2020 年 8 月 18 日

『情報科学概論（データサイエンス大系）』

（田中琢真 著，学術図書出版社）

章末問題略解

第 1 章

1-1 内閣府の消費動向調査で公表されている主要耐久消費財等の普及率を利用するとよい．

1-2 消費動向調査を使ってもよいが，新聞の全文検索データベースを利用し，「コンピュータ」や「情報技術」などの語の使用頻度の推移を調べてもよい．

1-3 様々な解答がありうる．たとえば，医療や流通網の発達が著しい．医療機器の発達は電子機器やコンピュータの発達によるところも大きいが，公衆衛生の向上によるところも大きい．流通網の発達は内燃機関の進歩（自動車の普及）によるところが大きいが近年は電子化でより効率的な配送が可能になった．

1-4 略．

1-5 縫い針は⓪単純な道具，足踏みミシンは針を刺したり抜いたりする①複合的な道具，電動ミシンは③繰り返し動作する道具だ．近年では様々な刺繍パターンを記憶させられるミシンも普及している．これは④動作が変わる道具にあたる．

1-6 蒸気機関の原理は古代にすでに発明されていたが，労働力の供給が逼迫していなかったので実用化されなかったとされる．アーサー・オードヒューム『永久運動の夢』（ちくま学芸文庫，2014），三輪修三『工学の歴史—機械工学を中心に』（ちくま学芸文庫，2012）などを参照するとよい．

1-7 ネットで “Jacquard loom” を検索すると海外の博物館が製作した動作原理の解説動画がみつかるので参照せよ．ジャカード織機は現代のコンピュータと違い，分岐はできない．

第 2 章

2-1 この話題についての古典的名著，吉田洋一『零の発見—数学の生い立ち』（岩波新書，1986）などを参照するとよい．

2-2 $10011100010000_{(2)} = 23420_{(8)} = 2710_{(16)} = 10000$.

2-3 $11100100_{(2)}$, $100011000_{(2)}$, $11.1111_{(2)}$, $56_{(8)}$

2-4 $100 = 01100100_{(2)}$ なので，各桁の値を反転して 1 を足すと $10011100_{(2)}$ と $1111111110011100_{(2)}$ になる．符号なし整数ではそれぞれ 156 と 65436.

2-5 奇数番目の数の和+チェックディジットを a，偶数番目の数の和を b として，$a + 3b \equiv 0 \pmod{10}$ だ．1 箇所を書き間違える場合は 2 種類ある．

①奇数番目の数かチェックディジットを書き間違えた場合（a が $a' \not\equiv a \pmod{10}$ になる）　このとき，$a' + 3b \equiv 0 \pmod{10}$ とすると，$a' \equiv a \pmod{10}$ となる．つまり書き間違えがない場合のみチェックディジットは 10 で割り切れる．

②偶数番目の数を書き間違えた場合（b が $b' \not\equiv b \pmod{10}$ になる）　このとき，$a + 3b' \equiv 0 \pmod{10}$ とすると，$3b' \equiv 3b \pmod{10}$ だ．3 と 10 の最大公約数は 1 なので，p. 27 の脚注 8) を使うと $b' \equiv b \pmod{10}$ となり，書き間違えがない場合のみチェックディジットは 10 で割り切れる．

よって，1 箇所の書き間違えは必ず検出できる．

本書の ISBN を 978-**5**-77**0**6-0702-4 と書き間違えた場合はチェックディジットでは間違いが検出されないから，2 箇所の書き間違えは必ずしも検出できない．

2-6 16 bit 符号付き整数ならば，1970 年 1 月 1 日 0 時 0 分 0 秒から $2^{15} - 1$ 秒までは正確に表せるが，2^{15} 秒つまり 1970 年 1 月 1 日 9 時 6 分 8 秒になると -2^{15} 秒つまり 1969 年 12 月 31 日 14 時 53 分 52 秒に戻る．16 bit 符号なし整数ならば，$2^{16} - 1$ 秒つまり 1970 年 1 月 1 日 18 時 12 分 15 秒までは正確に表され，1 秒後に 1970 年 1 月 1 日 0 時 0 分 0 秒に戻る．

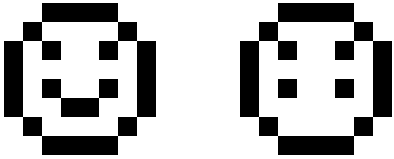
2-7 x の概数 $\tilde{x}$ と y の概数 $\tilde{y}$ を考える．$x = \tilde{x} + \epsilon_x$ かつ $y = \tilde{y} + \epsilon_y$ だとする．$\tilde{x} + \tilde{y}$ と $x + y$ の差（概数計算の誤差）は，$\epsilon_x + \epsilon_y$ だ．$\tilde{x}$ が x を 10 の位で四捨五入したものだとすると，$|\epsilon_x| \leq 50$ で，$\tilde{y}$ が y を 100 の位で

四捨五入したものだとすると，$|\epsilon_y| \leq 500$ だ．このとき，$|\epsilon_x + \epsilon_y|$ は最大で 500 を少し上回り，数百程度になる．つまり，$|\epsilon_x + \epsilon_y|$ は $|\epsilon_x|$ と $|\epsilon_y|$ の大きい方の誤差と同程度になる．要するに，$|\epsilon_x|$ と $|\epsilon_y|$ の片方だけを小さくしても最終的な誤差はほとんど小さくならない．だから，この 2 つが同程度になるように同じ位で四捨五入するのがよい．

$\tilde{x}\tilde{y}$ と xy の差は $\epsilon_x\tilde{y} + \epsilon_y\tilde{x} + \epsilon_x\epsilon_y$ だ．ここで，実際の数と概数の差が小さい，つまり $|\epsilon_x| \ll |\tilde{x}|$ かつ $|\epsilon_y| \ll |\tilde{y}|$ とすると，誤差は $\epsilon_x\tilde{y} + \epsilon_y\tilde{x}$ で近似できる．x の値が数万，y の値が数十万で，x も y も 1000 の位までで四捨五入したとする．すると xy は数十億から数百億の値で，誤差は数万 × 数百 + 数十万 × 数百 となり，第 2 項が大きいので誤差は数千万から数億になる．$|\epsilon_x\tilde{y}|$ と $|\epsilon_y\tilde{x}|$ が同程度になるように上から数えて同じ桁数で四捨五入するのがよい．

浮動小数点数は計算結果の上からある桁までを保持するので，かけ算・割り算の概数計算に適している．そのためかけ算では問題が起こりにくい．一方，足し算・引き算では不適切な場合があり，桁落ちを起こす．

2-8 元の画像と不可逆圧縮した画像は次の図のとおり．圧縮前の 4 bit が圧縮後に 3 bit になるので，容量は 75%になった．



2-9 NOT $X = X$ NAND X，X AND Y = NOT (X NAND Y) = (X NAND Y) NAND (X NAND Y)，X OR Y = NOT ((NOT X) AND (NOT Y)) = (X NAND X) NAND (Y NAND Y)．他の作り方もある．

2-10 (a) X_2 と X_4 を入力とする NOT ゲート 2 つの出力を入力とする AND ゲート a があり，X_1 と X_3 を入力とする AND ゲート b があれば，AND ゲート a と b の出力を入力とする AND ゲート c の出力は題意を満たす．すなわち，(X_1 AND (NOT X_2)) AND (X_3 AND (NOT X_4)) を論理回路にすればよい．

(b) 同様に，ビットパターンの中で 0 である入力は NOT ゲートを通し，1 である入力はそのまま通し，これをすべて AND ゲートで集約すればよい．

(c) $(X_1,\ X_2,\ X_3,\ X_4) = (1,\ 0,\ 1,\ 0)$ のときのみ出力が 1 になり，これら以外の入力では出力が 0 になる論理回路 a と，$(X_1,\ X_2,\ X_3,\ X_4) = (0,\ 1,\ 0,\ 1)$ のときのみ出力が 1 になり，これら以外の入力では出力が 0 になる論理回路 b を作り，この 2 つの出力を OR ゲートへの入力とすればよい．

(d) 以上から明らか．

(e) 以上のやり方で $m = 1$ の場合は目的の論理回路を作れる．m 個の出力それぞれについて同様にすればよい．

2-11 符号列を 7 bit ごとに区切ると

(0,1,0,1,1,0,1)　(1,1,0,0,1,0,1)　(0,1,0,1,1,0,0)　(0,0,1,0,1,0,0)

となる．それぞれ (7,4) ハミング符号でハミング距離が 1 以下のものを見つけると，

(0,1,0,0,1,0,1)　(0,1,0,0,1,0,1)　(0,1,0,1,1,1,0)　(0,0,1,1,1,0,0)

なので，元のビット列は

0,1,0,0,0,1,0,0,0,1,0,1,0,0,1,1

となる．8 bit ずつに区切って 16 進数に直すと，$44_{(16)}$，$53_{(16)}$ となり，ASCII コードの表を参照すると「DS」だ．

第 3 章

3-1 略．

3-2 $2^{10}/10^3 = 1.024$，$2^{80}/10^{24} \approx 1.209$．

3-3 速度と容量と可塑性・揮発性の軸でのまとめは図 3.10 を参照．速度が速く容量が大きいものがあればすでに普及している．よって，現在普及しているストレージは相対的に速度が遅く容量が大きいか，相対的に速度が速く容量が小さいかのいずれかだ．相対的に速度が遅く容量が小さいものは使われない．

3-4 USB A・B・C，ミニ USB A・B，マイクロ USB A・Bがある．ソケットについてもまとめること．

3-5 略．

3-6 (1) フリップフロップ a と b の値から計算した結果をフリップフロップ a に書き込むとする．このとき，フリップフロップ a と b の値がいつでも書き込み可能だとすると，フリップフロップ a と b の値から計算した結果がフリップフロップ a に書き込まれ，その値とフリップフロップ b の値から計算した値がフリップフロップ a に書き込まれ，……と際限なく続く．計算と書き込みを一度で終わらせるためにはクロック入力が 0 から 1 になった瞬間だけ書き込めるのがよい．

(2) 「クロックが 1 のときの入力 D の値を出力 Q の値として保持する」のだと，クロックが 1 の間は際限なく計算と書き込みが続くから．

(3) クロックが 0 から 1 になった瞬間に，左端の D フリップフロップの出力が現在の I の値ではなく，直前にクロックが 0 から 1 になった瞬間の入力 I の値であることが必要だ．また，他の D フリップフロップも 1 クロックサイクルあいは 2 クロックサイクル前の I の値を保持していることが必要だ．つまり，クロックが 0 から 1 になった瞬間の出力が以前から保持している値のままであることが必要だ．

第 4 章

4-1 略．

4-2 地図を写すのには手間がかかるので，手間をかけたくない場合は①が都合がよい．しかし出先で地図に書き込みをしたい場合は原本を汚さないため②が都合がよい．サブルーチンがメインルーチンの変数の値をコピーして使う（**値渡し**）か，同一の変数をそのまま使う（**参照渡し**）かはプログラムの挙動を大きく変える（言語によっては一方しか使えない）．サブルーチンがメインルーチンの変数の値を変更した方が便利か，不便かを考えて選ぶ．

4-3 変数 S の値が 0 に最初に設定されていない（**初期化**されていない）．1 か

ら 9 までの総和を求めている．

4-4 サブルーチン「n で計算」は n の階乗（$1 \times 2 \times \cdots \times n$）を求める．よって，$5! = 120$ を出力する．

4-5 (a) $N/2$ 個と $N/2$ 個に分けて和をとってから足せば速い．

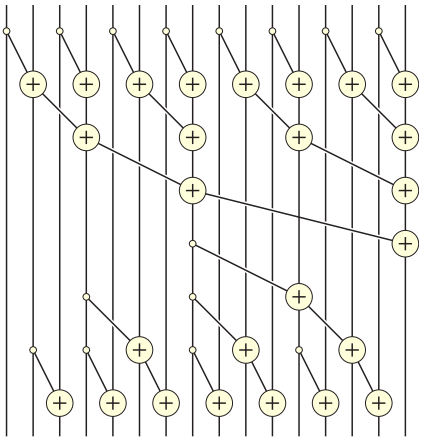
(b) x_i から x_{i+1} を順番に計算しなければならないので，2 人で作業しても速くならない．

(c) 束を 2 つに分けて 2 人で探せば速い．

(d) 1 人で探す場合，1 km の洞窟から探すとすると，歩く距離は 50%の確率で 2 km，50%の確率で 8 km．よって平均 5 km．2 人で探す場合，1 人は 2 km 歩き，もう 1 人は 6 km 歩く．2 km 歩く人は必ず 6 km 歩く人を待つ．よって平均すると 1 人の方が早く終わる．並列計算でも，すべてのプロセスの作業終了を待たねばならないことがある．

(e) 3 つのさいころが同じ目を出すのは珍しいが，2 人で同時に試せば，1 人で試すより早く出ることを見込める．

(f) 一見，2 人で手分けしても駄目そうだが，速くできる方法がある．下図をヒントに考えてみよ．



4-6 $\dfrac{SN}{2} > AN\log_2 N + SB\log_2 N$．$S$ が N より充分大きい場合は，およそ $\dfrac{N}{2}$ と $B\log_2 N$ の比較になり，N も大きいときには不等式が成立

する．

4-7 いずれも中置記法では 1*(2+(3*(4+5))) となるので，29．

4-8 (SI(KK)KK) = (IK(KKK)K) = (KKK) = (K)，
(SI(KK)(KI)K) = (I(KI)(KK(KI))K) = (KIKK) = (IK) = (K)，
(SI(KK)K(KI)) = (IK(KKK)(KI)) = (KK(KI)) = (K)，
(SI(KK)(KI)(KI)) = (I(KI)(KK(KI))(KI)) = (KIK(KI)) = (I(KI)) = (KI)，
なので，x OR y とみなせる．同様に (SS(K(K(KI)))xy) は x AND y だ．さらに，S と K と I を組み合わせて論理演算や条件分岐，ループも表現できる．これを SKI コンビネータ計算という．

4-9 Emacs, vi, Atom, xyzzy などが代表的．テキストエディタは時間をかけて慣れる必要があるが，それだけの価値はある．

第 5 章

5-1 ②．

5-2 略．

5-3 SELECT **日付，商品コード，個数** FROM **表** A は表 A から日付と商品コードと個数の列を抽出した表，SELECT **日付，顧客名** FROM **表** A INNER JOIN **表** C ON **表** A.**顧客** ID=**表** C.**顧客** ID は表 A と表 C を顧客 ID でつきあわせて顧客名を見つけ，日付と顧客名の列を抽出した表．

第 6 章

6-1 1 行目の<h2>が閉じていない．4 行目の「部屋の中の象」が
と
で囲まれていない．5 行目で
を閉じるのが
になっていない．6 行目で<h2>が</h3>で閉じられている．

6-2 画像が表示されないブラウザで画像代わりに表示する．視覚障害者のための説明文となる．検索エンジンなどが画像を自動分類するのに使う．

6-3 以下，クライアントの内蔵時計での時刻を大文字 T，サーバの内蔵時計での時刻を小文字 t で表す．クライアントがサーバにリクエストを送り，

内蔵時計で時刻 T_1 を記録する．リクエストを受け取ったサーバは内蔵時計で時刻 t_2 を記録し，時刻 t_3 にレスポンスを返す（このレスポンスには時刻 t_2 と t_3 が書き込まれる）．クライアントはレスポンスを内蔵時計の時刻 T_4 に受け取る．クライアントはサーバよりも τ だけ遅れているとする（この τ は直接計測できない）．すると，クライアントはサーバの時計で時刻 $T_1+\tau$ にリクエストを発信し，$T_4+\tau$ にレスポンスを受け取る．クライアントがリクエストを発信してからサーバが受信するまでの時間と，サーバがレスポンスを発信してからクライアントが受信するまでの時間は同じなので，$t_2-(T_1+\tau)=T_4+\tau-t_3$ だ．往復にかかった時間は両辺の和で，$\Delta=t_2-(T_1+\tau)+T_4+\tau-t_3=T_4-T_1-(t_3-t_2)$ だ（直接計測できない τ が消えることに注意する）．すると，クライアントの時刻 T_4 にはサーバの時刻は $t_3+\Delta/2$ なので，クライアントはこの時刻に時計を合わせればよい．NTP はこの方式で時刻を合わせる．

6-4 IP アドレスが表示される．

6-5 `telnet www.ds.shiga-u.ac.jp 80` の `www.ds.shiga-u.ac.jp` の代わりに ***6-4*** で表示された IP アドレスを使ってもよい（`Host: www.ds.shiga-u.ac.jp` は置き換えてはならない）．この命令は，`ttakuma` ディレクトリの `index.html` ファイルを取得するリクエストを送る，プロトコルは HTTP のバージョン 1.1，Telnet 経由での接続，サーバは `www.ds.shiga-u.ac.jp` と指示している．3 行目が必要なのは，このサーバが複数のサイトのウェブサーバとして使われているため，IP だけではどのサイトかがわからないからだ．存在するファイルについては HTML ファイルの本体の前に `HTTP/1.1 200 OK` などのメッセージが出て，正しくファイルを取得できたことを示す．存在しないファイルでは `HTTP/1.1 404 Not Found` などのメッセージが出てファイルが存在しないことを示す．

6-6 32 bit の IP アドレスのうち 28 bit がネットワークを表すので，残りは 4 bit．IP アドレスの範囲は 133.102.70.0 から 133.102.70.15 だが，ネットワーク全体を表す 133.102.70.0 とブロードキャストアドレスの

133.102.70.15 を除くと 14 台となる．ただし，ルータが 133.102.70.1 だとすると残りは 13 台だ．

6-7 略．

6-8 暗号を鍵で暗号化すると復号される．$(a \text{ XOR } b) \text{ XOR } b = a$ だからだ．

第 7 章

7-1 $S_k = \sum_{i=1}^{k} 4^i$ がおよそ 10×10^6 と 100×10^6 になる k を求めればよい．S_{11} が 500 万あまり，S_{13} が 9000 万弱なので，11 手と 13 手だ．計算時間を 10 倍にしても読める手数は 2 手しか増えない．$S'_k = \sum_{i=1}^{k} 2^i$ なら S'_{22} が 800 万あまり，S'_{25} が 6000 万あまりなので，22 手と 25 手になる．深く読むためには，計算時間を長くするより，読むべき手を減らすヒューリスティック（これを枝刈りという）が有効なのがわかる．

7-2 $\hat{\boldsymbol{x}}_i = [1,\ x_{i,1},\ x_{i,2},\ \ldots,\ x_{i,n}], \hat{\boldsymbol{w}} = [-h,\ w_1,\ w_2,\ \ldots,\ w_n], \hat{y}_i = 2y_i - 1$ と定義する．このとき，$u_i = \hat{\boldsymbol{w}} \cdot \hat{\boldsymbol{x}}_i$ なので，すべての i で $\hat{y}_i u_i > 0$ を満たす $\hat{\boldsymbol{w}}$ が存在することを示せばよい．$n+1$ 個のベクトルが一般の位置にあるから，すべての i について $\boldsymbol{p} \cdot (\boldsymbol{x}_i - \boldsymbol{q}) = 0$ を満たすベクトル $\boldsymbol{p} \neq \boldsymbol{0}$ と $\boldsymbol{q}$ は存在しない．これは，すべての i について $\hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{x}}_i = 0$ を満たすベクトル $\hat{\boldsymbol{p}} = [-\boldsymbol{p} \cdot \boldsymbol{q},\ p_1\ , p_2,\ \ldots,\ p_n] \neq \boldsymbol{0}$ が存在しないことを意味する．つまり，$\boldsymbol{X} = (\hat{\boldsymbol{x}}_1\ \hat{\boldsymbol{x}}_2\ \cdots\ \hat{\boldsymbol{x}}_{n+1})^T$ と定義すると，$\boldsymbol{X}\hat{\boldsymbol{p}} = \boldsymbol{0}$ を満たす $\hat{\boldsymbol{p}} \neq \boldsymbol{0}$ は存在しない．すなわち，行列 $\boldsymbol{X}$ は正則行列なので，$\boldsymbol{X}\hat{\boldsymbol{w}} = \hat{\boldsymbol{y}}$ には解がある．これが求める $\hat{\boldsymbol{w}}$ だ．

7-3 記憶容量の削減になる．計算の速度が向上する．（ドロップアウトや疎性と類似の仕組みで）精度が向上することがある．

7-4 略．

付録

A-1 1 字あたり $-0.5 \log_2 0.5 - 0.25 \log_2 0.25 - 2 \times 0.125 \log_2 0.125 = 1.75$ bit．A を 0，B を 10，C を 110，D を 111 に対応させると，1 字あたり平均

7/4 bit になる．

A-2 エントロピー $H = -\sum_{i=1}^{n} p_i \log p_i$ が最も高くなる場合を求める．$p_1, \ldots, p_n$ は確率を表すので，制約条件 $\sum_{i=1}^{n} p_i = 1$ のもとでの最大化を行う．λ をラグランジュ乗数として，$F = -\sum_{i=1}^{n} p_i \log p_i + \lambda \left(\sum_{i=1}^{n} p_i - 1 \right)$ とおく．すべての i について $\dfrac{\partial F}{\partial p_i} = -\log p_i - 1 + \lambda = 0$ つまり $p_i = \exp(\lambda - 1)$ なので，p_i はすべて同一の値をとる．すなわち，$p_i = 1/n$ となり，圧縮しても一番小さくならないのはすべての文字が等確率で出現する場合だ．

A-3 略．

A-4 $m = 10$，$\phi(m) = 4$，$d = 3$，$b \equiv 8 \pmod{10}$．

A-5 $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{U}\boldsymbol{y}$ から $\boldsymbol{U}(\boldsymbol{x} - \boldsymbol{y}) = \boldsymbol{0}$ がわかる．ここから $|\boldsymbol{x} - \boldsymbol{y}| = 0$ つまり $\boldsymbol{x} = \boldsymbol{y}$ が出る．