# 「アルゴリズム的思考法」はそんなに難しいか

## — 8年間その教育にあたって —

# Is it so hard to study "algorithmic thinking" ?

## — Looking back my education in these 8 years —

ネットワーク情報学部　佐藤　創
School of Network and Information　Hajime SATO

**Keywords** : Algorithms, Programming, Way of Thinking, Abstraction

### まえがき

2001年の本学部開設と同時に始まった「アルゴリズム的思考法」という科目の授業を担当して8年が過ぎた．科目の名称も，プログラミング教育と分離したところもユニークで，本学部の独創性を誇ることができる．

この科目は2009年度から選択科目に移行する．必修から選択になった理由の一つは，新入生の多くがこの科目の履修に違和感を覚え，そこでつまずいてしまうことが学部の教育上好ましくないと判断されたからであろう．

ここでは，なぜ学生諸君がこの科目の履修に困難をきたすのかという問を立て，この方針変更の妥当性について少し論じることにしたい．

## 1　教える側と学ぶ側

この科目が学生諸君と諸先生方から「注目」されたのは，不幸にして単位修得率の低さで悪名を馳せたからであった．ほぼ全員に無難に合格点を与えていたら，とくに注目もされなかったであろう．

不幸の原因は，教える側と学ぶ側の思い描くものがあまりにも隔たっていたことではなかろうか，と考えている．具体的な例に沿って，説明してみよう．

**例題1**　$1+2+3+4+5+6+7+8+9+10$ を計算するアルゴリズムを記せ．

どこにもある平凡な例題だから，学生はいつか向かい合うことになる（高校の教科書にも載っていた）．

この問題に対して，「先生，答えは55でしょう」と指摘する学生がいた．「いや，答えが知りたいのではなく，やり方に注目するのです」と言うと一瞬，キョトンとする．全然「空気」を読んでないが，反応には感謝したい．

昔は小学校でソロバン（珠算）を習い，指ならしに1から100までの和を計算した．答えが5050になることは百も承知で，何度もやって速さを競っていた．

この例題の答えを「カッコ表現」で示せば，

$$(\cdots((1+2)+3)+4)+5)+6)+7)+8)+9)+10$$

である．足し算は2つの数が対象であり，一度に10個の数を足すことはできないので，その順序が問題となる．だから，$(\cdots((10+9)+8)+\cdots)+1$ でもよい．しかし，決して $1+(2+(\cdots+(9+10))\cdots)$ ではない．

$(1+10)+(2+9)+\cdots+(5+6)$ とすればすぐ $11\times5$ となり，$1+2+\cdots+n=(n+1)n/2$ という周知の公式にたどり着くが，アルゴリズムとは別の話になる．

ソロバン塾に通った経験があるか，電卓を使う学生なら，「ご破算（ゼロ・クリア）」の後，次々に加算を繰り返すやり方は直ちに理解できる．上の式はこれを意味している．この「やり方」を順序立てて記述すると，

$$(A)\quad \begin{cases} S \leftarrow 0 & (S\text{を}0\text{とする}) \\ S \leftarrow S+1 & (S\text{に}1\text{を加える}) \\ S \leftarrow S+2 & (S\text{に}2\text{を加える}) \\ \quad\vdots & \quad\vdots \\ S \leftarrow S+10 & (S\text{に}10\text{を加える}) \\ S\text{に答えがある} & \end{cases}$$

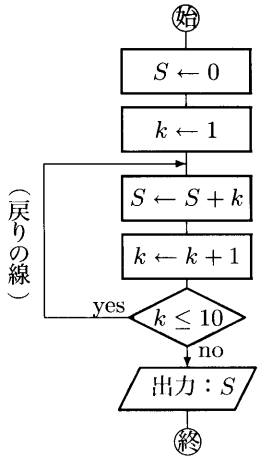
となる．ただし，上の点々のところをきちんと書くと，12行にもなっていかにもアホらしい．そこで，

$$(B)\quad \begin{cases} (1)\quad S \leftarrow 0 & (S\text{を}0\text{にする}) \\ (2)\quad k \leftarrow 1 & (k\text{を}1\text{にする}) \\ (3)\quad S \leftarrow S+k & (S\text{に}k\text{を加える}) \\ (4)\quad k \leftarrow k+1 & (k\text{に}1\text{を加える}) \\ (5)\quad \text{もし } k \le 10 \text{ ならば, (3) にもどる} & \\ (6)\quad S\text{に答えがある} & \end{cases}$$

と工夫をすると，実にうまい！　これなら，1から1000までの和を求めるアルゴリズムも同じ6行で書ける．

授業では，箇条書き (B) を次のような図（流れ図，フローチャート）で表現することを教えている．表現が「読

む」から「見る」になって直観に訴える.




ここまでに多くの「障害物」が横たわっている.

まず, 箇条書き (A) で,「$S$ とは何だ」「なぜ $S \leftarrow 0$ か」「$S \leftarrow S+1$ とは何だ」という疑問が発生する（このような疑問に分解できればまだよい）. 最初に一通り説明をするが, よく聞いてないか, ノートを取らないことが多い.「直観で分かる」ことを期待しているから, それでもよしとしよう. 実は,「変数だ」「初期化だ」「$S = S+1$ とは違う」などと説明すればするほど, ややこしくなる. 繰り返し真似をして馴れ, 自分なりに抽象化して「分かった」と感じる方が早く身に着くと思う.

箇条書き (B) を「うまい！」とは実感できずに, 12 行連ねる「一本道方式」からなかなか離陸できない. 昔ながらの「双六遊び」で「3 へ戻る」と同じことなのだが, 戻ることに不安がともなう. 実際, 戻れない答を書く人が多いから, この不安感は彼らなりの防衛本能とも言える. しかし, 不安を乗り切り, 自分の思考の枠を拡げなければ進歩がない. (B) の核心は「$S \leftarrow S+k$」において $k$ の値が毎回変化するところである. これを「御利益」とみるか「災難」と思うかが, 分かれ道である.

フローチャートはまさに両刃の剣である. 言語から解放され, 曖昧さが減り, 分岐先がよく分かる, と喜ぶべきところだが,「正式なものは図より文」という文化が根付いているためか, 図の簡潔さを喜べない人が予想外に多い. 分からないうちに丸暗記しようとする. こんなものを暗記するんじゃない, と言われても, どうすればよいかわからない.「戻りの線」の先が $k \leftarrow 1$ や $S \leftarrow 0$ の前になる人がいる. 教える側は「信じ難い！」と嘆くが, 学ぶ側は「ちょっと違っただけじゃないか. 大して重要じゃないよ」と気にしない様子だ. 誤りの理由を分かろうとしないから, 繰り返し誤る. テスト本番でも堂々と誤る. 最近は「自分流で何とかなる」と考える人が増えてきた. 単純なアルゴリズムを簡単なルールに従って記す練習をしているわけで, ここに個性尊重を主張する余地はないのだが, 教える側は褒め褒め作戦で対抗する.

「フローチャート」がつまずきの石だとの意見を受けて, 試みに今年度の授業で紹介した「フローチャートのないアルゴリズム」の資料を付録に掲載する.

## 2 面白いことを教えたい

学生が学ぶ気を削がれるのは無理もない面がある. 数の合計なんてそもそも面白くない. 誰しも興味のないものを進んで学ぶわけがない. もっと意外性のあることを話して, 知って得をした気分にさせる必要がある.

卒業生の中に,「先生のあの授業は面白かった」と言ってくれる人がいないわけではない. 実際, 教える側は, 当初から楽しい授業を心がけた. 学生はプログラミング言語の「文法」から解放されていて, 時間的・精神的に余裕がある. 4 月の最初の授業で「イントロ」と称して次の例題を解いてみせた.

**例題 2** 裏に数字の書かれた 20 枚のカードが一列に並んでいる. どこかに最大値（ピーク）があり, その両側は端に向かって数が減る. 1 枚ずつ裏返してピークを早く見つけたい. どのような順で裏返せば効率的か？

授業では A4 版の紙に数字を書き, 裏にして 1 枚ずつ磁石で黒板にとめる. 例えば裏の数字はこうだとしよう.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 8 | 17 | 19 | 26 | 43 | 44 | 51 | 58 | 60 | 66 | 67 | 80 | 82 | 75 | 61 | 34 | 33 | 11 | 9 |

左から通し番号を付け,「何番を見るか」と聞いてみる.「9 番」と声が掛かれば, 9 番を裏返して「58」が現れる. 次が「17 番」ならば「34」が見えて, その結果 17〜20 番はピークでないと分かる. 最後は 13, 14, 15 番が表にされて, 14 番「82」がピークであることが確定する.

裏返したカードの枚数を数え,「さて裏返す枚数を最少にするには最初はどこを見るべきだったか」と問題提起をする. そして, 学生たちに数列を作ってもらい, どんなときも 6 枚以下でピークを探してみせる.

「カードが 1 万枚でも 20 枚見ればピークを探せる. 1 億枚のときは何枚だと思う？ 39 枚で OK. 理由は最後の授業で」と言ってイントロを締めくくる.「興味のある人は, 30 枚で考えてみるといい」と付け加えると, 意欲を燃やす学生が数名出てくる.

これは「フィボナッチ探索」と呼ばれるアルゴリズムとして知られていて, 上記の条件を「単峰性」という.

この授業の顛末はどうであったか？ 緒戦は大勝利, その後は無残な敗北だった. 一般に「面白い」と思うには, それなりの「素養」が必要である. 落語の「落ち」が分からず笑えなかった人に, いくら解説してもちっとも面白がらない. 解説した方も面目ない.

トントントーンと弾みを付けて, 調子の上がったところで「なるほど！」と落としたいのだが, まず配列 (array) で大きくつまずく. 高校までに数列やベクトルを多少かじっていれば何でもないのだが, 受験科目になっていなければ期待できない. とにかく, $a_1$, $a_2$, $P_1$, $P_2$ など添え字 (subscript, index) の付く表記法を知らない学生が圧倒的に多い. $a1$ と $a_1$ の違いを気にするデリカシーはなく, $a_k$ のように添え字が変数になったり, $a_{k_1}$ などと

添え字が2重にでもなると，ブーイングの嵐だろう．

筆者は学生時代，社会人対象のプログラム講習会を手伝ったことがあり，配列がネックになることを身に染みて体験した．サッカーの勝敗表などを例題として，全くの手探り状態から始まり，ようやくミスを指摘すると笑みが浮かぶようになり，最後は喜びの握手で終わった．

その体験と比較すると，目の前の若者たちは案外保守的だと思う．失敗を恐れるのか，新しい「文化」を拒否しているように見える．それも多数をたのんで団結し，「分かりませーん」の大合唱をしている．

さて講義の終盤では約束通り，フィボナッチ数列（$f_1 = 1, f_2 = 2, \cdots$）を使って，探索アルゴリズムを記述する段になる．数列を $x_k\,(k = 1, 2, \cdots, 20)$ とすれば，$20 = f_7 - 1$ であり，最初は $i = 6$ として $f_{i-1}$ 番と $f_i$ 番のカードを表にする．このとき，$x_{f_{i-1}}$ と $x_{f_i}$ の大小関係が必要で，ここに2重の添え字が現れる．ピークの存在範囲は次第に狭まり，添え字の始点（オフセット）を移して考えると，このアルゴリズムの仕組みがパッとひらめく（この仕組みを埋め込み構造とよぶ）．$i$ が $5, 4, \cdots, 1$ と減少して，チェックメイト！

この「ひらめき」がきたら，帰りに一杯やりたいほど嬉しくなる（未成年は上等のケーキとお茶で）．このような知的喜びが学習と教育の原点であってほしい．

この例題の必然の流れは，カードの枚数を一般化し，アルゴリズムの最適性や裏返すカードの枚数を数学的帰納法によって論ずることだが，深入りしないのがよい．

例題2を，底の見えない川の最も深い箇所を探すイメージで紹介したが，「現実味がない」と妙にこだわる学生がいた．微分できない関数の最大値の探索に使えるよ，と答えるわけにはいかない．複雑すぎて分かりにくいという感想も多かった．そこで3年目からイントロを「2分探索法」に変えた（それはそれで受けが悪い）．

このほかに面白そうな題材はいろいろ準備できる．

「迷路脱出」，「魔方陣」，「虫食い算・覆面算」，
「三山くずし」，「ハノイの塔」，「最短経路」，
「継子立て」，「数独」，「詰め将棋・詰め碁」，$\cdots$

はたして面白がってもらえるだろうか，演出方法をあれこれ考えるのは楽しい．

## 3 やはり難しい！

**3.1** すでに述べたように，教える側では，「アルゴリズム的思考法」の内容はとても自然なことだから，先入観を捨てて冷静に受けとれば，大きな苦労をすることなく，むしろ楽しみながら理解を進めることができる．学生に努力はさせるが，過度な負担はかけない．脅かして学ばせる教え方は下の下である，と考えている．実際に，毎年多くの人が優秀な成績を残して，難なく履修を終えている．

一方，この科目のねらいが何なのかさっぱり分からない．何にどのように努力をすれば自分が理解できるようになるのか手がかりが得られない．とにかく自分なりに努力をしたが前期で合格点を取れなかった．そのような悩みを抱える人が少なくない．

成績に大きな差が生ずることはどの科目にもあることで，教える側に立てばそのこと自体に特別な意味はない．問題は，分からない状態にある人が自分は決して分かる状態にはならない，と絶望するような性質をこの科目がもっているかどうか，である．

分からない人が分かるようになるのか？　そのための方策は？　それは授業で実践できるか？

この8年間，これらの問を反芻してきた．

**3.2** 「さっぱり分からない」となる理由の一つに，入学までの準備不足がある．入試科目だけ勉強し，そうでない科目は力を抜く．入試合格だけが目標ならそれは「合理的な作戦」であるが，入学後に「失速」するなら本末転倒も甚だしい．定員までは学生を受け入れる義務が大学側にあり，教育する責任を負う．その責任範囲は一体どこまでか？

分からない学生を絶望させる科目の典型が，「数学」である．「アルゴリズム的思考法」はどうか．

「入力される数の絶対値を出力せよ」「ある年がうるう年かどうか判定するアルゴリズムを記せ」などの問題は，判断・分岐（場合分け）を教えるときに例題となる．一応説明するが，絶対値や割り算の余りは小・中学校の知識である．たとえ知らなくても構わないが，問題の主旨を考えて自ら対策を立てることを要求している．しかし，「これは数学だ！　もうだめだ」となって，思考停止に陥る人が出る．確かに例題には，大小関係，最大値，数列，順序など，数学用語を用いたものが多い．それは問題を正確に，しかも簡潔に表現するのに便利だからであって，数学の問題を解かせてはいない．数学用語といっても日常語に近いが，それでも不安を感じさせてしまう．

**3.3** この科目の難しさは，記憶だけでは対処できないところにある．高校まで覚えることが主流の勉学であったから，半年や1年で方向転換はできない．数学でも公式を暗記して何とか凌いできた．「なぜだろう？」と考える間もなく，選択肢の中から「当たり」を選ぶ訓練をしてきた．「自分でよく考えなさい」と言われても，戸惑うばかりかもしれない．

（変数は）$i$ ですか，$j$ ですか／初期化は0でなくていいのですか／入力が最初にあってもいいのですか／出力をループの中に入れてもいいのですか／$\cdots$

これらの素朴な質問は，学生たちがとにかく覚えて対処しようとする「健気さ」と，指示を待つ自信のなさと，「目の付けどころの悪さ」をよく表している．

問題を出すと，下書きなしでいきなりフローチャートを書き始める人が実に多い．方針が決まらないのになぜ？と不思議に思うが，覚えていた部分を配置し，それらをつなげようとしている．その部分が必要かどうか，変更

しないでよいかどうかを検討しない．目的に適っているかどうかと全体を見直すこともしない．これではこの科目のねらいから外れるが，注意に耳を傾けない．

**3.4** 学部の特徴を十分認識せずに入学すると，自分のイメージとのギャップが「やる気」を失う原因になる．学部紹介のパンフやホームページには，「プログラミング必修」は明記されている．覚悟していれば，その準備にあたるこの科目にもっと力を注いでもよいが，人はそれほど論理的ではない．学ぶ側の「入学すれば何とかなる」の楽観に，教える側は歩調を合わさせられている．

難しいのはこの科目だけではなく，「数的推理」「数的推理演習」「情報処理概論」「プログラミング入門」「プログラミング演習」…と目白押しだ．これらの科目を担当する先生方と頻繁に話し合ったが，中心は「歩みの遅い人をどうするか」などの苦労話であった．

根本は，それぞれの教育内容を「必修」とする必然性があるかどうかの見極めであり，学部全体の方針にかかわる．当時，カリキュラム委員長であった筆者は，「教育」に関する懇談会を何度か試みたが，ついに実現しなかった（成績評価を聖域視したか，単に忙しかっただけか）．

**3.5** 分からない人を減らす「形式的」対策は，教科内容を減らすことと，教える人数を減らすことである．

シラバスでは，2重ループと2次元配列まで学ぶとしているが，1重ループと1次元配列まで分かればよいことにした．少なくともこの段階をクリアしないと，プログラミングで困難をきたす．単に講義を聞くだけでなく，実力をつけることを要求した．そのために，話は最小限とし，小問題を解決する演習に重点を移し，教壇から降りて個々の質問に応じた（質問の少ないのが問題だ）．

2年目にFD事務課の授業補助員制度を活用して，情報管理学科の上級生に演習指導に来てもらった．手を挙げて呼ぶと自分の疑問を解決してくれる．「上級生はちゃんと分かっている！」この事実が衝撃となり，演習に真剣に取り組む人が増えた．この変化は実に印象的であったが，マンネリ化すると効果を失っていった．

大学院生のTAに小問題の採点を頼んで，結果を学生に返却したが大きな効果がない．学生は手を挙げて質問せずに，「ねえ先生，教えて」と個人的に質問にくる傾向があり，1クラス150人は多すぎる．

非常勤の先生を頼んで4展開，クラス学生数を半分にする案が，2年前からようやく実現した．内容を減らし要求水準を下げると「上級者」は退屈するので，A（advance）とB（basic）に分けた．筆者の事情で，少人数クラスを自ら担当できなかったことは非常に残念だったが，担当者の若返りと相乗してその効果は大きかったと思う．

**3.6** 本学部の特長の1つに，1年次の再履修制がある．前期に単位を落とした人は後期に取り戻すことができる．それだけ1年次の基礎教育を重視している．人により学習ペースが違うのが普通で，半期で登れない坂は1年かけて登ればよい．大部分はとにかく単位を取り戻して無事進級できた．前期50点そこそこの人より，ずっとよい成績の人も多かった．4年生にもなると「なーんだ，こんな簡単なことだったのか！」と開眼する人もいた．

制度上の問題点は，再履修クラスが6時限に置かれていたことと，後期にしか置かれてないことである．再履修で落とすと次年度前期には履修できずに1年待たされる．これではセメスター制ではない．一方の学生の便宜が他方の学生に不便を生むことが多い．キャップ制も自習すればよい程度の再履修の場合，不合理にはたらく．だから，試験だけの履修を認める大学もある．

**3.7** 本学部には1年次に進級条件があり，専門必修を4科目以上落とすと2年生に進めない．学部に適合しない人が早く進路変更できるように，との配慮である．

成績評価の方法については明確な共通ルールはないが，出席点という名目では加点しないことになっている．再履修の成績を甘くすると不公平になるので，常に一定の基準で評価するのが公明正大でよいのだが，基準を厳格に適用すると，どうしても卒業できない人が出てくる．

ある先生から「合格率」で評価する方法を知らされた．各年度の出来不出来に関係なく，例えば下位25％を不合格にする．300人のクラスなら最初の試験で75人が落ちるが，以後19人，5人，1人と減少して必ず全員が単位を取れる（欠席者を除く）．再履修クラスは常に上限100名になる．うまい方法だと思ったが，抵抗感が残り，「脅しの教育」にもなるので，採用しなかった．

**3.8** 世の趨勢からか文系色が一層濃くなった本学部の1年生全員が，この科目の内容を「マスターする」ことは難しい．「マスターさせる」ことも難しい．

総括すれば，学生の志望動機の多様化に呼応して，「プログラミング」も含めてこの科目を選択科目にするのが妥当であると考える．必修を続けるのならば，学科制をとって入学時に選択させるのよいが，人気の偏りに翻弄される恐れや，転科の希望に応ずる煩わしさがある．

結果として，新カリキュラムでは「アルゴリズム的思考法」は選択科目となり，「プログラミング教育」は必修を継続し，充実させるという方針が決定された．

前者は筆者の主張のままだが，実は「廃止」される案になっていた．青天の霹靂で，これまでの苦労が全く評価されなかったのかと訝り，今でも肩を落としている．

後者については，「覚悟の苦労」と了解した．

## 参考文献

[1] 佐藤 創，「学生を叱咤激励す」，専修大学全学FD委員会広報誌 6, 2003.

[2] カリキュラム委員会（編集），「学部の必修専門科目について（上）」，ネットワーク&インフォメーション 5, pp.59–64, 2004.

[3] 飯田周作, 飯田千代, 清藤武暢, 佐藤 創，「アルゴリズム的思考法の教育」コンピュータと教育研究集会, 情報処理学会, 2008.

付録　講義資料

# アルゴリズム的思考法への接近

ネットワーク情報学部　　佐藤　創　（2008年4月）

## まえがき

皆さんにアルゴリズムに親しんでもらうための読み物を書いてみました．例題を解きながら，アルゴリズムの世界に近づいてください．

## 1　アルゴリズムとは

アル・フワリズミ
（780頃－850頃）

「アルゴリズム」（algorithm）は，もともとは中世アラビアの数学者・天文学者の名前でしたが，レオナルド・ダ・ビンチのダ・ビンチと同様に，名前の一部アル・フワリズミという地名（現在，ウズベキスタン共和国のヒヴァ）の部分だけが残り，それが変化して現在にいたったということです．

さて，「アルゴリズム」とは何でしょうか．「算法」という意訳がありますが，これでは意味がよく分りません．一言で表現すれば，「問題解決の手順」ですが，これでも漠然としていて誤解を生むもとになります．「問題解決」というと，例えば「地球温暖化問題」「パレスチナ問題」などの「困った問題，取り組むべき課題」が連想されます．しかし，このような問題は一般的すぎてアルゴリズム論の対象にはなりません．どうなれば解決か，何が実行可能な手段か，などを明確に規定できないからです．

一方，アルゴリズムを厳密に定義しようとすると，それを理解するために深い知識が必要となり，初心者にはかえって分りにくいことになります．

ものごとを説明する方法として，例示があります．いくつかアルゴリズムの例を提示すると，あるイメージが伝わります．それに修正を加えて，抽象化していくと，次第に正しいものに近づいていくことが期待されます．

例えば，「おいしい中華料理，麻婆豆腐の作り方」はどうでしょう．食材，調味料，道具，設備をすべてそろえた上で，コックさんがある手順（レシピ）に従って材料に手を加えていくと，やがておいしい麻婆豆腐が出来上がります．いつ注文しても同じものが出来るので，「手順」が決っていると考えられます．再現可能性はアルゴリズムの重要な要件です．では，コックさんがすっかり身に付けた手順を，客観的に表現できるでしょうか．その手順を追えばだれでもこの麻婆豆腐を作れるでしょうか．この点で疑問が残ります

では，「毛糸の手袋の編み方」はどうでしょう．編み方を知らない人には，その方法を容易に編み出すことはできませんが，編み方を知っている人は身の回りにたくさんいます．編み方を

教えてもらえば，だれでも手袋を編めるようになります．手の大きさ，指の形，毛糸の太さなど様々な変化にも対応できます．編み方の手順をすべて言葉で表すのは難しいですが，見れば真似ができるので，「毛糸の手袋の編み方」はアルゴリズムの要件を備えています．

「アルゴリズム」の語源に戻るともっと明解になります．アル・フワリズミの書いた本は，勘定を素早く正しく行う必要のあったアラビア商人たちのための，インド式記数法と加減乗除の計算法に関する教則本でした．皆さんはすでにマスター済みですから，次の計算は簡単です．

**例題1**　3600（=MMMDC）に 24（=XXIV）を掛けるといくつか（1日の秒数）．

昔の人々にこの問題がどんなに難しかったかは，九々を憶えたての小学生にこの掛算を教えることを想像すると，よく理解できます．「アルゴリズム」とはまさに「算術」のことでした．四則演算はどのような基本操作に分解できるか，それをどのように組み合せると $3600 \times 24$ の計算ができるか．徹底的に分析すると，アルゴリズムの本質に迫ることになります．しかし，このアプローチは皆さんを退屈させる恐れがあるので，別の例を求めることにしましょう．

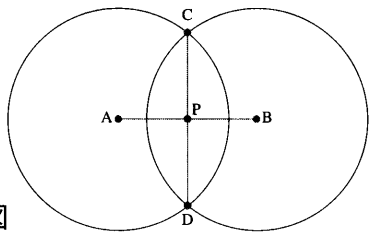
## 2　コンパスによる作図

計算法は典型的なアルゴリズムですが，ここではもう一つの典型を紹介します．

**例題2**　平面上の2点 A, B の中点 P の位置を，コンパスだけを用いて決定せよ．

これは「作図」と呼ばれる古代ギリシャ以来の問題形式ですが，「コンパスだけ」というところがミソです．通常，作図には定規の使用も許されていて，次の作図はご存じの通りです．

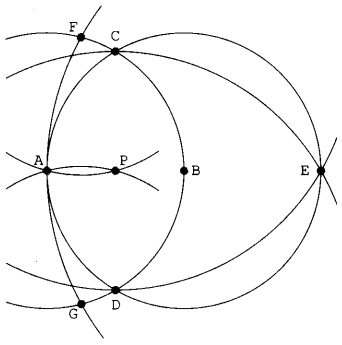
(1) A と B を直線で結ぶ．
(2) A と B をそれぞれ中心とする同じ半径の円を描き，2つの円の交点を C, D とする．
　（円の半径は AB の半分より大とする）
(3) C と D を直線で結ぶ．
(4) 直線 AB と CD の交点が求める点 P である．



参考図

定規の使用を禁止するとちょっとしたパズルになります．例題2の解答を一つ示しますから，実際に試してみることを勧めます（なぜこれでよいか，納得できたら喜ばしい）．

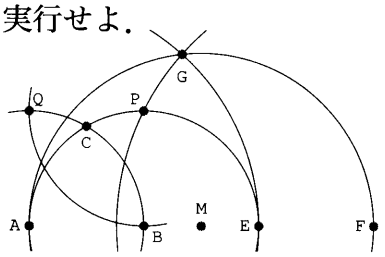
(1) A と B をそれぞれ中心として，半径 AB の円を描き，その交点を C, D とする．（参考図の C, D とは異なる）
(2) C と D をそれぞれ中心として，半径 CD の円を描き，B に近い方の交点を E とする．
(3) E を中心に半径 AE の円を描き，中心 A 半径 AB の円との交点を F, G とする．
(4) F と G を中心に半径 FA (= GA) の円を描き，A でない方の交点が求める点を P とする．




驚くべきことが知られています．「定規とコンパスでできる作図は，コンパスだけでも作図できる」というのです．ただし，直線は引けませんから，直線上の2点を指定することで直線を表すものとします．その証明は初等的にできますが，残念ながらここでは省略します．

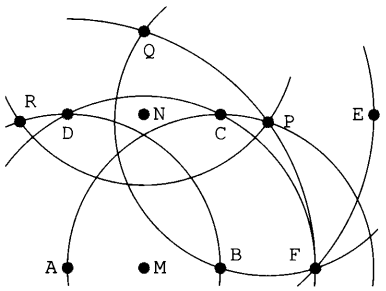
作図問題を続けて，正方形と正五角形をコンパスだけで描くことにチャレンジします．

**例題3**　ABを1辺とする正方形ABPQを描く次の作図法を実行せよ．




(1) 2点A，Bについて，例題2の(1)，(2)によって点Eを定める．
(2) 2点B，Eについて，(1)と同様に点Fを定める．
（E，Fは直線AB上にあり，AB=BE=EFとなる）
(3) 例題2の方法で，BEの中点Mを定める．
(4) Mを中心に半径AM，Aを中心に半径AEの円を描き，その交点の1つをGとする．
(5) Fを中心に半径FGの円を描き，中心B半径ABの円との交点の1つをPとする．
(6) Pを中心に半径PB（=AB）の円を描き，中心A半径ABの円との交点でBでない方をQとする．

**例題4**　ABを1辺とする正5角形ABPQRを描く次の作図法を実行せよ．




(1) 例題3の方法で正方形ABCDを描き，直線DC上にDC=CEとなる点Eを定め，例題2の方法によりABの中点MとCDの中点Nを定める．
(2) Mを中心とする半径MCの円と，Nを中心とする半径NEの円を描き，Bに近い方の交点をFとする．
(3) Aを中心とする半径AFの円を描き，中心B半径ABの円との交点の1つをPとする．
(4) Pを中心とする半径ABの円を描き，中心A半径AFの円との交点のうちEでない方をQとする．
(5) Qを中心とする半径ABの円を描き，中心A半径ABの円との交点のうちBから遠い方をRとする．

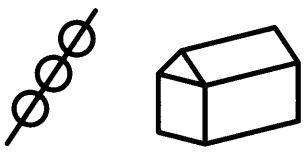
正3角形や正6角形の作図は極めて簡単です．一方，正7角形の作図アルゴリズムは存在しません．「ないこと」を示すのは容易ではありませんが，手段を限定してはじめて，可能・不可能を厳密に議論できるようになることに，注意を向けてください．

作図法のように，実行可能な要素的操作の組み合せによって表現され，それを手順どおりに実行すれば，有限時間内に問題の解が得られるもの，それがアルゴリズムです．ここでは分岐や反復を含まない単純な構造のアルゴリズムを考えました．

## 3　一筆書き

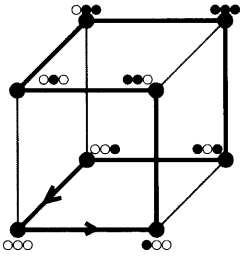
次に，直観的で分りやすいアルゴリズムの例として，「一筆書き」を取り上げます．

**例題5**　右のような串団子の絵（線画）があります．
これを一筆書きで描くことができるでしょうか．
では，家の絵の場合はどうでしょうか．



この例題は楽勝ですが，どんな線画なら一筆書きできるか，どこから書き始めてどう進めるか，一般的にすっかり解決しています．皆さんも少し考えれば，きっと分ります．

次の例を考えましょう（すべての「点」をたどる一筆書きです）.



鉄板の上に3枚のお好み焼きがあります. 1枚ずつ返しながら，表裏のパターンをすべて1度ずつたどってもとに戻すにはどうしますか.

お好み焼きを右から1，2，3と番号でよび，表を○，裏を●で表します. 初期パターン○○○の1番を返すと，●○○になります. 返すお好み焼きの番号を，最初から1，2，1，3，1，2，1，3とすれば，

○○○→●○○→●●○→○●○→○●●→●●●→●○●→○○●→○○○

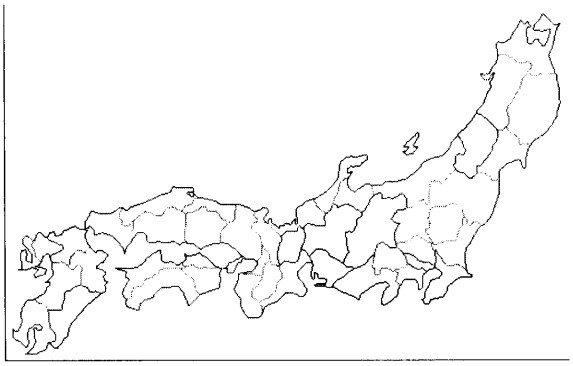
となります. これは立方体の8頂点を1度ずつたどり，戻ってくる一筆書きに対応します.

**例題6**　4枚のお好み焼きを1枚ずつ返しながら，4枚の表裏パターンをすべて1度ずつたどり，もとに戻すにはどうするか.

1つの解答は，番号の列で表すと，1，2，1，3，1，2，1，4，1，2，1，3，1，2，1，4となります. お好み焼きが何枚でも対応できる巧妙なアルゴリズムが知られています.

さて，北海道と沖縄を除く日本の45都府県は，関門大橋と本四架橋を含めると道路で結ばれています. そこで，次の例題ができます.

**例題7**　これらの都府県をすべて一度ずつ通る縦断ルートがあるか. あれば求めよ.



長野県は8県と隣り合っています. 長崎県の隣りは佐賀県だけですから，長崎県が始点（または終点）になります. 例題7の答えは，残念ながら「ノー」です. 福岡県を2度通らないと山口県に行けないからです.

次は一筆書きではないのですが，地図に関連して別の例題を考えましょう.

**例題8**　隣り合う都府県が別の色になるように，なるべく少数の色で塗り分けなさい.

どんな地図でも4色あれば十分，という「4色定理」がコンピュータによって証明された話は有名です. 例題8では，3色では不可能か，どのように塗るか，が問われています.

最後に，これまた有名な「巡回セールスマン問題」という未解決問題を紹介しましょう. 上の地図で路線と距離が与えられているとします. 都府県庁所在地の一つから出発してすべての都府県庁所在地をたどって戻ってくる最短巡回経路はどれでしょう. 経路は一筆書きでなくても構いません. この問題について，基本的に全部調べる以外の良いアルゴリズムが発見されていません. $n$ 個の都市があれば，$n!/2$ 通りの巡回順序があります. $n = 45$ のときこれは

$$5981111043274009728098158074782885753219186688000000000$$

という凄まじい数になり，ギブアップです. 皆さん，何か良い方法はないでしょうか.

## 結び

例をいくつか紹介しながら，アルゴリズム＝「問題解決の手順」にスポットライトを当ててみました. アルゴリズムをあれこれ考えることに，面白さを見出してもらえたでしょうか.