アルゴリズム B 第 6 回　　2022 年 11 月 14 日

6　ソートアルゴリズム (1)

以下では，配列 $a[1], a[2], \ldots, a[n]$ を昇順に並べ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してソートする．アルゴリズムは次のように書ける．ただし，$\mathrm{swap}(a[i], a[i+1])$ 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ソートする．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，入力がほぼソート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ソートする．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．ただし，作業領域を節約するため，以下では最小値の代わりに最大値を探索する．

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．また，max ヒープと min ヒープをまとめて**ヒープ** (heap) という．順序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 $1, 2, \ldots$ 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 h 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．Maximum の最悪計算量は $O(1)$ であり，他の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 をソートすることができる．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] = ExtractMax(h)
```

空のヒープ h を用意する代わりに配列の一部 $a[1], a[2], \ldots, a[i]$ をヒープと見なすことで，ヒープのための作業領域を使わずにヒープソートを行うことができる．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．