コンピュテーショナルシンキング

Computational Thinking

磯辺秀司
小泉英介
静谷啓樹
早川美徳　著

Abstraction

Automation

共立出版

コンピュテーショナル シンキング

Computational Thinking

磯辺秀司
小泉英介
静谷啓樹
早川美徳　著

Abstraction

Automation

共立出版

まえがき

本書はコンピュテーショナル・シンキングの名で呼ばれる問題解決法の実践的教育を趣旨とした，大学教育レベルの教科書である．

コンピュテーショナル・シンキングはまったく新しい概念というわけではない．20 世紀中庸にアルゴリズミック・シンキングとして考案された問題解決法が，21 世紀の現在，特定分野に限定されない万人に必須のスキルとしてリニューアルして再登場したというのが冷静な歴史認識であろう．アルゴリズミック・シンキングは，時代が早すぎたのかもしれない．何でも時代のせいにするのは思考停止に等しいが，計算機科学の流儀で考えて問題を解決するなどというスタイルが社会に受け入れられるためには，いわゆる IT 革命を経てディジタルネイティブ世代が一定の人口を占める現代を待たねばならなかったという筋書きに，ひどく不自然な点は見当たらないように思われる．

計算機科学は抽象化の科学であると言われているが，それに呼応してコンピュテーショナル・シンキングもまた，抽象化のスキルを涵養する．日常生活で遭遇する具体的な問題から出発し，問題を抽象化した先には数学的なモデルがある．そしてそのモデルに基づき，問題解決の自動化・定型作業化手順を記述する言語としてのアルゴリズムが登場する．コンピュテーショナル・シンキングの教育が目指すのはアルゴリズムをプログラミング言語に翻訳して計算機上にインプリメントすることではなく，抽象化とアルゴリズム構築のスキルの涵養である．プログラミング教育が最終目的であるかのような誤解を受ける流れにはしない努力が，教育者に求められている．

そうは言っても，コンピュテーショナル・シンキングの教育の歴史は浅い．言い訳にする意図はないが，教育の現場で様々な試行錯誤が繰り返され，その度にコンピュテーショナル・シンキングの教育理念は外力を受けて少しずつ変容するであろうが，それは一種の成長とも言えるであろう．読者諸氏の厳しいご批判を仰ぎたい．

最後に，本書の執筆にあたって東北大学学務審議会情報基礎委員会から深い理解と激励をいただいたことに感謝するとともに，共立出版の寿日出男氏と中川暢子氏には，企画から出版までの長いもたつきを辛抱していただき，お詫びと紙一重の心からの謝意を表したい．

2016 年 2 月

著者一同

目　次

第 1 章　はじめに　1

1.1　コンピュテーショナル・シンキングとは　1
1.2　具体例　2
1.3　具体例の補足　7
1.4　まとめ　9

第 2 章　準備編 – プログラミングの基礎知識　11

2.1　変数と代入　12
2.2　反復処理　17
2.3　条件判断　24
2.4　配列を使う　28
2.5　手続き (サブルーチン) を使う　33
2.6　再帰を使う　38
2.7　練習問題　43

第 3 章　問題解決事例演習・基本編　47

3.1　利益を増やしたい　47
3.2　コインゲームに勝つにはI　51
3.3　コインゲームに勝つにはII　57
3.4　買い物を計画する　63
3.5　優秀な人材募集　73
3.6　続・優秀な人材募集　79
3.7　焙煎コーヒー売ってます　89
3.8　プラネタリウムカフェへようこそ　103

第 4 章　問題解決事例演習・発展編　109

4.1　カードマジック　109
4.2　ケーキを分けあってみんなで幸せになる　115
4.3　最適な経路を探す　121
4.4　学生を配属する　134
4.5　ディジタルの傷を癒す　145
4.6　自然数の分割　156

第 5 章　情報の表現と文字列の操作　**165**
5.1　情報表現の任意性 …… 165
5.2　文字情報の表現 …… 167
5.3　文字コードの符号化 …… 170
5.4　DNA のテキスト情報の「解読」 …… 172
5.5　文字列の検索 …… 173
5.6　パターンマッチと正規表現 …… 175
5.7　「離れた」視点からテキストを解釈する …… 177

付録 A　情報数学の基本用語　**189**
A.1　集合 …… 189
A.2　写像 …… 191
A.3　行列 …… 193
A.4　グラフ …… 196

参考文献　**201**

索　引　**203**

第 1 章　はじめに

1.1　コンピュテーショナル・シンキングとは

Computational Thinking には安定した訳語がまだない．そのような状況で，本書では粗雑にもコンピュテーショナル・シンキングとカタカナで書くこととした．本来の語が透けて見えれば，欧文の文献や教材をかえって調べやすくなるであろうという打算もある．例えば北米ではコンピュテーショナル・シンキングの教育は大学で文系・理系の区別なく実施されており，充実した教科書も出版されている (David D. Riley and Kenny A. Hunt, *Computational Thinking for the Modern Problem Solver*, Chapman and Hall / CRC, 2014)．それどころか，主に STEM 教育においてだが初等中等教育にすら展開されている．これは読み書き算数のスキルと同列の，21 世紀を生きるすべての人間に必須のスキルと位置づけられていることの反映である．

コンピュテーショナル・シンキングをひと言で言うならば，「計算機科学の流儀で考えて問題を解決すること」となるであろう．教育や研修の文脈では，このスキルの獲得が目標となる．以下，この標語的な表現の主旨を少し詳しく説明したい．

まず「計算機科学」(コンピュータ・サイエンス) という言葉だが，これはとても誤解を受けやすいので補足が必要である．計算機科学はコンピュータそのものに関する科学を指すわけではない．コンピュータの出現に伴って重要性が再認識された既存の学問分野や，新たに生まれた様々な専門領域に対する包括語的な言葉であり，実際それが包括する範囲も広い．例えば，理論計算機科学は計算機科学の一分野であるが，数学の一分野でもある．計算できるとは何か，計算が難しいとは何かという根本的な問題と向き合う学問として数学基礎論に源流を持ち，コンピュータという機械が現れる以前から深い研究がなされている．

次に，「問題解決」の意味であるが，これは入学試験のように与えられた問題を解く行為を指すものではもちろんない．問題解決とは，何らかの目標を達成するための方策を模索して解を発見するまでの思考全般を指す．問題解決のための方法論としては様々なものが知られているが，コンピュテーショナル・シンキングでは，その方法を計算機科学の流儀に依拠しようというものである．

それでは，「計算機科学の流儀」とは何か．それはコンピュータを使ったりプログラムを書くことではない．むしろ，そのようなことは行わない場合のほうが圧倒的である．計算機科学の流儀とは，計算機科学分野に広く見られる知的活動のスタイルであって，現実世界の問題を抽象化によって分析し，それに基づき，問題を解決するための手順 (アルゴリズム) を構築するという二段階の思

考活動からなる．特に後者は，問題解決のプロセスを定型化・自動化することが目的である．コンピュータにそのプロセスを任せられるようアルゴリズムをプログラムに変換することは，定型化・自動化の先にある一種のオプションである．これは理解を深めるために強調すべきことであるが，オプションはあくまでオプションであって，プログラミングというステップはコンピュテーショナル・シンキングには含まれない．要するに，**抽象化 (abstraction)** と**自動化 (automation)** という「**2つの A**」がコンピュテーショナル・シンキングの核心である．

コンピュテーショナル・シンキングの語と概念は Wing [25, 26] の提唱により広く知られるようになった．21世紀初頭のことである．しかし Denning [21] が指摘しているように，まったく新しい概念というわけではなく，1950年代から60年代に Algorithmic Thinking (アルゴリズミック・シンキング) と呼ばれていたものの現代版と位置づけられている．日本でも2003年から，コンピュテーショナル・シンキングの教育が目指すものと本質的に重なる教育が高等学校の情報科目において組織的に開始され，2013年に改訂版が実施された後も，強化された教育内容で継続されている．そこでこの際，その教育をコンピュテーショナル・シンキング教育と呼んでもひどく的外れではないように見受けられる．しかし，歴史的に後発の言葉によってすでに実施されている教育を一方的に再定義するのは，高い理念でその教育を企画した人々に対する不遜な態度と言わざるを得ないし，そのような意図も皆無である．したがって本書では，高等学校における情報教育の内容に寄りかかった説明はしない．

1.2　具体例

コンピュテーショナル・シンキングとは，計算機科学の流儀による現代的な問題解決の方法であって，抽象化とそれに続く自動化という「2つの A」が思考スタイルの基本であることがわかった．実際，「計算機科学は抽象化の科学である」[18] と言い切る人々もいる．

ここで，具体例をいくつか見てみよう．方法論の詳細を説明するよりは，コンピュテーショナル・シンキングの実像を的確に把握する助けになるであろう．最初の例は比較的簡単に抽象化できて，自動化も簡単なもの．次は努力すると抽象化できて，自動化にも努力を要するもの．最後のものは簡単に抽象化できるが，効率的な自動化が難しいもの，となっている．

1.2.1　問題 1: 互選の当確ライン

例題 1.1. クラスで合コンの企画と準備を担当する合コン委員3人を投票で選ぶことになった．これを聞いて僕は，この委員になるためにこの大学に入ったのだと思った．むしろ，そのために生まれてきたに違いないとさえ…
1人1票で上位3人が当選，クラスは45人．どうすれば当選できるのか？

抽象化　法令・学則に基づく選挙ではないとはいえ，とりあえず定められた投票以外の非常識な方法は考えないことにする．この問題の本質を抽出して一般化すると，「N 人の集団で互選を行う．1人1票を投票し，上位 r 人を選出する．少なくとも何票を獲得すると当選確実か？」となるであろ

う．これを解く手順を考えることになる (以下の基本アイデアは文献 [4] による).

得票数を上位から下位へ順に $x_1 \geq x_2 \geq \cdots \geq x_n > 0$ とすれば,

$$x_1 + x_2 + \cdots + x_n = N.$$

第 r 位 (最下位) で当選確実になるためには,

$$x_{r+1} + x_{r+2} + \cdots + x_n < x_r$$

であればよい．これは，$r+1$ 位以下の支持者が結託しても r 位の票に及ばないことを意味しているからである (図 1.1 参照).

1	2	$\cdots$	r	$r+1$	$\cdots$	n
x_1	x_2	$\cdots$	x_r	x_{r+1}	$\cdots$	x_n

図 1.1　順位 (上段) と得票状況

ところで,

$$\begin{aligned} x_{r+1} + x_{r+2} + \cdots + x_n &= N - x_1 - x_2 - \cdots - x_r \\ &\leq N - rx_r \end{aligned}$$

であるから，$x_{r+1} + x_{r+2} + \cdots + x_n < x_r$ の条件により,

$$N - rx_r < x_r$$

であれば当選に十分．よって，$x_r > \dfrac{N}{r+1}$ となる票を取ればよい.

自動化 自動化とは処理の定型化であって，必ずしもプログラミングのことではない．自動化の実態はアルゴリズムの記述である．ここでアルゴリズムとは，何らかの入力（インスタンスとも言う）に対して所望の出力を得るまでの処理手順を指す用語である．そのアルゴリズムは何らかの方法で表現して記録に残さなければ他の人に伝わらないし，自分でも忘れてしまうかもしれない．そこで，一つでもプログラミング言語に慣れている人はそれを使って書いてもよいが，そのような言語を知らない人は，とにかく何をどういう順番でどうするのかを自然言語 (日本語，英語など) で書けばよい．それも立派にアルゴリズムの表現である.

上記の問題に対する処理手順は極めて単純で，入力に対して出力すべきものは明確である.

(1) Input $N, r > 0$.
(2) Find the least $\ell \in \mathbb{Z}$ such that $\ell > N/(r+1)$.
(3) Output ℓ.

もともとの問題に戻ってこの手順を適用すると，入力 $N = 45, r = 3$ であるから，$45/(3+1) = 11.25$ により，$\ell = 12$ となる．つまり，自分を除いて 11 人以上の支持者がいれば当確となり，大学に入学した意義も，この世に生を受けた意味も実感できる.

1.2.2　問題 2: 少人数教育科目のクラス分け

例題 1.2. 某大学の少人数教育科目は1年生対象の必修科目で，並列に150クラスが提供されている．学生は受講したいクラスを第1希望から第5希望まで提出するが，各クラスの定員は20人程度に設定されており，学生は一つのクラスしか履修できない．1学年2400人の学生ができるだけ満足できるクラス分けをしたい．どのようにすればよいか？

抽象化 これは大学教育の現場でしばしば直面する問題である．しかし，あっさり解けるような問題ではないのは明らかで，既知の研究成果に基づくいくつかの道具を要する．以下，文献 [8] の秀逸な解説に沿って抽象化の手法を紹介する．

まず，状況を改めて確認しよう．クラス数を L とし，各クラスは 1 から L の番号で識別されるものとする．また，各クラスの定員を c_i $(1 \leq i \leq L)$ とする．全学生数を N とし，各学生は 1 から N の番号で識別されるものとする．各学生は L クラスの中から第 k (≥ 1) 希望まで提出するが，クラス分けにより一つのクラスしか履修できない．このとき，クラス分けの各パターンに対する評価 (もしくは満足度) を，希望どおりの学生が多いほど高得点になるように定義し，これが最大値となるようなクラス分けを探ることにする．

ここで，変数 x_{ij} を導入する．これは i と j の組み合わせによって 1 か 0 の値をとる．すなわち，クラス i を学生 j が履修するとき $x_{ij} = 1$, それ以外は $x_{ij} = 0$ とする．縦軸をクラス，横軸を学生とすると，図1.2のような L 行 N 列の表 (値は 1 または 0) を考えていることになる．

この変数を使うと，「条件 1: 各クラスには定員が設けられている」は，一つの行について和をとると定員以下になっていればよいので，

$$\sum_{j=1}^{N} x_{ij} \leq c_i$$

がすべてのクラス i $(1 \leq i \leq L)$ について成立することと表現される．また，「条件 2: 各学生は一つのクラスしか履修できない」は，一つの列について和をとると正確に 1 になることを意味するので，

$$\sum_{i=1}^{L} x_{ij} = 1$$

がすべての学生 j $(1 \leq j \leq N)$ について成立することと表現される．

次に，学生の満足度を表現するための評価尺度を導入する．学生 j がクラス i にクラス分けされたときの満足度を表す値を p_{ij} としよう．例えば，第3希望まで提出する場合には次のように設定することができる．

$$p_{ij} = \begin{cases} 100 & \text{学生 } j \text{ の第1希望がクラス } i \text{ のとき} \\ 40 & \text{学生 } j \text{ の第2希望がクラス } i \text{ のとき} \\ 1 & \text{学生 } j \text{ の第3希望がクラス } i \text{ のとき} \\ -10^6 & \text{学生 } j \text{ の希望ではないとき} \end{cases}$$

	1	2	$\cdots$	j	$\cdots$	N
1	x_{11}	x_{12}	$\cdots$	x_{1j}	$\cdots$	x_{1N}
2	x_{21}	x_{22}	$\cdots$	x_{2j}	$\cdots$	x_{2N}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
i	x_{i1}	x_{i2}	$\cdots$	x_{ij}	$\cdots$	x_{iN}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
L	x_{L1}	x_{L2}	$\cdots$	x_{Lj}	$\cdots$	x_{LN}

図 1.2　クラス (1 行から L 行) と学生 (1 列から N 列) の対応表

これらの値はあくまで一例であって，100, 40, 1, -10^6 はそれぞれ「よかった!」,「ま，いいか」,「なんだかなぁ」,「ざけんな!」という気持ちに対応する値を感覚的に割り当てたものである．実際，100 が 1000 でも構わないし，-10^6 は本当は $-\infty$ にしたいくらいなのだが，無限大は値ではなく概念的なものなので，適当に大きい負の値を割り振っている．

とにかくこれにより，クラス分けに対する学生全体の満足度 z を次のように表現できる．

$$z = \sum_{i=1}^{L} \sum_{j=1}^{N} p_{ij} x_{ij}$$

この z (目的関数と呼ばれる) を最大にする x_{ij} $(1 \leq i \leq L, 1 \leq j \leq N)$ を条件 1 と条件 2 の制約のもとに求めればよい．

自動化 以上のように，なすべきことは目的関数 z の最大化である．

(1) Input L, N, c_i, p_{ij} $(1 \leq i \leq L, 1 \leq j \leq N)$.

(2) Find $x_{ij} \in \{0, 1\}$ that maximizes $z = \displaystyle\sum_{i=1}^{L} \sum_{j=1}^{N} p_{ij} x_{ij}$

subject to $\forall i \quad \displaystyle\sum_{j=1}^{N} x_{ij} \leq c_i$ and $\forall j \quad \displaystyle\sum_{i=1}^{L} x_{ij} = 1$.

(3) Output x_{ij} $(1 \leq i \leq L, 1 \leq j \leq N)$.

ステップ 2 で，最大となるような x_{ij} を求めるとは書いたが，これはそれほど簡単なことではなく，一般には総当たり (しらみつぶし) と本質的に変わらない求め方しかない．ここで，もとの状況に戻ってよく考えれば，$L = 150$, $N = 2400$ であるから，$x_{ij} \in \{0, 1\}$ の割り当てパターンは，条件 1(クラス定員) を無視した単純見積りで 150^{2400} 通りとなる．これは概ね 4×10^{5222} 通りにもなり，この宇宙のビッグバンと同時に 1 テラ通り/秒のペースでチェックを始めたとしても，現在なおしらみつぶしは終了していないどころか，ほんの始まったばかり，という果てしないことになる．

実はこれは，$x_{ij} \in \{0,1\}$ と制限していることによる．そこで，$0 \leq x_{ij} \leq 1$ と制限を緩和すると「線形計画問題」と呼ばれるよく知られた問題となり，効率的な (総当たりではない) アルゴリズムも知られているので，それを使えばよい．特に本問題の場合，制限を緩和して線形計画問題のアルゴリズムで得た解 x_{ij} が，都合よく 0 または 1 になることが知られている [8].

1.2.3　問題 3: 補助貨幣の大改革

例題 1.3. 素数が大好きな首相が，現在の補助貨幣 1 円，5 円，10 円，50 円，100 円，500 円を廃止して，デザインも新たに 3 円，7 円，13 円，59 円，107 円，503 円の硬貨による補助貨幣体系に変更すると言い出した．私はスーパーのレジをバイトで担当しているのだが，おつりがうまく支払えるかとても不安．もしかして，硬貨だけでは支払えない金額があるのでは??

抽象化 この問題には自明な解がある．それは，1 円と 2 円である．したがって，硬貨だけで支払えない金額は確かに存在するのだが，他にも払えない金額があるかもしれない．それを問題にしている．だいたい，2 も素数なのに仲間外れにするなんて，かわいそうである．一方，この金額は硬貨だけで払えるだろうかと考え込んでしまうことは，不安ではなくむしろ知的な楽しみであることが指摘されており [19], その観点からはそう悪い体系というわけでもない．以下，文献 [19] の明快な解説に沿って紹介しよう．

まず，問題の本質を抽象化する．正整数の有限集合 $A = \{a_1, a_2, \ldots, a_d\}$ を一つ固定し，A には共通因子がないとする (すなわち $\gcd(a_1, a_2, \ldots, a_d) = 1$). このとき，非負整数 n が A で表現可能であるとは，ある非負整数 $m_1, m_2, \ldots, m_d$ が存在し，

$$n = m_1 a_1 + m_2 a_2 + \cdots + m_d a_d$$

であることと定義する．本問題は，与えられた n (≥ 0) が A で表現可能かどうか判定することである．

自動化 A を固定して任意の n が与えられたとき，

$$n = m_1 a_1 + m_2 a_2 + \cdots + m_d a_d$$

を満たす非負整数 m_i $(1 \leq i \leq d)$ を効率的に求める方法は，実は知られていない．

注意してほしいのは，m_i $(1 \leq i \leq d)$ が非負整数ではなく単に整数 (負の整数を含む) であるならば，話はまったく違う点である．古典的な一次不定方程式の研究で知られている事実であるが，m_i が存在するための必要十分条件は $\gcd(a_1, a_2, \ldots, a_d)$ が n を割り切ることである．この問題の場合，$a_1, a_2, \ldots, a_d$ はすべて素数なので，$\gcd(a_1, a_2, \ldots, a_d) = 1$ であり，常に n を割り切る．したがって，m_i は整数の範囲に必ず存在する．しかしながら，そのような m_i が実際に求められたとしても，例えば 3 円玉を -2 枚と 7 円玉 1 枚を合わせて 1 円を支払うことは，普通の人にはできない．

一方，A で表現可能ではない整数は有限個しか存在しないこともわかっている．その中で最大のものを $g(a_1, a_2, \ldots, a_d)$ で表すことにすると，$d = 2$ については $g(a_1, a_2) = a_1a_2 - a_1 - a_2$ という事実が知られているが，$d \geq 3$ については難しい状況にある．

要するに，そのつど計算して判定するスタイルでの処理の定型化は，できることはできるが，総当たり的なことを試すのが精一杯である．効率的な自動化などを望むのは贅沢なことであって，現状ではどうにもならない (どうにかできるなら，それは素晴らしい発見である).

なお，別のスタイルによる処理の定型化を考えたとき，「A で表現できない n は有限個」という事実は現実的な対応の重要なヒントになる．すなわち，A では表現できない n の一覧表を作り (表の面積も有限である), レジの脇に張り出しておくことが考えられる．硬貨だけで払えるおつりかどうかは，この表を参照すればよい．しかしながら，A の中身によっては，一覧表は非常に大きな有限である可能性もあり，一般にはこれで効率的に処理の定型化ができるわけではない．通常，非常に大きな有限は無限より扱いにくい．

1.3　具体例の補足

1.3.1　抽象化の道具

これらの例では，具体的な問題を数論ないし代数学の基本言語に対応させ，数式で表現して抽象化を行った．このほか，日常に見出される問題を抽象化するときによく現れるものがある．それはグラフである．この場合のグラフとは棒や円や折れ線のことではなく，頂点の集合と，それらの結びつきの様子を表す辺の集合で定義される概念である．一般にグラフ G は，頂点集合 $V = \{v_1, \ldots, v_n\}$ と辺集合 $E = \{(v_i, v_j)\} \subseteq V \times V$ を使って $G = (V, E)$ と表される．辺に向きをつけたり，重みをつけることもある．グラフの性質に関する事実を解明する科学はグラフ理論と呼ばれており，計算機科学や数学の一分野となっている．

グラフは様々な問題の抽象化に際して最も基本的な道具として普通に使われており，高等学校の情報科目の教科書にも取り上げられているのだが，言葉の準備が必要なので前節の具体例には含めなかった．例えば，地理的に離れたいくつかの地点をトラックが最短経路で回って元の地点に戻る問題や，同様の状況で，最短ではないかもしれないが経路のガソリン代も勘案して最経済経路を考える問題などはどれもグラフで抽象化できて，グラフ理論の成果を適宜，適用することで処理の自動化がなされる．ただし自動化とは言っても，本質的に総当たりでしか対応できないような事態も含まれる．

問題解決の話とは少し離れるが，蕎麦屋・寿司屋など出前をする業界では昔から，複数の客を回るときには「出前は近くから，(丼・桶などの) 回収は遠くから」という知恵が引き継がれてきた．この知恵の妥当性は直観的に明らかではあるが，グラフでモデル化してみると上記の最経済経路の問題に還元され，論理的に検証できる．

1.3.2　計算の難しさ

具体例の中で，総当たりの計算が意外に時間を要することに触れた．総当たりは，ほとんど何も

工夫をしていないアルゴリズムである．ビッグバンの時点から計算を開始しても終わらないケースが現れたように，効率的ではない (もちろん，効率がすべてであるかのような原理主義的な価値判断をする意図はない).

例えば，与えられた自然数 n が素数であるかどうかを判定する問題を考える．一つの解決策として，n を素因数分解するという方法がある．これは，n を割り切る因数 d $(1 < d < n)$ を見つけようというもので，実際に見つかれば素数ではなく，見つからなければ素数と判定できる．しかし，素因数分解は難しく，計算に非常に時間を要する．何しろこれが難しいお陰で現代暗号が成立し，インターネット上での認証や秘密通信などが機能しているくらいである．計算時間の見積は，総当たりよりはひどくないが，総当たりに近い「準指数関数時間」と呼ばれる量になっている．何の関数かというと，n を2進数で表したときの長さ (ビット長) の準指数関数である．

しかし，非常に洗練されたアルゴリズムを用いると，「多項式時間」で確定的に判定できる．多項式とは n のビット長の多項式，確定的とは判定に誤りがないという意味である (乱数を使う別のアルゴリズムもあり，その場合は判定に間違いが発生する可能性が僅かにある). n を大きくすると，指数関数はどんな多項式より大きくなるので，多項式時間のアルゴリズムのほうが高速アルゴリズムと評価できる．

コンピュテーショナル・シンキングでは，アルゴリズムの独自開発までは求めていないが，採用するアルゴリズムによって自動化の結果としての効率は差が出ることになる．ただし，繰り返しになるが，効率がよくないことがよくない自動化という訳ではない．

このように問題の難しさを研究する学問は，計算量理論と呼ばれる計算機科学の一分野である．その最高峰の未解決問題は「P≠NP 予想」である．これは直観的に言うと，本質的に総当たりと同じ方法しか解法が見つかっていない問題の集合 (NP) と，多項式時間で効率的に解ける問題の集合 (P) は一致していない，ということを証明しようというものである．もし一致しているなら何でも効率的に解けてしまうが，多くの計算機科学者は一致していないと予想している．現在なお，研究が続けられている．

1.3.3　連続と離散

グラフの説明で紹介した具体例も含めて，それぞれ異なる文脈のもとで問題として認識され，また抽象化の結果も処理手順の自動化の難しさにも個性があるが，どれも離散的な抽象化がなされている点が共通である．離散的とは，実数に対する整数のように，連続するものではなく不連続なとびとびの概念を指す．例えば問題1から問題3の自動化で，有限個の入力は連続量ではなく，すべて離散的な量となっている．実は計算機科学は離散的なモデルが得意である．まあ，ディジタル社会だからね，というのは因果律を無視した転倒した観測で，離散が得意な計算機科学の成果が社会に織り込まれた結果がディジタル社会である．

なお私たちの日常生活が，連続する実数よりは離散的な整数のほうに密着しているかというと，それはよくわからない．例えば体重は，実数軸上を連続して上下する．その意味で実数と密着しているが，観測される体重は多くの場合，有理数 (整数の比) である．文献 [6] が指摘するように，「私の体重は有理数か無理数か心配で」と悩む人は多くはない．

ただし，ビッグデータを扱うデータ科学の分野など，問題の抽象化や解決に統計量が関係すると，たとえ計算機科学の流儀でといっても，整数や有理数の世界からはみ出すことが多い．標準偏差を求めるのに平方根に使われたり，確率密度関数の積分などが現れるからである．このように，統計量が深く関係する問題を計算機科学の流儀で解決する一例として，異常検知技術と著者判定技術を，この補足の最後に簡単に触れておこう．

異常検知とは，日常のパターンを知った上で，そこから逸脱するデータを見出すことである．例えば，ネットワークに接続されたサーバのセキュリティを向上させたい，という問題を考える (目標の達成を検討するのだから確かに問題解決である). このとき，まず各利用者の日常行動を十分に学習し，特徴を把握する．その上で，普段はログインしない時間にログインする利用者を見つけたり，普段は見向きもしないアプリケーションに急に興味を示す利用者が現れるなど，いわゆる「外れ値」を検出した場合，それが統計的に確かに異常と確信されるレベルのときに侵入警報が出るようにすれば，セキュリティは向上する (詳しくは文献 [15]).

別の例として挙げた著者判定とは計量文献学のテクニックである．計量文献学は 19 世紀に成立した学問であるが，現代では分析の科学と技術を通じて計算機科学と共鳴している．例えば，ある人物 A の著書 X が本当は別の人物 B が書いたものではないかと疑われる場合，これを決着したいという問題を考える．このとき，A の著書 X に記されている言葉の多角的な統計量を計測し，B の論文や著書の文章も同様に計測し，両者の統計的距離を検討することになる．シェークスピアとフランシス・ベーコンは同一人物かどうかなどの (統計的) 決着に応用されている [14].

1.4　まとめ

これまで概観したように，コンピュテーショナル・シンキングとは，計算機科学の流儀による問題解決であって，問題の抽象化と処理の自動化という「2 つの A」と呼ばれるステップからなる．そのスキルは 21 世紀を生きるすべての人に必須と考えられており，児童から学生まで広範囲に教育が展開されている．

文献 [26] で指摘されているように，コンピュテーショナル・シンキングは現実世界の問題を抽象化して解決を目指すという意味で，数学的思考 (Mathematical Thinking)[22] と相当に重なっている．ただ，コンピュテーショナル・シンキングは自動化の段階でアルゴリズムを導入し，数学的な解決を手順記述という形にして別の洗練を図っている点が特徴的である．

本書では，日常に取材した様々な問題を取り上げ，それに対して「2 つの A」を施す．その意味でコンピュテーショナル・シンキングの事例集を兼ねた演習書と言える．ケーススタディの利点は，多くの事例に接して考えることで，帰納的な一般化・普遍化の能力が開発される点である．本書を通じてコンピュテーショナル・シンキングの基本スキルが獲得されると同時に，本書を離れて新たな問題に直面したとき，すでに獲得している普遍をその問題に速やかに適用できるスキルも自然に育まれているはずである．