アルゴリズム構造に基づくコア・イメージのアルゴリズム教育への応用

川口 順功
静岡産業大学　情報学部
kawaguti@ssu.ac.jp

概要：アルゴリズムは、使えるだけでなく本質的な理解によって他の問題への応用が可能になり、その価値が高まると考えられる。アルゴリズムの本質的な理解のためには、まずその構造を理解することが求められる。筆者は、アルゴリズム構造を理解するために、その構造を反映させたコア・イメージを作成し、それを利用したアルゴリズム教育を行っている。

本稿では、アルゴリズム構造を意識して作成されるコア・イメージを使ったアルゴリズム教育の試みを紹介しながら、大学でのアルゴリズム教育について考察する。

1　はじめに

大学の情報教育としてのアルゴリズム教育は、プログラミング教育の一部に位置付けられる場合が多い。筆者が担当している「情報構造基礎・同演習」「アルゴリズム応用論・同演習」でも、プログラミングを前提としたアルゴリズムが主なテーマとなっている。したがって、アルゴリズムを使ってプログラムを作成できるようになることが、主な目標となりがちである。しかし、アルゴリズムを学ぶ意義は、問題解決の手順としてのアルゴリズムの理解だけでなく、そのベースとなるものの捉え方や考え方などを含めた、その本質的な理解にもある。本質的な理解のためには、アルゴリズム構造を理解することが必要となる。したがって、アルゴリズム教育では、アルゴリズム構造を理解させることも重要なテーマとしなければならない。

本研究の目的は、大学におけるアルゴリズム教育において、問題を解くとき必要となるアルゴリズム構造を理解させ、その本質的な理解につなげることである。そして、本質的な理解の延長上にある他の問題への応用につなげることである。

本研究では、アルゴリズム構造の理解の根拠をアルゴリズムのコア・イメージ（詳細は後述）の理解と位置づける。コア・イメージは、アルゴリズム構造が反映されたイメージ図で、アルゴリズム名とともに想起され、その図をもとにアルゴリズムの説明ができるものとする。

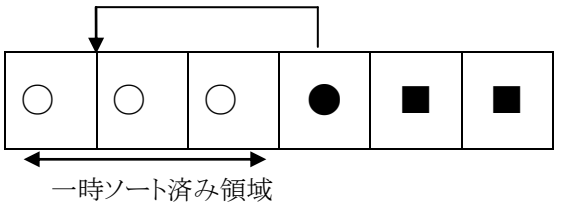
本稿では、これまでアルゴリズム教育で実施しているコア・イメージを使った試み[1][2][3][4]を紹介しながら、大学でのアルゴリズム教育について考察する。

2　アルゴリズム構造に基づくコア･イメージについて

まず、アルゴリズムのコア・イメージについて説明する。「イメージ（image）」は、一般的には､「心の中にうかべる姿･像､感覚像、心象」[5]などの意味で使われる。本稿では、対象とするアルゴリズム名を聞いたとき、想起される図、式、グラフ、文章、または、それらを組み合わせたものを、そのアルゴリズムのイメージとする。ただし、イメージ化されたものには、アルゴリズム構造が反映されていなければならない。したがって、本稿では、アルゴリズムのイメージは、その構造を想起できるイメージ図のことを意味する。特に､対象となるアルゴリズム名を聞いたとき、すぐに想起されるメインとなるイメージ図を「コア・イメージ（core image）」と呼ぶ。

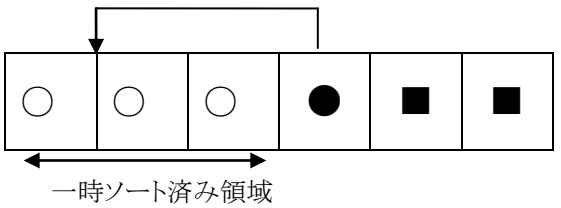
図 2-1 は、授業の中で使っているコア・イメージの例で、単純挿入ソートのアルゴリズム用のものである｡「図の一時ソート済み領域に、ソート対象領域の先頭要素（図の●）を挿入すべき位置に挿入して、一時ソート済み領域を配列の後方に拡げていくことによって全体の配列をソートする」というアルゴリズム構造をイメージ図にしたものである｡「単純挿入ソート」という言葉を聞いた時、想起されるべきイメージ図として作成したものである。テキストボックスの文章での説明は、図 2-2 の二分挿入ソートとの違いを意識した補足的な内容であり、文章の下線部分で 2 つのアルゴリズムの違いを表現している。授業では、文章として覚えるべきアルゴリズムを、このコア・イメージからアルゴリズム構造を理解し、これをベースにアルゴリズムを説明できるように指導している。

本研究では、上述した例のようにアルゴリズム名を聞いたとき想起される言葉としてのイメージを意識してコア･イメージを作成し、これによってアルゴリズム構造を理解させることを試みている。以下では、これらの試みの中からいくつかの例を紹介する。



挿入対象要素(●)から前の領域は一時ソート済領域とみなして、データをずらしながら、挿入対象要素の位置を求めて挿入する。

図 2-1 単純挿入ソートのコア・イメージ



挿入対象要素(●)から前の領域は一時ソート済領域とみなして、挿入対象要素の挿入位置を二分探索で求めた後、データをずらし、挿入する。

図 2-2　二分挿入ソートのコア・イメージ

3　コア･イメージのアルゴリズム教育への応用とその考察

筆者は、アルゴリズム教育として「情報構造基礎・同演習」「アルゴリズム応用論・同演習」の科目を担当している。本学でのアルゴリズム教育は、基本的にはプログラミング教育の中に位置づけられており、プログラミングを前提としたアルゴリズム（以下、プログラミング的アルゴリズム）を対象とする。しかし、アルゴリズム構造の理解という観点から数学(算数)の問題の解き方としてのアルゴリズム（以下、数学的アルゴリズム）も必要と考え、それらも授業の中に取り入れている。

ここでは、プログラミング的アルゴリズムから「再帰的アルゴリズム」と「二分探索」、数学的アルゴリズムから「n 進法」と「濃度算」を取り上げ、それらのコア・イメージの応用例を紹介しながら、大学でのアルゴリズム教育について考察する。

3.1 コア・イメージのプログラミング的アルゴリズムへの応用

再帰的アルゴリズムは、プログラム独特のアルゴリズムのように扱われがちであるが、考え方は文章構造の理解など身近なものにもつながるアルゴリズムである。一般的なアルゴリズム教育では、階乗計算、クイックソート、ハノイの塔など代表的な例を取り上げ、そこでの使い方を解説して、とにかく再帰的な処理を扱う場合に有効であると説明して終わることが多い。したがって、学生にとって、このアルゴリズムはどのような場合に有効に使えるのかなど、本質的な理解が困難なアルゴリズムの一つになっている。

ここでは、まず、再帰的アルゴリズムのコア・イメージについて説明する。次に、二分探索のコア・イメージを示し、この二分探索と再帰的アルゴリズムとの関連性について、コア・イメージの観点から述べる。

3-1-1 再帰的アルゴリズムのコア・イメージ

再帰的アルゴリズムは、「ある問題を部分問題に分割したとき、その部分問題が元の問題と同じ構造であることを利用して、部分問題を元の問題と同じアルゴリズムを使って解くアルゴリズム」である。プログラムでは、自分の関数から自分自身を呼び出す再帰呼び出しの形になる。

図 3-1 は、再帰的アルゴリズムのコア・イメージである。この図は、全体の問題が A、B、C の 3 つの部分問題からなり、問題 C が元の問題と同じ構造を持つ部分問題である条件のもとで、再帰的アルゴリズムのアルゴリズム構造が理解できることを念頭に作成されたものである。

このコア・イメージからアルゴリズム構造として理解してもらいたい主な点は、次のようなことである。

- 部分問題（C）が元の問題と同じ構造であり、この部分の処理が再帰的になる。
- 構造的には同じであるが、問題の内容（処理対象とするデータなど）が小さくなることを、図の大きさで示している。
- 再帰呼び出しには、必ず終了する条件が必要となる。最後の C は、実質的には再帰呼び出しをしない、または、呼び出しても実質的な処理は何もせずに戻ってくる処理となる。
- 矢印で示される再帰呼び出しは、プログラムでは自分の関数から自分自身を呼び

出す形であるが、実体は自分と同じコピー関数を呼び出すことである。つまり、基本的に上から下に逐次実行される処理が、矢印で示された所で自分のコピー関数を呼び出し、その関数の処理が終了した時点で呼び出し元に戻るという、一般的な関数呼び出しの形である。コピー関数を呼び出すとき重要なことは、引数の内容の推移である。

再帰呼び出しのパターンには、2 か所以上の再帰呼び出しがあるなど、いくつかのパターンがある。しかし、基本的にはこの形の構造で理解できれば十分と思われる。授業の中では、「再帰的アルゴリズム」と聞いたら、このコア・イメージを想起し、説明できるように指導している。

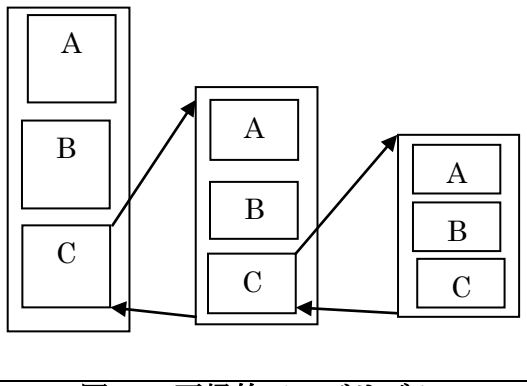


図 3-1 再帰的アルゴリズムのコア・イメージ

3-1-2　二分探索のコア・イメージ

ここでは、二分探索のアルゴリズムとコア・イメージを示し、これと再帰的アルゴリズムとの関連について述べる。

二分探索のアルゴリズムの基本的な考え方は、次のとおりである。

昇順（降順の場合もある）にソートされた探索対象領域の中央の値と探索 key の値を比較し、一致したら探索成功で探索終了、(探索 key < 中央の値）のとき中央より前の部分を次の探索対象領域に、(中央の値 < 探索 key) のとき中央より後の部分が次の探索対象領域になる。探索対象領域がなくなるまで以上の処理を繰り返す。

具体的なアルゴリズムで示すと、次のようになる。

以下の処理を探索成功か、探索対象領域がなくなる（探索失敗）まで繰り返す。

① 探索対象領域の配列の中央の値と探索 key を比較する
②（探索 key = 中央の値）→ 探索終了
③（探索 key < 中央の値）→ 探索対象領域を中央の値より前の部分にして①へ
④（探索 key > 中央の値）→ 探索対象領域を中央の値より後の部分にして①へ
(③④で探索対象領域がない場合は、探索失敗で終了)

図 3-2 は、二分探索のコア・イメージである。上記のアルゴリズムをイメージ化したものであるが、探索を継続するとき探索対象領域（図の■）がなくなるまで繰り返すことまでは表現していない。しかし、アルゴリズム構造の観点から言えば、補足的な説明で十分と思われる。

図 3-3 は、二分探索のプログラム例である。繰り返しの処理を do-while 文で表現している部分を含めて、上記のアルゴリズムを忠実にプログラム化したものと考えてよい。

一方で、二分探索のコア・イメージから不一致時の処理が再帰的に繰り返されると捉えると、再帰的アルゴリズムでのプログラム化も考えられる。図 3-1 の再帰的アルゴリズムのコア・イメージにおける全体の処理を A、B、C、D の 4 つの処理とし、A を探索範囲がない場合の探索失敗で return する処理、B を一致した時の処理、C と D を次の探索範囲を半分にした再帰的な処理とすれば、再帰的アルゴリズムの構造と考えられる。したがって、図 3-4 のような再帰的アルゴリズムを利用したプログラムも考えられる。処理効率を考えると薦められるプログラムではないが、コア・イメージの観点からのアルゴリズム構造を意識すれば、このようなプログラムでも良いように思われる。

この二分探索の例のように、コア・イメージを意識して、改めてアルゴリズム構造を捉えると、どのような場合に再帰的アルゴリズムが有効かの理解も深まると思われる。この件に関する検証については、今後の課題としたい。

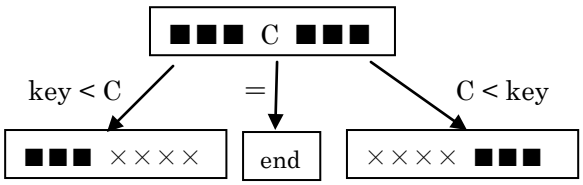


図 3-2 二分探索のコア・イメージ

```
int bin_search(const int a[], int n, int key)
{
      int   pl = 0;
      int   pr = n - 1;
      int   pc;
      do {
            pc = (pl + pr) / 2;
            if (a[pc] == key)
                  return (pc);
            else if (key < a[pc])
                  pr = pc - 1;
            else
                  pl = pc + 1;
      } while (pl <= pr);
      return (-1);
}
```

図 3-3 二分探索のプログラム

```
int bin_search(const int a[], int pl, int pr,
                                          int key)
{
      int    pc;
      if (pl > pr)
            return(-1);
      pc = (pl + pr) / 2;
      if (key == a[pc])
            return (pc);
      else if (key < a[pc])
            return(bin_search(a,pl, pc - 1, key));
      else
            return(bin_search(a,pc+1, pr, key));
}
```

図 3-4 再帰的アルゴリズムを使った
二分探索のプログラム例

3.2 コア・イメージの数学的アルゴリズムへの応用

学生の多くは、「n 進法」や「濃度算」などと聞いても、具体的なアルゴリズム構造としての知識は持っていないように思われる。しかし、問題を解く段階になると、経験的に覚えている個々の問題に対する解き方を思い出しながら問題を解き始める。もちろん、このようなことは、「n 進法」や「濃度算」などに限らず一般的な数学（算数）や理科の問題を解く場合でもあり得ることである。このようにアルゴリズム構造の知識なしで問題を解くということが起こる一つの原因は、それまでにアルゴリズム構造の理解という意識がなかったからだと推測される。

そこで、数学的な問題をアルゴリズム教育の授業の中で取り上げるとき、アルゴリズムを使って問題を解くだけでなく、コア・イメージによってアルゴリズム構造を理解し、それを利用して問題が解けるようになることを目標としている。ここでは、「n 進法」と「濃度算」のコア・イメージを応用したアルゴリズム教育を紹介しながら、これらのアルゴリズム構造の理解について考察する。

3-2-1　n 進法のコア・イメージ

n 進法の問題は、大きく「n 進法→10 進法（パターン n10）」と「10 進法→n 進法（パターン 10n）」の 2 つに分類できる。例えば、パターン n10 は、「[例題 1] 2 進法の 101011 は 10 進法ではいくつになるか？」、パターン 10n は、「[例題 2」10 進法で 43 を 2 進法で表すと？」というような問題である。どちらのパターンの問題も、一般的にはアルゴリズム化された計算式で求めることができる。しかし、アルゴリズムとしての計算式の意味や計算手順を理解するためには、n 進法のコア・イメージを使って、そのアルゴリズム構造を理解することが不可欠と考えられる。

n 進法のアルゴリズム構造を理解するには、桁上がりの仕組みに焦点を当てた n 進法用の入れ物を想定し、その構造を理解することが良いと思われる。図 3-5 は、桁上がりの仕組みの理解に焦点を当てた入れ物の構造としての n 進法のコア・イメージである。

n 進法の問題は、あるもの（例えば、コイン）が n 進法用の入れ物にどのような状態で入っているのか（パターン n10）、あるいは、入るのか（パターン 10n）を具体的に頭に思い浮かべることができれば良いと考える。図に示す入れ物は、各列が n－1 段の棚からなり、右から必要な列の数だけ並んだものである。図の各列は n 進法の各桁に対応し、右端が 1 桁目である。各列の段にはその列用のパックが載り、n 進法の各桁の数字は、対応する列の段に載っているパックの個数を表す。「(1)は 1 個のパック」「(n) は、(1)を n 個まとめてパックにしたもの」、「(n^2) は、(n) を n 個まとめてパックにしたもの」を表す。式で表すと、次のようになる。

$(n^{k+1}) = \{(n^k), (n^k), \cdots (n^k)\}$

$(k = 0,1,2 \cdots)$

各桁の位を各列に対応させて考えると、各

列には n－1 段の棚しかないので、その列に載せられるべきパックが n 個になったら、n 個分をまとめて一つのパックにして、その左の列の棚に移すことになる。いわゆる桁上がりの仕組みである。

アルゴリズム構造は、まず、各桁の個々の状態を表形式で表し、次に、各桁の状態の合計を求めるという、表構造と考えればよい。具体的には、n 進数（$a_k \cdots a_3\ a_2\ a_1\ a_0$）$_n$ に対して、次のように各桁の状態を表形式にまとめ、下位の桁(右の桁)の数から順に、1 ,n, n^2 … を掛けて合計を求める。

n^k	・・・	n^2	n^1	n^0 (=1)
a_k	・・・	a_2	a_1	a_0
$n^k \times a_k$	・・・	$n^2 \times a_2$	$n^1 \times a_1$	$1 \times a_0$

例題 1 の解は、次のようになる。

2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	1
32	0	8	0	2	1

合計を求めて、

101011＝32×1＋8×1＋2×1＋1＝43

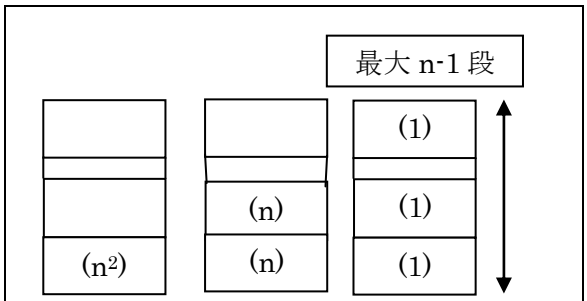


図 3-5 n 進法のコア・イメージ

3-2-2 濃度算のコア・イメージ

濃度算は、「5%の食塩水 40g と 9%の食塩水 60g を混ぜると何%の食塩水になるか？」というような問題である。濃度算は、それぞれ異なる状態にある 2 つ以上のものを一つにまとめたときの状態をテーマとした問題であるが、問題のパターンは大きく 2 つに分類できる。「2 つ以上のものを一つにまとめたときの状態を求める問題（パターン 1)」と「いくつかのものを一つにまとめたときの状態を与えて、初期状態のものを求めさせる問題（パターン 2)」である。上述した問題は、パターン 1 の典型的な問題である。

いずれのパターンでも、前述した n 進法と同様に、個々の状態を表形式で表し、最後に合計で求めるという表構造のアルゴリズムと考えてよい。具体的には、それぞれの状態を「食塩水(g)×濃度（%）＝食塩（g)」の式で表し、その合計を下段に求めて表形式にまとめ、その合計行で表現される一次方程式を解く、というアルゴリズムである。

図 3-6 は、濃度算のコア・イメージである。異なる状態にある 2 つ以上のものを縦に並べ、その横に対応する個々の状態「食塩水×濃度＝食塩」を書き、最後に食塩水と食塩の合計を求めるという表構造としてのアルゴリズム構造である。この図は 2 つの場合であるが、3 つ以上の場合も全く同じ構造となる。

コア・イメージに基づくアルゴリズムを具体的に示すと、次のようになる。

①「食塩水(g) × 濃度（%）＝食塩（g)」に基づき、下記に示すような表を作成する。そのとき、パターン 2 で求めるもの（未知数）があるとき、それを x とおいて式を立てる。

② 食塩水と食塩の合計を求める

③ 最終的な合計行から導かれる一次方程式により、未知数の値を求める。

図 3-7 は、上述した濃度算の例題に対する、表構造としてのアルゴリズム構造から求めた解答例である。実際には、次の表のように図の右にある 3 行からなる表（見出しは除く）を作成し、合計行から導かれる一次方程式を解く。

食塩水	濃度	食塩
40	5 / 100	2
60	9 / 100	5.4
100	x	7.4

最後の合計行より、

100 × x ＝ 7.4

x ＝ 7.4／100 ＝ 0.074 （7.4%）

次に、パターン 2 の問題「2%の食塩水 300g に 9%の食塩水を混ぜて 7%の食塩水を作るには、9%の食塩水を何 g まぜればよいか？」について考える。

前と同じように、右の 3 行からなる次のよ

うな表を作成し、合計行の一次方程式を解く。

食塩水	濃度	食塩
300	2 / 100	6
x	9 / 100	(9 / 100) x
300＋x	7 / 100	6 ＋ (9 / 100) x

最後の合計行より

(300＋x) × 7 / 100 ＝ 6 ＋ (9 / 100) x
x ＝ 750（g）

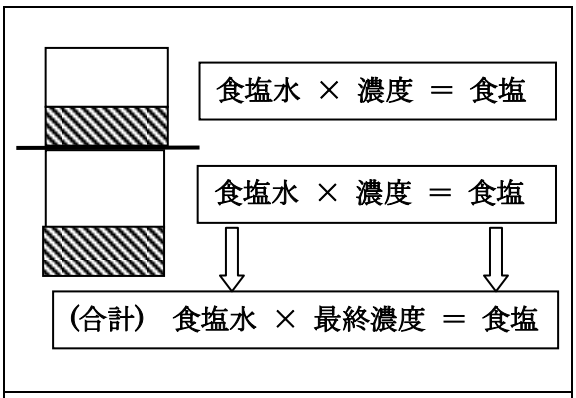


図 3-6 濃度算のコア・イメージ

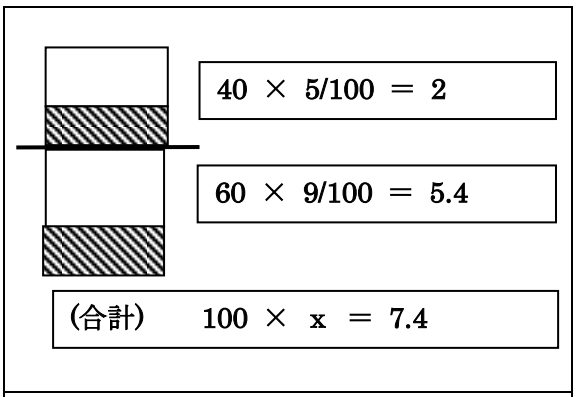


図 3-7 濃度算の解答例

「n 進法」と「濃度算」のアルゴリズムは、異なるものように思われる。しかし、アルゴリズム構造に注目すると、表構造という共通したアルゴリズム構造になる。

4 まとめ

例年、前期開講の「情報構造基礎」という授業の最後に、「コア・イメージを使ったアルゴリズムの学びについて、どのように思うか。」というアンケートを取っている。今年度は 30 名の受講生からの回答が得られた。

基本的には、全員が有効性を認めている。回答の中では、「文章や式だけでは理解が難しいアルゴリズムが、コア・イメージを用いて視覚的に理解できたように思う」という内容が多い。また、数は少ないが、「数学の問題が、コア・イメージを意識することにより解けるようになった」「コア・イメージの考え方は、アルゴリズム以外にも使えそうだ」などの意見もある。

本稿で報告した内容と、授業やアンケートによる学生の反応から言えることをいくつか挙げると、次のようになる。

① コア・イメージの理解をアルゴリズム構造の理解の根拠に置くことは、学生に受け入れられている。
② 再帰的アルゴリズムは、コア・イメージと連動させると、どのような場合に使えるかなど理解が深まる可能性がある。
③ 「n 進数」と「濃度算」のように、コア・イメージを作ることにより、共通するアルゴリズム構造（表構造）が見えてくる場合がある。
④ 数学的アルゴリズムのコア・イメージの応用の方が、学生の印象に残るようである。特に、苦手意識をもった学生には、有効となっているように思われる。

今後の課題の一つは、コア・イメージを利用したアルゴリズム教育の成果を厳密に評価・検証することである。

参考文献

(1) 川口順功,「アルゴリズムのコア・イメージを使ったアルゴリズム教育」, 第 27 回ファジィシステムシンポジウム講演論文集, pp. 1330–1333, 2011
(2) 川口順功, 高橋等, 永田奈央美, 大石義, 「アルゴリズム的思考のビジュアル化によるラーニング構造の抽出」, 大学 ICT 推進協議会 2011 年度年次大会論文集, pp. 282–287, 2011
(3) 川口順功,「イメージ化したアルゴリズムの教育への応用」, 大学 ICT 推進協議会 2012 年度年次大会論文集, 2012
(4) 川口順功,「コア・イメージを利用したアルゴリズム教育の試み」, 静岡産業大学 情報学部 研究紀要, pp. 339–365, 2013
(5) 西尾実, 岩淵悦太郎, 水谷静夫編,「岩波国語辞典第四版」, 岩波書店, 1986