

## 2 アルゴリズムと計算量

アルゴリズムの計算量を解析する例として用いるために、まず挿入ソート (insertion sort) を紹介する。

### 挿入ソート

数列を大きさの順に並べ替えることをソート (整列) という。入力は  $n$  個の数の列  $a_1, a_2, \dots, a_n$  であり、出力は入力を並べ替えて得られる列  $a_{i_1} \leq a_{i_2} \leq \dots \leq a_{i_n}$  である。

より一般に、 $a_1, a_2, \dots, a_n$  はある全順序集合に属しているとしてもよい。すなわち、任意の  $i, j$  に対して、 $a_i < a_j, a_i = a_j, a_i > a_j$  のいずれか一つが成り立つとする。

以下では、ソートの入力は配列で与えられているとする。ここで、配列 (array) とは、データを一行に並べ、添字 (index) によって要素を指定できるようにしたデータ構造である。本講義では配列の添字は 1 から始まるものとする。

配列  $a[1], a[2], \dots, a[n]$  をソートするとき、挿入ソートのアルゴリズムは次の通りである。まず  $a[2]$  を  $a[1]$  の前か後ろの正しい位置に挿入する。このとき列  $a[1], a[2]$  はソートされている。次に、 $a[3]$  を列  $a[1], a[2]$  の中で正しい位置に挿入する。このとき列  $a[1], a[2], a[3]$  はソートされている。以下同様に繰り返す。

擬似コードでは次のように書ける。

```
1: INSERTIONSORT( $a[ ], n$ )
2:   for  $i = 2$  to  $n$ 
3:      $t = a[i]$ 
4:      $j = i - 1$ 
5:     while  $j > 0$  and  $a[j] > t$ 
6:        $a[j + 1] = a[j]$ 
7:        $j = j - 1$ 
8:      $a[j + 1] = t$ 
```

### 計算量

アルゴリズムの実行にかかる時間で評価した計算量を時間計算量 (time complexity)、実行に必要な領域 (メモリ) で評価した計算量を領域計算量 (space complexity) という。以下では主に時間計算量を取り扱い、時間計算量を単に計算量と呼ぶ。

時間計算量を評価する際には、基本的な計算のステップをいくつか定め、それを数えることで評価する。(各ステップの計算時間に差があると考えられる場合は、それも考慮して評価する。) ソートの場合には比較・交換・代入などを計算ステップとして考える。最大公約数のように整数を扱う場合には、本講義ではビット演算を計算ステップとする (第 9 回以降に詳しく説明する)。

計算量は入力サイズ (input size) の関数で評価される。入力サイズは問題によって定まる。例えば、ソートの場合には数列の長さ  $n$  を入力サイズとし、最大公約数  $\text{gcd}(a, b)$  を求める場合は  $a, b$  のビット長を入力サイズとする。

計算量を評価する際、サイズが等しい入力のうち、計算量が最も大きいもので計算量を評価することが多い。この計算量を最悪計算量 (worst-case complexity) という。また、入力は何らかの分布に従うと仮定して、計算量の平均によって評価することもある。この計算量を平均計算量 (average-case complexity) という。

挿入ソートの場合、計算量が最も大きいのは入力が逆順になっているときで、最悪計算量は  $O(n^2)$  である。また、入力が一様ランダムなとき、平均計算量は  $O(n^2)$  であることが知られている。(より正確には、最悪計算量、平均計算量ともに  $\Theta(n^2)$  である。)