

## 6 整列アルゴリズム (1)

以下では、配列  $a[1], a[2], \dots, a[n]$  を昇順に並び替えるものとして説明する。

### バブルソート

バブルソート (bubble sort) は、隣接している要素の順序が逆のとき交換する操作を繰り返すことで整列する。アルゴリズムは次のように書ける。ただし、 $\text{swap}(a[i], a[i+1])$  は  $a[i]$  と  $a[i+1]$  を交換する操作を表す。

```
1: BUBBLESORT( $a[ ], n$ )
2:   for  $j = 1$  to  $n - 1$ 
3:     for  $i = 1$  to  $n - j$ 
4:       if  $a[i] > a[i + 1]$  then
5:         swap( $a[i], a[i + 1]$ )
```

バブルソートの最悪計算量は  $O(n^2)$  である。

### 挿入ソート

挿入ソート (insertion sort) は、要素を順に正しい位置に挿入することで整列する。詳細は第 2 回で説明した通りである。挿入ソートの最悪計算量は  $O(n^2)$  であるが、入力がほぼ整列されているときは高速である。これを利用して、部分列に挿入ソートを用いて高速にしたシェルソート (Shell's sort) が知られている。

### 選択ソート

選択ソート (selection sort) は、最小値を先頭の要素と交換する操作を繰り返すことで整列する。アルゴリズムは次のように書ける。

```
1: SELECTIONSORT( $a[ ], n$ )
2:   for  $i = 1$  to  $n - 1$ 
3:      $min = i$ 
4:     for  $j = i + 1$  to  $n$ 
5:       if  $a[min] > a[j]$  then
6:          $min = j$ 
7:     swap( $a[i], a[min]$ )
```

選択ソートの最悪計算量は  $O(n^2)$  である。

### ヒープソート

ヒープソート (heap sort) は、選択ソートにおける最小値の探索を、ヒープというデータ構造によって高速にしたソートである。

## ヒープ

2分木  $T$  を図示したとき、深さ最大の段は左詰めになっており、それ以外の段はすべて埋まっているとき、 $T$  を（広義の）完全 2 分木 (complete binary tree) という。完全 2 分木  $T$  の各頂点が値を持つとする。 $T$  が **max** ヒープ条件 (max-heap property) 「各頂点の値はその子の値以上である」を満たすとき、 $T$  を **max** ヒープ (max-heap) といい、**min** ヒープ条件 (min-heap property) 「各頂点の値はその子の値以下である」を満たすとき、 $T$  を **min** ヒープ (min-heap) という。max ヒープと min ヒープをまとめてヒープ (heap) という。順序関係を逆にすれば同じ議論が適用できるので、以下では max ヒープのみを考え、単にヒープと呼ぶ。

ヒープの上から順に左から右へ向かって添字  $1, 2, \dots$  を付けることで、ヒープを配列で表現できる。添字  $i$  の頂点の親は添字  $\lfloor i/2 \rfloor$  の頂点であり、左の子は添字  $2i$  の頂点、右の子は添字  $2i + 1$  の頂点である。

ヒープ  $h$  には次の操作を行うことができる。

- **MAXIMUM**( $h$ ):  $h$  に含まれる値の最大値を求める。
- **INSERT**( $h, x$ ):  $x$  を  $h$  に挿入する。
- **EXTRACTMAX**( $h$ ):  $h$  に含まれる値の最大値を返し、その値を持つ頂点をヒープから削除する。

これらの操作を持つデータ構造を優先順位付きキュー (priority queue) という。ヒープにおける各操作は以下のように実現できる。各操作の最悪計算量は  $O(\log n)$  である。

■ **MAXIMUM**( $h$ ) ヒープ条件から、ヒープの根が最大値であるので、それを返せばよい。

■ **INSERT**( $h, x$ ) まず配列としての末尾に  $x$  を追加する。 $x$  を 2 分木の頂点と見なしたときの親と値を比較し、 $x$  の方が値が大きいとき交換する。この操作を繰り返して交換が起きなくなったとき、ヒープ条件を満たしている。

■ **EXTRACTMAX**( $h$ ) まず **MAXIMUM** と同様に、ヒープの根の値を返す。次にヒープの根を削除し、配列としての末尾にあたる頂点  $x$  を根に移動する。 $x$  が子を持つとき、 $x$  の子が 2 個あれば、それらを比較する。 $x$  の大きい方の子と  $x$  を比較し、 $x$  の方が小さければそれらを交換する。この操作を繰り返して交換が起きなくなったとき、ヒープ条件を満たしている。

## ヒープソート

配列  $a$  の要素をヒープにすべて挿入し、最大値を順に取り出すことで、配列  $a$  を整列することができる。これをヒープソートという。アルゴリズムは次のように書ける。

- 1: **HEAPSORT**( $a[ ], n$ )
- 2: 空のヒープ  $h$  を用意する
- 3: **for**  $i = 1$  **to**  $n$
- 4:     **INSERT**( $h, a[i]$ )
- 5: **for**  $i = n$  **downto** 1
- 6:      $a[i] = \text{EXTRACTMAX}(h)$

空のヒープ  $h$  を用意する代わりに配列の一部  $a[1], a[2], \dots, a[i]$  をヒープと見なすことで、作業領域を使わずにヒープソートを行うこともできる。ヒープソートの最悪計算量は  $O(n \log n)$  である。なお、第 3 行、第 4 行のヒープの構成には最悪計算量  $O(n)$  のアルゴリズムも知られている。