの整数値を格納した双方向リスト

4.2 循環リスト

循環リスト（circularly-linked list）では、先頭と最後尾のノードを相互に連結する。循環リストには片方向、双方向のものがある。循環リストを辿る場合、任意のノードから出発して、好きな方向にたどっていき、最初のノードに戻ってくるまで続ける (循環リストは先頭や最後尾といったものが存在しないリスト)。リスト全体を指すポインタは、アクセスポインタと呼ばれる

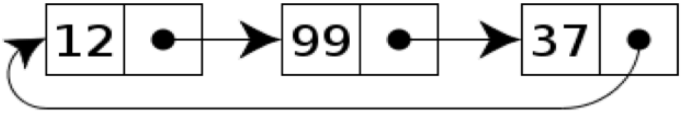

図 4.3　3 つの整数値を格納した循環リスト