イメージ化したアルゴリズムの教育への応用

川口 順功
静岡産業大学　情報学部
kawaguti@ssu.ac.jp

概要：アルゴリズム教育では、それを使って問題を解く、または、プログラムを作成することが優先されがちであるが、アルゴリズムを学ぶ意義は、その本質的な理解と応用にもあると考えられる。アルゴリズムの本質的な理解と応用の鍵は、その構造の理解にあると考えられる。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が重要なテーマとなる。

本稿では、アルゴリズムをイメージ化したものをコア・イメージとし、アルゴリズム構造の理解をコア・イメージの理解と位置づけ、コア・イメージによるアルゴリズムの本質的な理解と応用について考える。これまでアルゴリズム教育で試みてきたコア・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。

1　はじめに

大学でのアルゴリズム教育の目標は、それほど議論の対象とされない。アルゴリズムを使って問題を解ける(プログラム作成も含む)ようになることだと考えられているからである。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はなく、そのベースとなるものの捉え方や考え方などを含めた、その本質的な理解と他の問題への応用にもあると考えられる。アルゴリズムの本質的な理解と応用の鍵は、その構造の理解にあると考えられる。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が重要なテーマとなる。

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の構造を理解させ、その本質的な理解と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、アルゴリズムをイメージ化したもので、その構造が反映されたものでなければならない。そして、アルゴリズム名とともに想起され、アルゴリズムの応用につながるものとする。

本稿では、これまでアルゴリズム教育で試みてきたコア・イメージの応用例[1][2][3]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　アルゴリズムのイメージ化とコア･イメージについて

まず、アルゴリズムのイメージ化について説明する。「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」[4]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し、そのようにイメージされるものを作成することをイメージ化と呼ぶ。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化は、その構造を想起できるイメージ図を作成することを意味する。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。

筆者が「アルゴリズムのコア・イメージ」に注目し始めたのは、言葉に対する（一般的な意味での）イメージの重要性を強く意識するようになったからである。例えば、「オンライン（online）」と「オンデマンド（on demand）」にある「オン（on）」のイメージである。「オン」の本質的な意味はモノとモノの「接触」という位置関係[5]であり、「オン」のイメージも「接触している状態」でなければならない。このことを理解していると、「オンライン」は、回線との非接触(off)を意味する「オフライン（offline）」との対比で意味が明確になり、「オンデマンド」の「オン」は、「接触 →　モノ(コト)とモノ(コト)の接触でつながる →　～と同時に」とイメージを展開すれば、「要求と同時に」という意味になることがわかる。したがって、「オン」のイメージによる本質的な理解が、「オンライン」と「オンデマンド」という２つの言葉の理解と大きく関わってくる。

本稿でも、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、それをアルゴリズムのコア･イメージと考える。上述した一般的な言葉のイメージとの違いは、コア･イメージがアルゴリズム構造を反映し、その理解の根拠とされることである。

3　アルゴリズム教育へのコア･イメージの応用例とその考察

筆者は、アルゴリズム教育として「情報構造基礎」「アルゴリズム応用論」などの科目を担当している。大学で行うアルゴリズム教育は、基本的にプログラミング教育の中に位置づけられ、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、筆者は、それだけでなく、アルゴリズム構造の理解という観点から数学(算数)などで扱うアルゴリズム（以下、数学(算数)的アルゴリズム）も必要と考え、それらを積極的に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「クイックソート」と「再帰的アルゴリズム」、数学（算数）的アルゴリズムから「鶴亀算」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 プログラミング的アルゴリズムのコア・イメージの応用例

クイックソート（quick sort）は、ソートアルゴリズムの一つで、「ピボット（pivot：基軸となる値）により、データをピボットより小さい値のグループと大きい値のグループの２つに分け、さらに分けた 2 つのグループに同じ操作を再帰的に適用する」という考え方に基づく、再帰的アルゴリズムを利用したアルゴリズムである。ここでは、クイックソートと再帰的アルゴリズムの２つのコア・イメージを応用したアルゴリズム教育を紹介しながら、この２つのアルゴリズム構造の理解について考察する。

(1) クイックソートのアルゴリズムの基本的な考え方と具体的なアルゴリズム

クイックソートのアルゴリズム構造を理解するには、再帰呼び出しのプログラムを使った再帰的アルゴリズムの理解が必要となる。基本的には、ソート対象領域をピボットによりピボット以下とピボット以上の領域に分け、その分けた 2 つの領域に同じ操作を再帰的に適用するという考え方である。ピボットは、領域分割アルゴリズム（この理解も必要）によりどちらかの領域に入る。

再帰呼び出しを使う関数としての具体的なアルゴリズムは、次のようになる。

①配列のデータからピボットを選ぶ。
（配列の中央の値を取る場合が多い）
②領域分割アルゴリズムにより、ピボット以下のデータとピボット以上のデータの２つのグループに分ける。（ピボットのデータは状況によりどちらかのグループになる）
③ピボット以下(前方)のグループのデータ数が２以上なら、①～④を再帰的に繰り返す。
④ピボット以上(後方)のグループのデータ数が２以上なら、①～④を再帰的に繰り返す。
（③④で、再帰的アルゴリズムを使う）

(2) クイックソートのコア・イメージ

図 3-1 が、クイックソートのコア・イメージである。これは、処理される配列のデータの流れに焦点を当てたものである。一方で、プログラム構造に焦点を当てたコア・イメージも考えられる。それは、後述する図 3-4 に示すクイックソート用プログラム展開イメージ図である。このイメージ図は再帰的アルゴリズムのコア・イメージに基づいて作成されるもので、プログラム表現に重点を置けば、これをコア・イメージとすることもできる。しかし、グループ分けに使われる領域分割アルゴリズム、データ展開の順序、高速アルゴリズムである根拠などを意識して、図 3-1 をクイックソートのコア･イメージとしている。

処理されるデータに焦点を当てるので、基本的には、ピボットにより２つに分割されていく配列の状態がわかればよい。つまり、2 分木のイメージである。しかし、このコア・イメージでは、配列の要素が 1 個になったものはそれ以上分割する必要がなく最終状態であることを示すために、2 分木から切り離してある。これは、配列の状態によっては 2 分木でなく、配列の要素が 1 個になったものが中央に残り、3 つ以上の分岐になる場合があるからでもある。これはピボットによる領域分割アルゴリズムに基づくのであるが、ここではこのアルゴリズムの説明は省略する。

2 分木の展開の順序の理解も必要と考えられる。この展開は、再帰的アルゴリズム（後述）により深さ優先順探索（木のルートから

左方向に迷路をたどるように探索する）の順で行われる。コア・イメージには展開の順序は示していないが、これを使って具体的な展開の順序を説明することができる。

配列の値を明示しているのは、全体的な展開とともに、ピボット（アンダーラインの数値）による配列の領域分割アルゴリズムの理解も視野に入れているからである。学生には、このコア・イメージを自分で作成できるようになることが求められる。アルゴリズム構造の理解がコア・イメージの理解とリンクする典型的な例と考えてよい。

クイックソートは、現在、最高速のアルゴリズムと言われる。その主な要因は、配列を集合的に扱うことと、領域分割アルゴリズムによりデータ交換するときのデータ移動を大きくできることである。このことも理解させる必要があるが、コア・イメージには、このことも考慮されている。配列の集合的な扱いについては、図 3-2 に示す配列の扱い方の型構造を参照してもらいたい[3]。クイックソートの時間計算量は O(n log n)で、単純選択ソート、単純交換（バブル）ソート、単純挿入ソートなどが $O(n^2)$であることは、この配列の扱い方の型構造の違いから説明できる。

このコア・イメージと後述するクイックソート用プログラム展開イメージ図（図 3-4）をリンクさせると、再帰的アルゴリズムに基づく再帰呼び出しが具体的に追跡できるので、このコア・イメージは再帰的アルゴリズムの具体的な理解にも使える。

(3) 再帰的アルゴリズムの基本的な考え方とコア・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 3-3 は、再帰的アルゴリズムのコア・イメージである。この図では、全体の問題が A、B、C の 3 つの部分問題からなり、問題 C（C1、C2）が元の問題と同じ構造となる部分問題であることを表している。この条件のもとでアルゴリズム構造としての再帰呼び出しがわかればよい。再帰呼び出しが一つ深まるごとに、部分問題のレベル番号を加算するようにしてある。これは、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）が異なることを示すためである。

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、実体は自分と同じ関数（コピー関数）を呼び出すことであるので、このことが捉えられるように作成されている。要するに、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ことが理解できれば良いので、そのことが意図されている。

図 3-3 は、クイックソートの処理を再帰的アルゴリズムで展開したクイックソート用プログラム展開イメージ図である。この図は、クイックソートの具体的なアルゴリズムと再帰的アルゴリズムのコア・イメージから作成できる。①～④は、前述したアルゴリズムの番号と対応している。前述したように、この図 3-3 と図 3-1 のクイックソートのコア・イメージを併用すれば、再帰的アルゴリズムによる処理の展開を具体的に追跡できる。ここでは、詳細は省略するが、２つのコア・イメージを使って具体的な引数による再帰呼び出し説明することによって、再帰的アルゴリズムの理解も高まると考えられる。

(4) クイックソートのプログラム

図 3-5 は、上述したアルゴリズムに基づいて作成されたプログラム[6]である。プログラム中の①～④の番号は、前述したアルゴリズムの番号と対応している。再帰呼び出しは、③と④の部分である。このプログラムは、処理される配列のデータの流れに焦点を当てた図 3-1 のコア・イメージではなく、プログラムの構造に焦点を当てた図 3-4 のプログラム展開イメージ図を参照した方がよい。

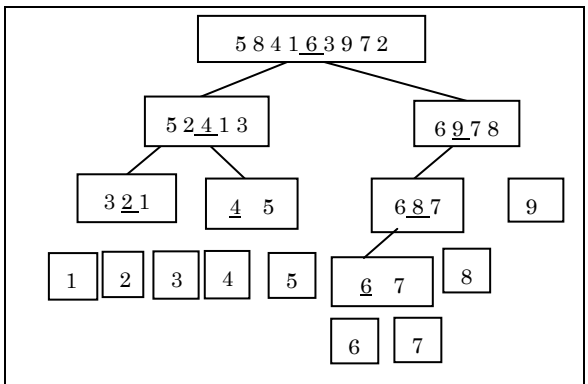

図 3-1 クイックソートのコア・イメージ

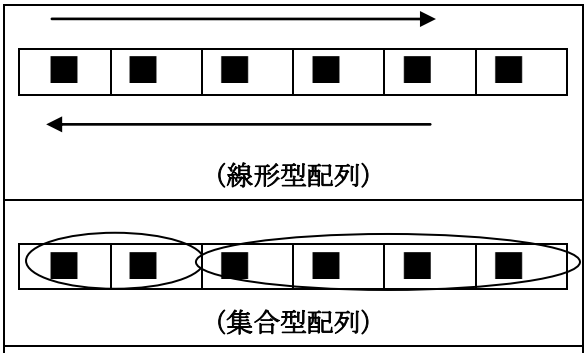

図 3-2 配列の扱い方の型構造

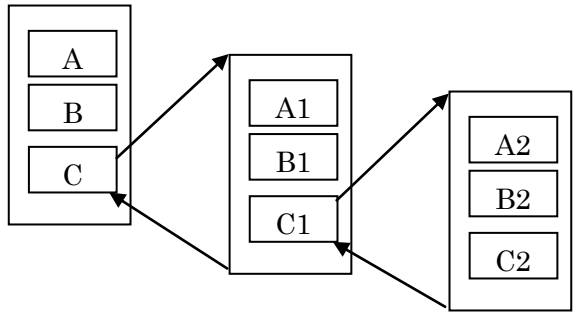

図 3-3 再帰的アルゴリズムの
コア・イメージ

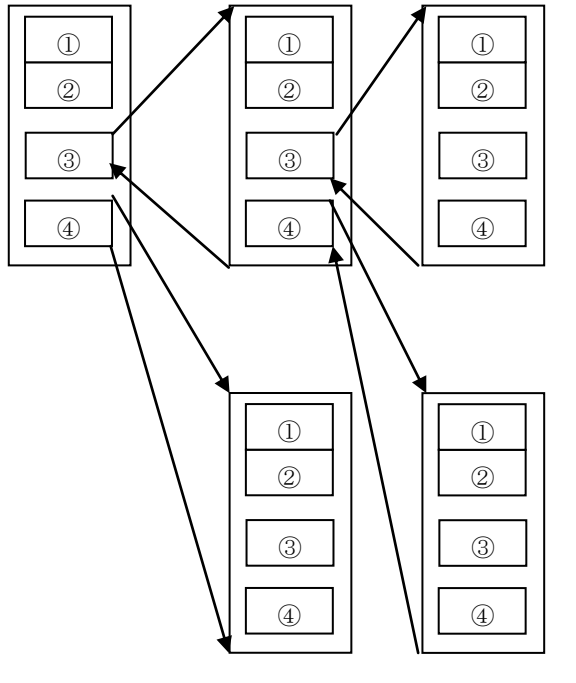

図 3-4 クイックソート用
プログラム展開イメージ図

```
void quick(int a[], int left, int right)
{
      int pl = left, pr = right, temp;
①     int x = a[(pl+pr)/2];
②     do {
            while (a[pl] < x) pl++;
            while (a[pr] > x) pr--;
            if (pl <= pr) {
                  temp = a[pl];
                  a[pl] = a[pr];
                  a[pr] = temp;
                  pl++;    pr--;
            }
      } while (pl <= pr);
③     if (left < pr)    quick(a, left, pr);
④     if (pl < right) quick(a, pl, right);
}
```

図 3-5 クイックソートのプログラム例

3.2 数学（算数）的アルゴリズムのコア・イメージの応用例

学生の多くは、「鶴亀算」や「濃度算」など「～算」と聞いても、文章による問題の内容のみが頭に浮かぶだけで、アルゴリズム構造としての知識を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アルゴリズムを思い出しながら問題を解き始める。もちろん、このようなケースは、「～算」という問題だけでなく、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、「～算」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はなく、そのコア・イメージにより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試みている。ここでは、鶴亀算のコア・イメージを応用したアルゴリズム教育を紹介しながら、鶴亀算のアルゴリズム構造の理解について考察する。

(1) 鶴亀算の基本的な考え方とコア・イメージ

鶴亀算は、「単位量が異なる２つのものがあり、２つを合わせた個数と全体量（合計）が与えられ、それぞれの個数を求める」という問題である。例えば、「鶴と亀が合計 20 羽(匹)いる。足の数は全部で 72 本あった。鶴と亀は何匹いるか？」という問題である。一般的に

は、方程式を使って解くが、アルゴリズム構造の理解という観点から、方程式を使わない方法を中心に扱っている。もちろん、方程式で解く方法についても説明はする。

まず、代表的な鶴亀算で、上述した「鶴と亀が合計 20 羽（匹）いる。足の数は全部で 72 本あった。鶴と亀は何匹いるか？」という問題で考える。問題の背景は、「2 つの異なるものの単位量（この例では暗黙の量となっている）、全体の個数、全体量の 4 つの値を使って、鶴と亀のそれぞれの個数を求める」ということである。アルゴリズムの基本的な考え方は、まず、単位量の差と、すべて鶴であると仮定した場合の量と実際の全体量の違いに注目して、亀の数を求めることである。ただし、すべて亀と仮定した考え方もある。

図 3-6 は、鶴亀算のコア・イメージである。まず、与えられた 4 つの値と求める 2 つの値の関係が、構造的に理解できればよい。図の縦軸方向が単位量、横軸方向が個数を表し、図の中央の太線を中心に左が鶴、右が亀の個数を表す。単位量と個数の単位は、それぞれ「数／個 : 足の数」「個 : 頭の数」であり、「単位量×個数」が四角の個数になり、足の数に対応する。

この図は、鶴亀算のアルゴリズム構造の一般的な形で、簡単に言えば逆 L 字の構造である。２つの異なる単位量のものが、それぞれ左右の位置に分かれて一列に並ぶ形である。

(2) 鶴亀算のアルゴリズムと解答例

上述した基本的な考え方に基づいた具体的なアルゴリズムは、次のようになる。以下では、単位量（単位あたりの足の数）の小さい方が鶴、大きい方が亀である。

①単位量差
　２つのものの単位量（単位あたりの足の数）の差を、単位量差として求める。
②仮の全体量（図の(A)部分の足の数）
　すべて小さい方（鶴）と仮定して、仮の全体量を求める。
③仮の差（図の(B)部分の足の数）
　仮の差＝与えられた全体量－仮の全体量
④大きい方（亀）
　大きい方＝仮の差 ÷ 単位量差
⑤小さい方（鶴）
　小さい方＝全体－大きい方

(3) アルゴリズムに基づく解

上述したアルゴリズムに具体的な数値を代入して解を求めると、次のようになる。

①（単位量差）4－2＝2
②（仮の全体量） 2×20＝40
③（仮の差）72－40＝32
④（大きい方：亀）32÷2＝16
⑤（小さい方：鶴）20－16＝4

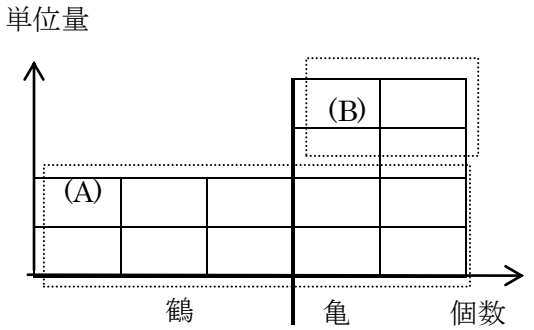

図 3-6 鶴亀算のコア・イメージ

(4) 鶴亀算の例 2

次に、「150 円と 180 円のボールペンを合計 30 本買い、5100 円払った。150 円と 180 円のボールペンをそれぞれ何本買ったか？」という問題について考える。図 3-7 は、図 3-6 のコア・イメージから作成される、この問題のイメージ図である。図 3-6 のコア・イメージとの違いは、与えられる単位量と全体量であり、ともに個数ではなく、量としてのお金である。この違いはアルゴリズムとしての違いにはならないので、この問題が鶴亀算であることがこの図から判断できる。

大きい方が 180 円、小さい方が 150 円、全体量が 5100 円であることに注意して、アルゴリズムにしたって解けば、次のようになる。

①（単位量差）180－150＝30
②（仮の全体量） 150×30＝4500
③（仮の差）5100－4500＝600
④（大きい方：180 円）600÷30＝20 (本)
⑤（小さい方：150 円）30－20＝10 (本)

図 3-6 と図 3-7 を比較すると、アルゴリズム構造（問題の構造）が全く同じであることがわかる。さらに、この図 3-7 を図 3-8 のように書くと、鶴亀算の問題は、簡単な図形のアルゴリズム構造であることがわかる。つまり、全体量を面積と考え、逆 L 字全体の面積と太線で示した線分の長さが与えられているとき、矢印の 2 つの長さを求めるという問題である。このような捉え方ができるのも、問題をイメージ化してアルゴリズム構造を明確

した結果であると考えられる。

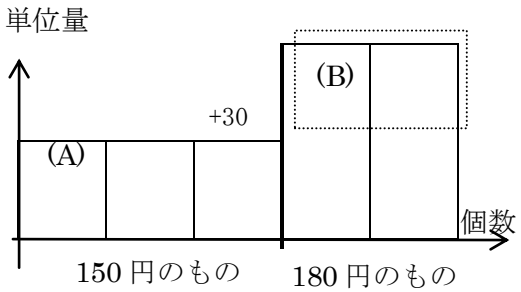

図 3-7 鶴亀算のイメージ図 1

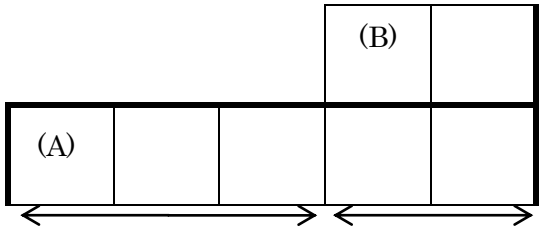

図 3-13 鶴亀算のイメージ図 2

4 まとめ

従来のアルゴリズム教育では、アルゴリズム構造の理解は、それ程意識されていない。本研究では、アルゴリズム構造の理解をそのコア・イメージの理解と位置づけ、コア・イメージ使った授業を試みている。この試みの成果の厳密な評価・検証は行っていないので、これは今後の課題としたい。

これまでの取り組みから言えることをいくつか挙げると、次のようになる。

①アルゴリズム構造を直観的に理解できるコア・イメージの作成を工夫すると、その構造が明確になり、アルゴリズムの説明もしやすくなる。

②コア・イメージを描けることを、アルゴリズムの本質的な理解につなげられる場合がある。例えば、クイックソートやマージソートでは、コア・イメージを正確に書けることがアルゴリズムの本質的な理解になると考えられる。

③再帰的アルゴリズムを使ったプログラミング的アルゴリズムの場合、そのコア・イメージと再帰的アルゴリズムのコア・イメージをリンクさせると、捉えにくい再帰的アルゴリズムによるデータの展開が具体的に把握できる。

④プログラミング的アルゴリズムの場合、プログラムの構造またはデータの流れのどちらに焦点を置くかによって、異なるコア・イメージが考えられる。目的によって使い分ければよいが、併用することを考えてもよい。

⑤鶴亀算が図形の問題として扱えることは、特に新しい考え方ではないが、コア・イメージの作成から必然的に意識されるようになることは、注目すべきことである。

今後の課題として、次のようなことが考えられる。

①コア・イメージを利用したアルゴリズム教育の成果の評価・検証を行う。

②検証の結果を踏まえ、コア・イメージのアルゴリズム教育への体系的な利用を考える。新しいアルゴリズム教育の提案ができる可能性がある。

③コア・イメージ作成のためのイメージ化の方法をさらに研究し、アルゴリズム構造の明確化についての方法論としてまとめる。

④アルゴリズム構造に着目し、アルゴリズムの類型化を図る。

参考文献

(1) 川口順功, 渡邉志, 高橋等,「言葉の理解を意識した情報教育の試み」, 平成 22 年度情報処理教育研究集会論文集, pp. 282-285, 2010

(2) 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

(3) 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

(4) 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

(5) 田中茂範, 佐藤芳明, 河原清志,「イメージでわかる単語帳」, p. 86, 日本放送出版協会, 2007

(6) 柴田望洋, 辻亮介,「C 言語によるアルゴリズムとデータ構造」, ソフトバンクパブリッシング, 2004