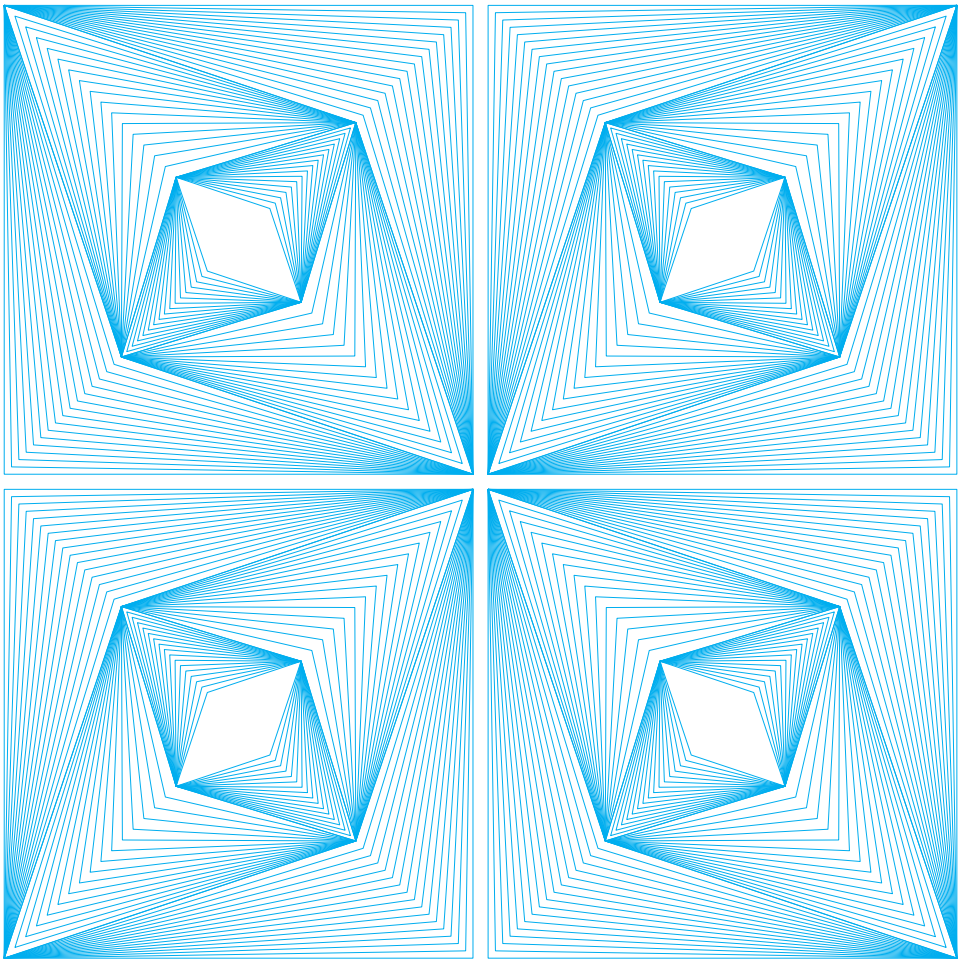
基礎から学ぶ
データ構造とアルゴリズム

Basic Knowledge of Data Structures and Algorithms

穴田有一・林　雄二
［著］



共立出版

第1章
データ構造とアルゴリズムの基本

データ構造とアルゴリズムはプログラミングの基礎です．また，よいプログラムを作るためには，効率よくデータを処理するアルゴリズムを採用する必要があります．この章では，データ構造とアルゴリズムの関係や計算量について説明します．

1.1 データ構造とアルゴリズムの基本

1.1.1 データ構造とアルゴリズムの関係

■コンピュータで問題を処理する

まずコンピュータで問題を処理する仕組みを復習しておきましょう．図1.1のように，コンピュータは一般的に演算制御装置（CPU：Central Processing Unit）と主記憶装置（メモリ）および補助記憶装置（ハードディスクなど）などから構成されています．みなさんがプログラムやデータを入力して，コンピュータに何か問題を処理させるとしましょう．入力されたプログラムやデータは主記憶装置に記憶され，随時演算制御装置に送られて実行されます．また，これらは必要に応じて補助記憶装置に保存されることもあります．すなわち，入力されたデータは，まず記憶装置に保持され，その後計算などの処理を施され，最終的にディスプレイやプリンタに出力されます．この一連の流れの中で，データが記憶装置に保持されるときデータ構造が問題になります．また，計算などの処理を行うときにはアルゴリズムが問題になります．みなさんがコンピュータで問題を処理するときには，その各段階でデータ構造やアルゴリズムがかかわっています．

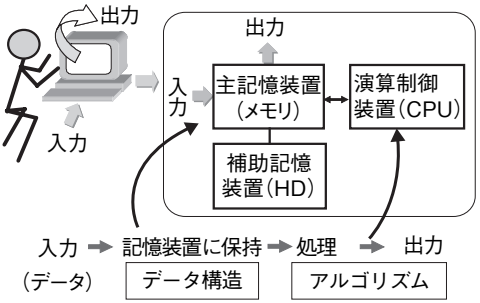


図1.1　コンピュータによる処理とデータ構造とアルゴリズム

■アルゴリズムとは何か

アルゴリズム（algorithm）という言葉はプログラミングや情報科学に限らず，広くさまざまな場面で使われます．アルゴリズムという用語を一般的に定義すると，「問題を解決する手順」ということになります．したがって，マラソン大会でランナーをタイムが小さい順に並べる方法はアルゴリズムですし，数学で連立方程式を解くための方法として使われる「代入法」や「消去法」もアルゴリズムです．また，カレーライスの作り方のように，料理を作る手順もアルゴリズムといえなくもありません．しかし，ここで学習するアルゴリズムとは，コンピュータで問題を処理するためにプログラムとして実現するのに向いている解法です．例を挙げると，グラフ理論，オペレーションズリサーチ，数値計算，データ構造の操作，整列，数式処理，言語処理などです．特に本書で扱う問題を具体的にいえば，データ構造の操作と整列です．これらのアルゴリズムについては，第2章以降で詳しく説明します．

本書で取り上げていないアルゴリズムにも有益なものが数多くあります．ここでは，一つだけ有名なアルゴリズムを挙げておきましょう．それは紀元前300年頃の古代ギリシャの学者ユークリッドの著書「幾何学原論（Elements）」に書かれているもので，二つの正の整数の最大公約数を求めるアルゴリズムです．これは次のような手順で表されます．

① m，n を正の整数とする．

② $m \div n$ の余りを r とする．$(0 \leqq r < n)$

③　(n, r)を新しい m，n とする．

④　$r=0$になるまで②，③を繰り返す．

⑤　$(n, 0)$の n が最大公約数

実際に数字を当てはめてみましょう．① $m=120$，$n=32$とすると，② $m \div n$ の余り r は24になります．そこで，③ 新しい m，n を $m=32$，$n=24$とします．④ $r=0$になるまで②，③を繰り返すと，以下のようになります．

(m, n)	→	(n, r)
(120, 32)	→	(32, 24)
(32, 24)	→	(24, 8)
(24, 8)	→	(8, 0)

こうして，得られた $n=8$が最大公約数となります．

■データ構造とは何か

データ構造（data structure）とは，データ（data）が相互に関連づけられた構造（structure）を形作ったものをいいます．データをブロックに例えればデータ構造とはブロックを規則的に積み上げて作った建物といえます．バラバラに集められたブロックの集まりはデータ構造とはいわないのです．たとえば

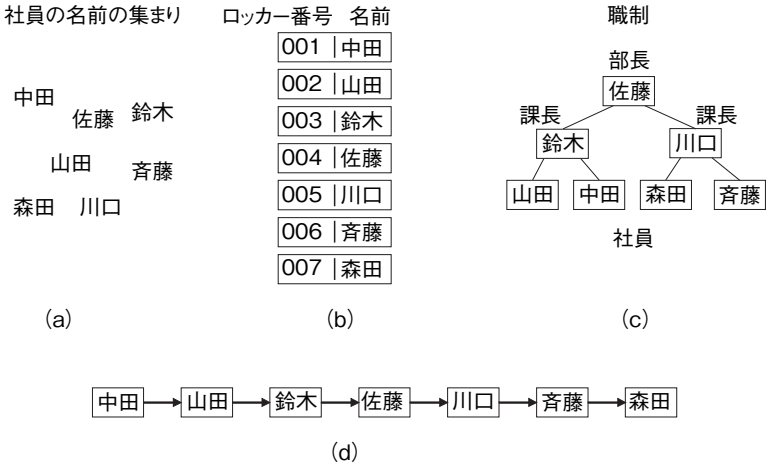


図1.2　社員の年齢データを例としたデータ構造

図1.2（a）のように，ある会社の社員の名前（データ）をただ単に集めたものは，データ構造とはいいません．名前は各社員の属性ですが，（b）のようにロッカールームに並んだ番号付きロッカーに社員の名前を割り当てると，名前データはロッカー番号によって相互に関連づけられた構造をもつことになります．また，職制によって（c）のように関連づけられた名前データもデータ構造です．このように同じ一組のデータでも，どのような構造に組み立てられるかで，異なるデータ構造になります．

ちなみに，（b）のようなデータ構造は配列であり，（c）は木構造といいます．また，ロッカールームでの名前の並びを（b）のようにロッカー番号で関連させるのではなく，隣合う社員の名前のつながりで表すと（d）のようになりますが，これは連結リストというデータ構造です．

■データ構造とアルゴリズム

実際の側面から見ると，データを処理する規則の集まりがアルゴリズムです．したがって，データの表現方法すなわちデータ構造が確定しなければ，アルゴリズムだけでは問題を解決することはできません．すなわち，アルゴリズムとデータ構造は密接に関係しています．コンピュータを使って実際に問題を解決するには，データ構造によって表現されたデータを適切なアルゴリズムによって処理するプログラムをつくる必要があります．まさにWirthの名言にあるように，「アルゴリズム＋データ構造＝プログラム」なのです．

効率のよいアルゴリズムを作るにはデータ構造の複雑さとアルゴリズムの複雑さのバランスが重要です．とくに**時間**（処理の速さ）と**空間**（記憶域の大きさ）の**トレードオフ**（Trade-off）は重要で，プログラミングに際してアルゴリズムとデータ構造を選ぶときに十分考慮すべき点です．トレードオフの一般的な意味は，取引とか交換ということであり，相手が自分にとって必要なものをもっているときに，自分がもっている大事なものを提供してそれを手に入れることです．自分の大事なものをすべて差し出して相手のものを全部手に入れるか，それとも一部を差し出して相手からも一部をもらうかは，交換で得られる利得の評価によって異なります．アルゴリズムの場合には，1.1.3項で述べる計算量の評価で時間と空間のトレードオフが決まります．

1.1.2　アルゴリズムの表現

人間の創造活動は「表現」を通じて実現します．人間が他の動物と区別される大きな特徴である発明・発見・工夫による創造活動は，人間が高度な言語を獲得したために盛んになりました．すなわち，高度な「言語」で思考を「表現」できたから創造活動が発展したのです．アルゴリズムにも「表現」するための道具が必要です．

JIS の定義によると，アルゴリズムとは「明確に定義された有限個の規則の集まりで，有限回適用することにより問題を解くもの」とあります．したがって，アルゴリズムは文章や図で表現できることになります．しかし最終的には，アルゴリズムをプログラム言語で記述してコンピュータに問題を解かせることになります．アルゴリズムの表現には次のようなものが使われます．

・手続きを時間順に箇条書きする．
・フローチャート，PAD などの図を用いる．
・擬似言語で記述する．
・プログラミング言語で記述する．

1.1.3　アルゴリズムと計算量

■計算量とは何か

計算量（computational complexity）はアルゴリズムの性能を表現するために使われ，使用する計算時間や記憶領域の大きさで表します．時間と空間のトレードオフを考える上でも重要です．計算時間，記憶領域の大きさに関する計算量は，具体的には次のように評価されます．

・**時間計算量**（time complexity）：ある特定の操作の操作回数
・**領域計算量**（space complexity）：変数の数など

先に述べた時間と空間のトレードオフは，時間計算量と領域計算量から考えます．ただし，コンピュータに問題を処理させるときに実際上重要なのは，記憶領域の使用量もさることながら，処理時間です．したがって通常，計算量と

いえば時間計算量を指します．また，n 個のデータをあるデータ構造に格納して，あるアルゴリズムで処理する場合，データが同じでも，どのような状態で格納されているかによって，数回の操作で処理が終了したり，n 個のデータを隈なく操作することになったりします．したがって，計算量を正確に評価することは簡単なことではありません．

そこで，実際に計算量を求めるときには，**最悪計算量**（worst case complexity）や**平均計算量**（average case complexity）を求めることになります．最悪計算量とは，あるアルゴリズムの計算量が大きさ n のデータの格納状態または入力される順序によって異なるときに，最も手間がかかる場合の計算量のことをいいます．平均計算量は，n の入力データがある確率分布をもつとして，その確率分布について平均をとって求めます．実用上は平均計算量が重要な場合も多いのですが，評価するのが難しい場合も多いのが実状です．

■ O 記法

計算量を評価する際に基礎になるのは，各処理の実行回数です．これは処理に要する時間に相当します．しかし，これをそのまま計算量として用いるのではなく，十分大きなデータを処理した場合にどの程度の時間すなわち手間がかかるかを求めます．たとえば，時間計算量が$3n^2$で処理できるアルゴリズムを用いるときに，$n=100$のデータを処理する場合に比べて，$n=200$のデータを処理する場合は何倍の時間がかかるでしょうか．これを次のように計算してみると

$$\frac{3\times 200^2}{3\times 100^2}=\frac{200^2}{100^2}=4$$

4倍になることがわかります．ただし，この計算を見るとわかるように，入力データの大きさ n の値によって何倍の時間がかかるかを問題にするときには，$3n^2$の係数「3」は意味をもちません．したがって，このアルゴリズムの計算量は「n^2」の程度と考えてよいでしょう．

このように見積もる計算量は**漸近的計算量**（asymptotic complexity）と呼ばれます．漸近的計算量を表す方法はいくつかありますが，普通は **O 記法**が用いられます．O 記法をもっと正確に定義すると次のようになります．

二つの関数 $f(n)$，$g(n)$について

$$f(n) \leqq cg(n),\quad n > n_0$$

を満足する正の定数 c，n_0が存在するとき，$f(n)$は**オーダー** $g(n)$であるといい，次のように書く．

$$f(n) = O(g(n))$$

これをグラフで説明しましょう．図1.3は，$3n^2+4n+1000$と n^2と$4n^2$の n に対する変化を示しています．単に$3n^2+4n+1000$の漸近的な関数を求めるのであれば，n^2のほかにも n^2+n や$3n^2$などいろいろ考えることができます．しかし，ここで述べたオーダーの定義によると，「$f(n)$は高々 $g(n)$の定数倍にしかならない」ということです．図1.3を見ると，n_0よりも大きな n の値に対して $f(n)=3n^2+4n+1000$は n^2よりも大きくなっていますが，たとえば$4n^2$よりも小さいことがわかります．このことから，$f(n)=3n^2+4n+1000$は n^2を適当な定数（c）倍したものの値を超えないことがわかります．すなわち，「$f(n)$は高々 $g(n)$の定数倍にしかならない」ということです．

O 記法の定義を今度は計算で確認してみましょう．ある関数 $f(n)$に対して，$\lim_{n\to\infty}\frac{f(n)}{g(n)}$ が収束するとき，$f(n)$の n の次数の最大のものは，$g(n)$の n の次数と等しくなります．もう少し詳しく説明すると，$f(n) \leqq cg(n)$が成り立つときには$\frac{f(n)}{g(n)} \leqq c$が成り立つので，$\lim_{n\to\infty}\frac{f(n)}{g(n)} \leqq c$となり，収束することになります．

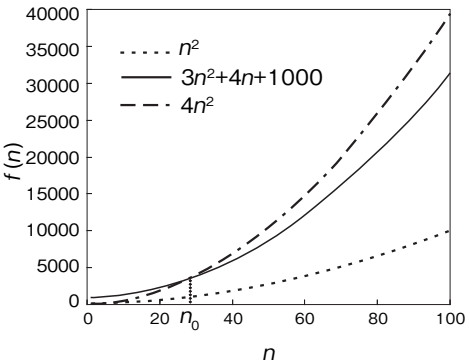


図1.3　$f(n)$と $g(n)$の n に対する変化

このとき，$g(n)$は$f(n)$のオーダーであるといい，式では$f(n)=O(g(n))$と表します．例を挙げて説明しましょう．

例1.1　$f(n)=3n^2+4n+1000$のオーダー

$g(n)=n^2$とすると，

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{3n^2+4n+1000}{n^2}$$

$$=\lim_{n\to\infty}\left(\frac{3n^2}{n^2}+\frac{4n}{n^2}+\frac{1000}{n^2}\right)$$

$$=\lim_{n\to\infty}\left(\frac{3n^2}{n^2}\right)+\lim_{n\to\infty}\left(\frac{4n}{n^2}\right)+\lim_{n\to\infty}\left(\frac{1000}{n^2}\right)$$

$$=3+0+0$$

$$=3$$

すなわち，$g(n)=n^2$としたとき$\frac{f(n)}{g(n)}$は収束し，$f(n)=O(n^2)$としてO記法の定義を満たします．試しに$g(n)=n$として，上と同じ計算をしてみてください．$\frac{f(n)}{g(n)}=4+3n$となって発散することがわかります．オーダーについて，他の例をあげておきます．

$f(n)$	$g(n)$	
n	n	$\mathrm{O}(n)$
n^2+3	n^2	$\mathrm{O}(n^2)$
$3n^2+4n+1000$	n^2	$\mathrm{O}(n^2)$

問題1.1　実行回数が以下の関数で与えられているとき，それぞれのオーダーを求めよ．

(1)　n

(2)　$4n+1$

(3)　n^2+3n+8

■ O 記法の和と積

アルゴリズムの計算量を評価するときに，全体のアルゴリズムを部分のアルゴリズムの和や積で求めることができます．ここでは，**O 記法の和と積**につ

いて説明します．計算量と漸近的計算量が次の式で表されるとき，

$$f_1(n)=O(g_1(n))$$
$$f_2(n)=O(g_2(n))$$

$f_1(n)$と $f_2(n)$の和は次のように計算されます．

$$f_1(n)+f_2(n)=O(\max(g_1(n), g_2(n)))$$

ここで，$\max(g_1(n),\ g_2(n))$は $g_1(n)$，$g_2(n)$のうちオーダーの大きいほうをとるという意味です．積は次のように計算されます．

$$f_1(n)\cdot f_2(n)=O(g_1(n)\cdot g_2(n))$$

すなわち，積のオーダーは単純に $g_1(n)$と $g_2(n)$をかけたものです．以下に計算の例をあげておきます．

$f_1(n)=2n$　　のとき　$g_1(n)=n$
$f_2(n)=3n^2+4$　のとき　$g_2(n)=n^2$

和　$f_1(n)+f_2(n)=O(g_1(n))+O(g_2(n))=O(\max(g_1(n)\cdot g_2(n)))$
$\quad=O(\max(n, n^2))=O(n^2)$
積　$f_1(n)\cdot f_2(n)=O(g_1(n))\cdot O(g_2(n))=O(g_1(n)\cdot g_2(n))$
$\quad=O(n\cdot n^2)=O(n^3)$

問題1.2　実行回数を表す関数 $f(n)$，$g(n)$，$h(n)$，$i(n)$が以下のように与えられているとき，(1)～(5)を求めよ．ただし，n はデータの個数で，非常に大きいとする．

$f(n)=1$，$g(n)=n$，$h(n)=n^2$，$i(n)=\log n$

(1)　$O(f(n))+O(g(n))$
(2)　$O(g(n))\cdot O(h(n))$
(3)　$O(g(n))+O(h(n))+O(i(n))$
(4)　$O(f(n))\cdot O(g(n))+O(i(n))$
(5)　$(O(f(n))+O(g(n)))\cdot O(i(n))$

では，実際のアルゴリズムで和と積はどのような関係になっているのでしょうか．プログラミングで指令（コマンド）を組み合わせるとき，アルゴリズムの基本として順次，選択，反復の三つがあります．ある処理を行う単位としてのアルゴリズムの組合せも同様です．このときの順次と反復に相当するのが和と積になります．図1.4に示すように，二つのアルゴリズムA，Bが順に実行されるとき，これらの全体のオーダーは各アルゴリズムのオーダーの和になります．また，アルゴリズムAが反復されるときには全体のオーダーは積で求められます．なおこの図で，$f_1(n)$，$f_2(n)$はそれぞれの部分の実行時間を表しています．

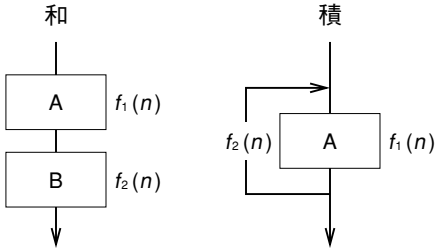


図1.4　オーダーの和と積

問題1.3　n 個のデータを処理するアルゴリズムがある．このアルゴリズムは処理Aと処理Bの部分から構成されている．処理Aの時間計算量は $O(n)$，処理Bの時間計算量は $O(\log n)$である．次のそれぞれの場合について，アルゴリズム全体の時間計算量のオーダーを求めよ．

(1)　処理Aが終わってから，処理Bが実行される．

(2)　処理Bを n 回繰り返す．

■よく現れるオーダーの例

ここで扱うほとんどのアルゴリズムの実行時間は次のオーダーのどれかで表されます．また，上から下へ行くに従ってオーダーは大きくなります．