第5章　プログラミングの基礎

5.1　アルゴリズム

ある問題を解決するための手段をアルゴリズム[1] (algorithm) といいます。すでに紹介した、10進数を r 進数に変換する手順や、シフト演算と足し算のみで掛け算や割り算を行なう手順も、問題を解決するためのアルゴリズムと言えます。以下に 3 つのアルゴリズムを紹介しておきます。

5.1.1　平方根

正の整数 N (>0) の平方根 $\sqrt{N}$ の近似値を求めるアルゴリズムを考えてみましょう[2]。ここでは、奇数列の和が

$$1+3+\cdots+(2\cdot n-1)=\frac{\{1+(2\cdot n-1)\}\cdot n}{2}=n^2$$

であることを利用し、正の整数 N $(=n^2)$ の平方根 $\sqrt{N}$ $(=n)$ の近似値を求めるアルゴリズムを紹介します。例として、5 の平方根の近似値を求めてみましょう。

$$5=(1+3)+1=\{1+(2\cdot 2-1)\}+1=2^2+1$$

より、5 の平方根の整数部分は 2 となります。小数第 1 位まで求めるには、

$$100\cdot N=(10\cdot n)^2=(n')^2 \Longleftrightarrow \sqrt{N}=n=\frac{n'}{10}$$

[1] 紀元 9 世紀、バクダッドのアル・フワーズリズミーという数学者が書き著した「キターブ・アルジャブル・ワ・ルムカーバラ (移項と消去の計算書)」という書名に由来して、algebra (代数学) という言葉が生まれました。また、この数学者の名前 アル・フワーズリズミーが変化してアルゴリズムという言葉になったといわれています。

[2] 正の整数に限らず、正の実数であれば、このアルゴリズムを用いて平方根の近似値を求めることができます。

より、両辺を 100 倍したものを

$$\begin{aligned} 100 \cdot 5 = 500 &= (1 + 3 + \cdots + 41 + 43) + 16 \\ &= \{(1 + 3 + \cdots + 41 + (2 \cdot 22 - 1)\} + 16 = 22^2 + 16 \end{aligned}$$

のように奇数列の和に変形し、22 を 10 で割れば 5 の平方根の近似値 2.2 が求められます。同様に、小数第 2 位まで求めるには、

$$10000 \cdot N = (100 \cdot n)^2 = (n'')^2 \Longleftrightarrow \sqrt{N} = n = \frac{n''}{100}$$

より、両辺を 10000 倍したものを

$$\begin{aligned} 10000 \cdot 5 = 50000 &= (1 + 3 + \cdots + 443 + 445) + 271 \\ &= \{1 + 3 + \cdots + 443 + (2 \cdot 223 - 1)\} + 271 = 223^2 + 271 \end{aligned}$$

と変形することで、223 を 100 で割れば 5 の平方根の近似値 2.23 が求められます。

ただし、上記の方法だと求める桁数が増えるにしたがってかなりの数の奇数を足さなければならず、効率的とはいえません。そこで、下記のようにすると平方根の近似値を効率的に求めることができます。まず、

$$5 = (1 + 3) + 1 = \{1 + (2 \cdot 2 - 1)\} + 1 = 2^2 + 1$$

より、5 の平方根の整数部分 2 を求めます。次に、両辺を 100 倍し、

$$\begin{aligned} 5 \cdot 100 = (2 \cdot 10)^2 + 100 &= 20^2 + (2 \cdot 21 - 1) + (2 \cdot 22 - 1) + 16 \\ &\quad \uparrow \\ &\quad 1 + 3 + \cdots + 37 + (2 \cdot 20 - 1) \\ &= 22^2 + 16 \end{aligned}$$

と変形することで小数第 1 位まで近似値が求められます。更に、両辺を 100 倍し、

$$\begin{aligned} 500 \cdot 100 = (22 \cdot 10)^2 + 1600 &= 220^2 + (2 \cdot 221 - 1) + (2 \cdot 222 - 1) + (2 \cdot 223 - 1) + 271 \\ &\quad \uparrow \\ &\quad 1 + 3 + \cdots + 337 + (2 \cdot 220 - 1) \\ &= 223^2 + 271 \end{aligned}$$

と変形することで小数第 2 位まで近似値が求められます。同様に、この操作を繰り返せば 5 の平方根の近似値を必要な桁数まで効率的に求めることができます (各桁の値を決定するのに、高々 10 回の奇数の引き算で済む)。

問題 1　$1 + 3 + \cdots + (2 \cdot n - 1) = n^2$ を利用して、7 の平方根を小数第 3 位まで求めなさい。

問題 2　$1 + 3 + \cdots + (2 \cdot n - 1) = n^2$ を利用して、12345 の平方根をを小数第 3 位まで求めなさい。

問題 3　$1 + 3 + \cdots + (2 \cdot n - 1) = n^2$ を利用して、0.00003 の平方根をを小数第 5 位まで求めなさい。

5.1.2 最大公約数と最小公倍数

2 つの正の整数 x, y $(x \geq y > 0)$ の最大公約数 (greatest common divisor; gcd) と最小公倍数 (least common multiple; lcm) を求めるアルゴリズムを考えてみましょう。初等的な代数学を学習したみなさんであれば、**ユークリッドの互除法**という有名なアルゴリズムを用いて最大公約数が求められることを知っていることでしょう。ユークリッドの互除法は、関数

$$\gcd(x,\ y) = \begin{cases} x & ,y = 0 \text{ のとき}, \\ \gcd(y,\ x \bmod y) & ,y \neq 0 \text{ のとき} \end{cases}$$

によって定義され、$y = 0$ になるまで繰り返し関数 gcd() を計算すれば最大公約数が求められます[3]。ただし、$x \bmod y$ は x を y で割った余り (剰余) を計算します。また、最小公倍数は

$$\mathrm{lcm}(x,\ y) = (x \times y) \div \gcd(x,\ y)$$

を計算することで求められます。例として、$x = 1234$ $(= 2 \cdot 617)$ と $y = 56$ $(= 2^3 \cdot 7)$ の最大公約数と最小公倍数を求めてみましょう。ユークリッドの互除法を繰り返し適用すると、最大公約数は

$$\begin{aligned} \gcd(1234,\ 56) &= \gcd(56,\ 1234 \bmod 56) = \gcd(56, 2) \\ &= \gcd(2,\ 56 \bmod 2) = \gcd(2, 0) \\ &= 2 \end{aligned}$$

となり、最小公倍数は

$$\mathrm{lcm}(1234,\ 56) = (1234 \times 56) \div \gcd(1234,\ 56) = 69104 \div 2 = 34552$$

となります。

問題 4 ユークリッドの互除法によって最大公約数が求められることを証明しなさい。

問題 5 ユークリッドの互除法を用いて $x = 3235$ と $y = 875$ の最大公約数と最小公倍数を求めなさい。

問題 6 問題 5 の x と y を入れ替え、ユークリッドの互除法を用いて $x = 875$ と $y = 3235$ の最大公約数を求めるとき、どのような事が起こるか考察しなさい。

問題 7 3 つの正の整数 x, y, z $(x \geq y \geq z)$ の最大公約数はどのようにすれば求められるか考察しなさい。

問題 8 3 つの正の整数 x, y, z $(x \geq y \geq z)$ の最小公倍数はどのようにすれば求められるか考察しなさい。

[3] ユークリッドの互除法のように、繰り返し関数を適用するようなアルゴリズムを**再帰的アルゴリズム**と呼びます。その他にも、$n!$ (n の階乗) を求めるアルゴリズムなどが再帰的アルゴリズムとしてよく紹介されます。

5.1.3　ハノイの塔

ハノイの塔は、フランスの数学者エドゥアール・リュカが 1883 年に発売したゲーム『ハノイの塔』がルーツとされています (図 5.1 参照)。また、このゲームのリーフレットには以下のような伝説が書かれていています。

ゲームのルール

- 3 本の杭と、中央に穴の開いた大きさの異なる複数の円盤から構成される。
- 最初はすべての円盤が左端の杭に小さいものが上になるように順に積み重ねられている。
- 円盤を 1 度に 1 枚ずつどれかの杭に移動させることができるが、小さな円盤の上に大きな円盤を乗せることはできない。

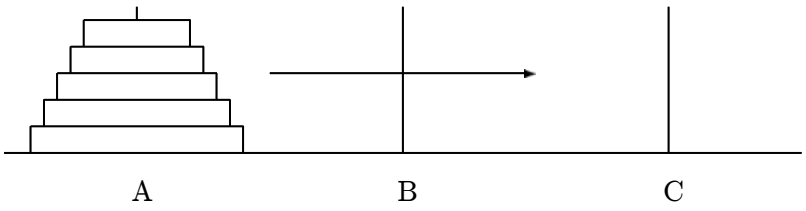

図 5.1: ハノイの塔

伝説　インドのガンジス河の畔のヴァラナシ (ベナレス) に、世界の中心を表すという巨大な寺院がある。そこには青銅の板の上に、長さ 1 キュビット、太さが蜂の体ほどの 3 本のダイヤモンドの針が立てられている。そのうちの 1 本には、天地創造のときに神が 64 枚の純金の円盤を大きい円盤から順に重ねて置いた。これが「ブラフマーの塔」である。司祭たちはそこで、昼夜を通して円盤を別の柱に移し替えている (移し変えのルールの説明は省略)。そして、全ての円盤の移し替えが終わったときに、世界は崩壊し終焉を迎える。【ウィキペディアより抜粋】

例えば、n 枚の円盤を使ってゲームを行うと、最低でも

$$2^n - 1 \text{ 回}$$

円盤を移動する必要があることが知られています (証明してみよう)。したがって、伝説のように 64 枚の円盤では $2^{64} - 1 = 18,446,744,073,709,551,615$ (1844 京 6744 兆 737 億 955 万 1615) 回も円盤を移す必要が出てきます。仮に、1 秒間に 1 枚の円盤を動かすことができたとしても、約 5,845 億年かかってしまいます。なお、現在の高速なコンピュータ (例えばクロック数が 4GHz で 8 コアの CPU を 1 個搭載したコンピュータ; 並列処理が可能で 1 枚の円盤を動かすのに 10 命令必要だと仮定する; その他のオーバーヘッドは考えない) を使っても、約 183 億年かかってしまい、たとえコンピュータが壊れなかったとしても私たちが生きている間にゲームを終わらせることはできません [4]。このように、このゲームには膨大な時間がかかるため、ゲームが終わる頃には世界が滅亡してしまうという実しやかな伝説となっています。

[4] 日本が誇るスーパーコンピュータ「京」だと、1 秒間に 1 京 ($= 10^{16}$) 回の演算が可能だから、1 枚の円盤を動かすのに 10 命令必要 (並列処理が可能で、その他のオーバーヘッドは考えない) だと仮定すると、約 5 時間ほどで終了することができる。

このゲームを最短で終わらせるための手順 (アルゴリズム) を考えて見ましょう。まず、図 5.2 のように大きく 3 つのステップに手順を分けます。

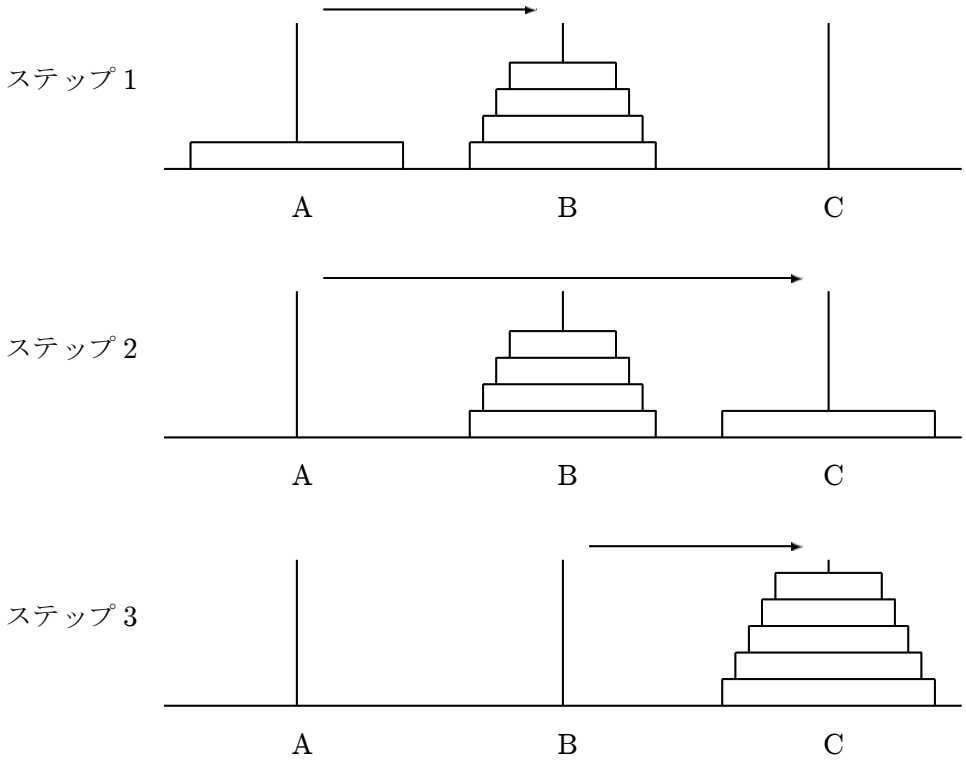

図 5.2: ハノイの塔のアルゴリズム

このとき、ステップ 1 とステップ 3 は、棒の位置は違いますが、$n-1$ 枚の円盤を移すハノイの塔のゲームに帰着できることがわかります。実際、ステップ 1 とステップ 3 を図 5.3 のように書き換えてみれば、その様子がよくわかります (ステップ 2 は円盤を動かす操作)。

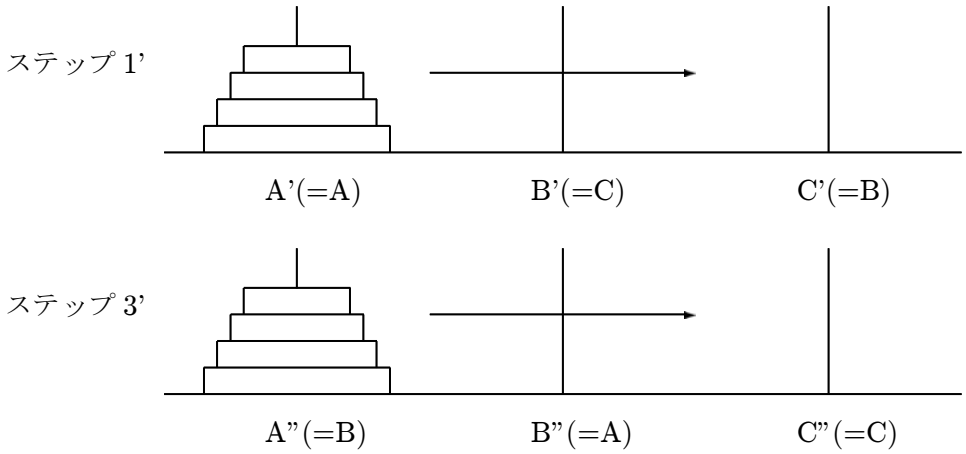

図 5.3: 帰着されたハノイの塔

このように、手順の中で同じ手順を繰り返し行うようなアルゴリズムを**再帰的アルゴリズム**と呼び、プログラミングにおいても**再帰的プログラム**の例として必ず取り上げられます。図 5.4 は、4 枚の円盤を最短の手順で移動させる様子を再帰的アルゴリズムで模式的に表したものです (これまで例として挙げてきた 5 枚の円盤の場合はもう 1 段増えることに注意してください)。なお、白い箱 □ は再帰的に呼び出される図 5.2 のステップ 1 とステップ 3 を表し、黒い箱 ■ は図 5.2 のステップ 2 を表しています。特に、黒い箱 ■ は、実際に円盤を動かす操作で、$2^4 - 1$ $(= 15)$ 回であることも確認することができます。

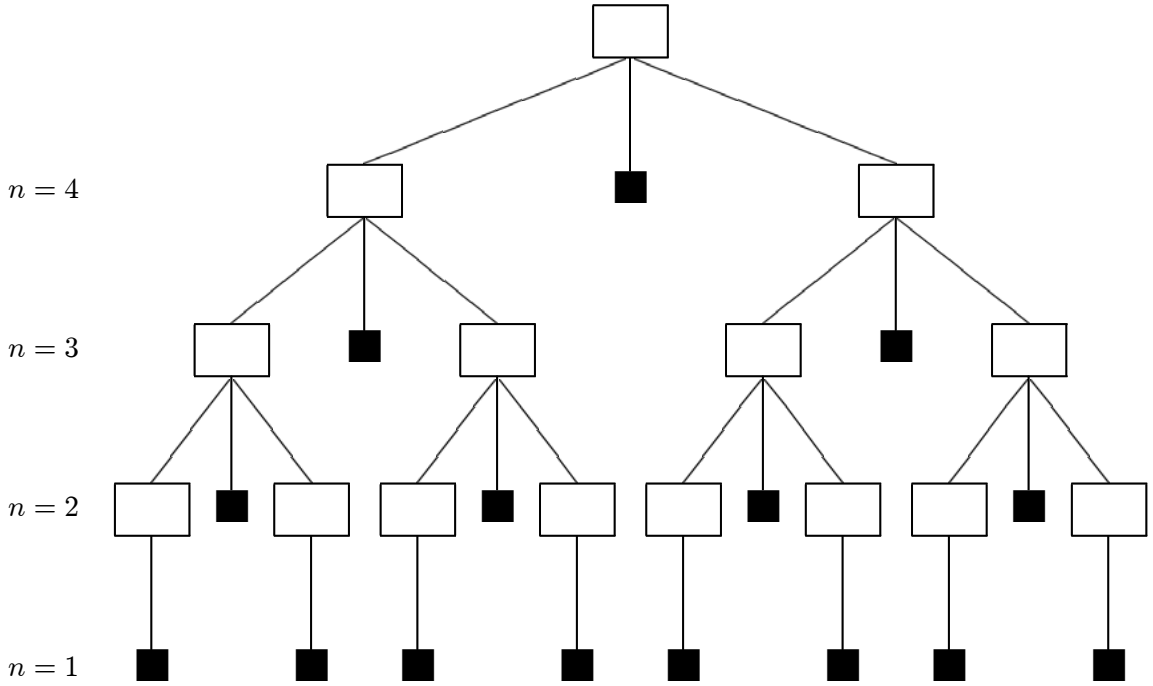

図 5.4: 再帰的アルゴリズムの展開 (ハノイの塔)

ただし、再帰的アルゴリズムは非常に便利な反面、図 5.4 からもわかるように階層を作りながらプログラムを進めていきます。そのため、実際のプログラムでは上位層の情報を記憶しておく必要があり、前節で紹介した**スタック領域**をたくさん使うことになります (階層が増えると指数関数的にスタック領域を使用する)。その結果、オペレーティングシステムから割り当てられたスタック領域を使い切ってしまい、プログラムが正常に実行できないことがあります (アルゴリズムが正しいからといってプログラムとして上手く動くわけではない)。したがって、再帰的アルゴリズムを用いたプログラムを作成する際は注意が必要なことを心に留めておいてください。