アルゴリズム論（第5回）

2003.11.10
櫻井　彰人
algorithm@ae.keio.ac.jp
http://www.sakurai.comp.ae.keio.ac.jp/

きょうの講義概要

- **整列アルゴリズム（2）**
  - » **クイックソートアルゴリズム3.3.1の補充**
  - » **クイックソートの計算量**
  - » **ヒープソート**
  - » **ヒープソートの演習**
  - » **ソートアルゴリズムのまとめ**
- **原理、アルゴリズム、計算量、正当性、実際問題での利用**

アルゴリズム3.3.1　pivotの位置決定

- **p=t とし、i = t+1 から始めて r まで、iを増やすごとに以下を繰り返す**
  - **i番目の要素がpivotより小さいときだけ p←p+1としてから、p にある要素とi にある要素とを入れ換えて、iを1増やす。i番目の要素がpivot以上であれば、何もしないで i を1増やす。**
- **上の操作が終わる**
  - **p が確定＝pivotが入るべき位置**
  - **pにあるものとpivotとを入れ換える。**
- **(1.5)は、この手続き自身の再帰呼び出し**

クイックソートの計算量（1）

- **最悪の場合**
  - **pivotがいつも左端（最小値）または右端（最大値）＝再帰で長さが1つ減るだけ**
  - **(n-1) + (n-2) + … + 2 + 1 = (n(n-1))/2**
  - **$\Theta(n^2)$**
- **最良の場合**
  - **pivotがいつも真ん中に＝再帰のたびに長さが半分**
  - **再帰の回数：log n、各再帰でpivotの位置を決めるのに n-1 回以下の比較**
  - **Θ(n log n)**

クイックソートの計算量（2）

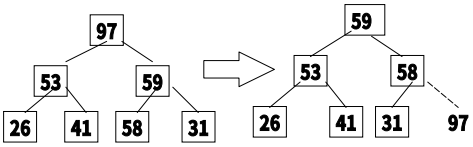
- **平均の場合**
  - **n個の要素を整列、pivotとそれ以外の n-1 個の要素を比較**
  - **pivotで分割された要素の個数を a と b、このレベルでの計算量をQ_n**
    - » **$Q_n = (n-1) + Q_a + Q_b$**
  - **a と b の組を (a, b)**
  - **分割の可能性**
    - » **(0, n-1), (1, n-2), (2, n-3), …, (n-1, 0)**
  - **分割が等確率と仮定**

クイックソートの計算量（3）

- **平均の場合（続き）**
  - **$Q_n = (n-1) + (1/n) \{ (Q_0 + Q_{n-1}) + (Q_1 + Q_{n-2}) + \ldots + (Q_{n-1} + Q_0) \}$**
  - **$Q_n = (n-1) + (2/n) \Sigma_{k=0,n-1} Q_k$　　(n>1)**
    - » **ただし、$Q_0 = 0$, $Q_1 = 0$**
  - **$Q_n = 2(n+1)(H_{n+1} - 2) + 2$**
    - » **ただし、$H_n = 1 + 1/2 + 1/3 + \ldots + 1/n$**
    - » **H_n：調和数　≒　$\log_e n$**
  - **$Q_n \fallingdotseq 2n \log_e n \fallingdotseq 1.39n \log_2 n$**
  - **Θ(n log n)**

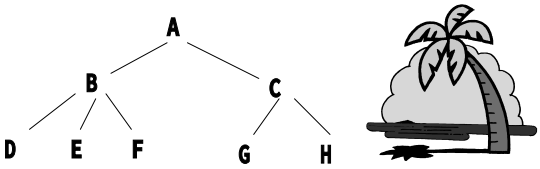
3.5 ヒープソートの考え方

■ ヒープ（heap）
- 根が最大値、親は子より小さくない木（tree）、Williams, J. W. J. (1964)
  - » ヒープができれば、根が最大値
  - » 根を除いてヒープを作り直すという操作を繰り返すとソートしたことになる



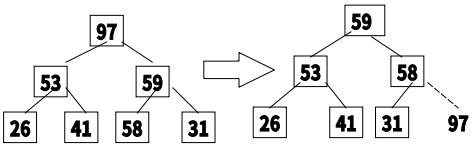
木とは

■ 木（tree）
- 階層的な構造で情報を表現



ヒープソートの概略

- 与えられたデータからヒープを作る
- 次の手順をデータが2個になるまで繰り返す
  - 根にある要素と、対象にしているデータの最後にある要素を入れ換える
  - 対象とするデータから最後にあるデータをとり、ヒープを再構築する

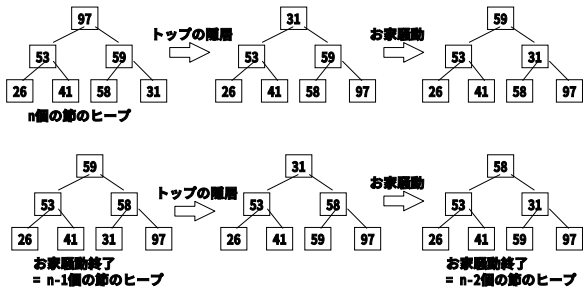


ヒープソートの概略（続 or 俗）

- 与えられたデータからヒープを作る
  - どの派閥も（派閥の中の派閥も）トップが最大
- 以下の手順をデータが2個になるまで繰り返す
  - 根にある要素と、対象にしているデータの最後にある要素を入れ換える
    - » 全体のトップは引退いただく。
    - » 離れを作ってそこを隠居所にする
  - 対象とするデータから最後にあるデータをとり、ヒープを再構築する
    - » 離れにいた人を（いる場所がないので）とりあえず、トップにすえる。お家騒動（派閥の中の派閥も含め、トップに最大がいるとは限らないので発生）を収束させ、どの派閥（派閥の中の派閥も含め）トップが最大となるようにする

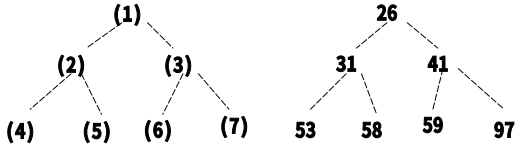
木の上でのヒープソート

■ ヒープソート中の木



ヒープでの大小関係

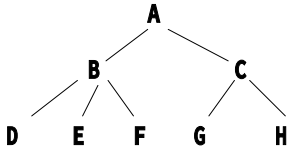
■ ヒープソート終了時の木



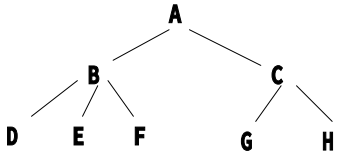
n個のデータ ⇨ n個の節でヒープ ⇨ n-1個でヒープ
⇨ …… ⇨ 2個の節でヒープ

木の定義

- 木(tree)T:節(node)の有限集合
  - (1)根(root)と呼ぶ特別な節が1つ
  - (2)根以外の節は、互いに素な集合 $T_1, T_2, \ldots, T_m$ $(m \geq 0)$ に分割でき、各 T_i も木。T_i :Tの部分木(subtree)
- 葉(leaf)
  - 部分木を持たない節
- 枝
  - 根と部分木の根との対



木の例

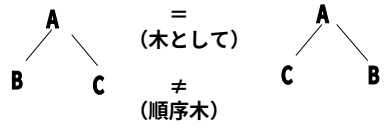


$T = \{A, B, C, D, E, F, G, H\}$ $Root(T) = A$
$T_1 = \{B, D, E, F\}$ $Root(T_1) = B$ $T_2 = \{C, G, H\}$
$Root(T_2) = C$
$T_{11} = \{D\}$ $T_{12} = \{E\}$ $T_{13} = \{F\}$ $T_{21} = \{G\}$ $T_{22} = \{H\}$
$Leaf(T) = \{D, E, F, G, H\}$

木の高さと順序木

- 節のレベル
  - 根からその節までの枝の数
- 木の高さ
  - 木に含まれる節のレベルの最大値
    - » 例にあげた木Tの高さ:2
- 順序木
  - 葉以外の節において、部分木($T_1, T_2, \ldots, T_m$)の間に順序関係($i<j$ なら $T_i<T_j$)が成り立つ木

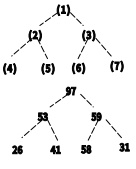
木、順序木、2分木



2分木:
枝分かれが2以下の順序木

ヒープの表現法

- 配列による表現(Floyd, R. W. (1964) など)
  - 根
    - » 配列の1番目
  - i 番目の要素の左の子
    - » 2i 番目
  - i 番目の要素の右の子
    - » 2i+1 番目



97	53	59	26	41	58	31	.....
1	2	3	4	5	6	7	...

配列aでのヒープの(再)構成

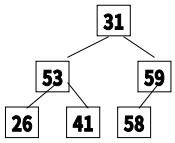
- ヒープの再構成arrangeheap(k,r)
  (あだな:派閥のお家騒動)
  - k が根、r が葉の最後
  - a[k]の子供a[2k]とa[2k+1]を根とする二つの部分木はすでにヒープ
  - a[k]を含む木全体をヒープに再構成
  - a[k] の子供のうちで大きいものと a[k] とを比べ、子供のほうが大きければ入れ替え、入れ替えたところでまた子供の大きいものと比較する
  - 子供のほうが大きくないか子供がいなくなるまで、以上の操作を繰り返す

アルゴリズム3.4 arrangeheap

- ヒープの再構成arrangeheap(k,r)
- 配列 a[k,k+1,…,r] をヒープに再構成
- 内容
  - (1)ループを抜けるまで以下を繰り返す
    - » (1.1) j ←2k
    - » (1.2) j>r なら終了
    - » (1.3) a[j+1] > a[j] のとき j←j+1
    - » (1.4) a[k] ≥ a[j] なら終了
    - » (1.5) a[k] と a[j] を入れ替える
    - » (1.6) k←j

ヒープ再構成の演習

- 次の図に対応するデータを配列で表現し、アルゴリズム3.4でヒープが再構成されることを確かめなさい。



ヒープソートアルゴリズム

- (1)与えられたn個のデータから木(実際には配列)を作り、葉に近い部分木からヒープ再構成アルゴリズムを適用して木全体をヒープに
- (2)根と葉の最後の要素を入れ換え、対象とするデータの個数を一つずつ減らしながら、対象になるデータが2個になるまでヒープ再構成アルゴリズムを繰り返して適用する

アルゴリズム3.5 ヒープソート

- 関数heap(1,n)
- 入力と出力は配列a[1,2,3,…,n]に
- 内容
  - (1)n=1なら、何もしないで終了
  - (2)i = div(n,2) から1まで、arrangeheap(i,n) を繰り返す
  - (3)i = n から2まで以下を繰り返す
    - » (3.1) 1番目の要素と i 番目の要素とを入れ替える
    - » (3.2) arrangeheap(1,i-1)
- div(n,2) :nが偶数のときn/2、奇数のとき(n-1)/2

ヒープソートの実行

はじめのデータにheap(1,n)を

26	53	58	97	41	59	31
1	2	3	4	5	6	7

(2) でarrangeheap(3,7), a-heap(2,7), a-heap(1,7)を

97	53	59	26	41	58	31

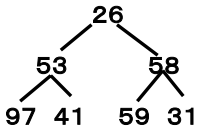
(3.1) を実行すると

31	53	59	26	41	58	97

ヒープソートの実行(2)

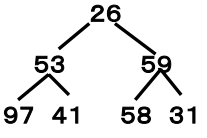
はじめのデータ

26	53	58	97	41	59	31
1	2	3	4	5	6	7



arrangeheap(3,7)

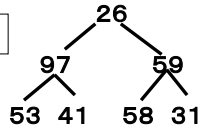
26	53	59	97	41	58	31
1	2	3 k	4	5	6	7 r



ヒープソートの実行（3）

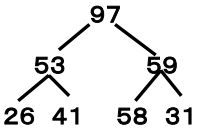
arrangeheap(2,7)

1	2	3	4	5	6	7
26	97	59	53	41	58	31
	k					r



arrangeheap(1,7)

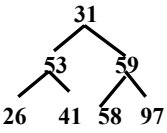
1	2	3	4	5	6	7
97	53	59	26	41	58	31
k						r



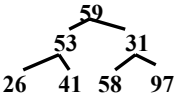
ヒープソートの実行（4）

１と７の要素を入れ換えa-heap(1,6)

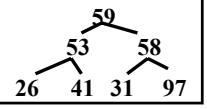
1	2	3	4	5	6	7
31	53	59	26	41	58	97
k					r	



1	2	3	4	5	6	7
59	53	31	26	41	58	97



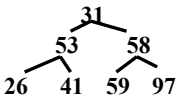
1	2	3	4	5	6	7
59	53	58	26	41	31	97



ヒープソートの実行（5）

１と６の要素を入れ換えa-heap(1,5)

1	2	3	4	5	6	7
31	53	58	26	41	59	97
k				r		



１と５を入れ替えてa-heap(1,4)，。。。。

1	2	3	4	5	6	7
26	31	41	53	58	59	97

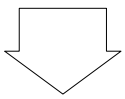
ヒープソートの演習

- 示した例にならい、クイックソートの演習で使ったものと同じ数列をヒープソートで小さい順に並べ換えなさい。ヒープの変化なども分かるように図を書いてみること。

7	3	1	4	2	6	5

ヒープソートの正当性

- ヒープの再構成アルゴリズムが正しい
- （1）で全体のヒープができる

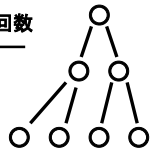


- 累積帰納法の原理でデータの個数にかかわらずヒープソートで整列可能

ヒープソートの計算量（1）

- （1）与えられたデータ全体で最初にヒープを作る ＋ （2）根のデータを最後の節と入れ替えながらヒープの再構成を繰り返す
  - 説明は、データ（節）の数$n = 2^k - 1$（完全2分木）のケースで

節	個数	ヒープ再構成の最大比較回数
高さ0（葉）	$(n+1)/2$	0回 ＝ 2x0 回
高さ1	$(n+1)/2^2$	2回 ＝ 2x1 回
高さ2	$(n+1)/2^3$	4回 ＝ 2x2 回
高さ3	$(n+1)/2^4$	6回 ＝ 2x3 回
高さ4	$(n+1)/2^5$	8回 ＝ 2x4 回



ヒープソートの計算量（2）

- （1）与えられたデータ全体で最初にヒープを作るときの比較回数（つづき）
  - 上限は 2(n+1)

$$2x0 \times ((n+1)/2^1) + 2x1 \times ((n+1)/2^2) + 2x2 \times ((n+1)/2^3) + 2x3 \times ((n+1)/2^4) + \cdots\cdots\cdots$$
$$= (n+1)\{ 0 + 1/2 + 2/2^2 + 3/2^3 + 4/2^4 + 5/2^5 + \cdots\cdots \}$$
$$= 2(n+1)$$

ヒープソートの計算量（3）

- （2）根のデータを最後の節と入れ替えながらヒープの再構成を繰り返す
  - 一度ヒープができれば、根と最後の節を入れ替えたあとで1度再構成するときの比較回数の最大値は
    - （節の高さ） × 2
    - 一番高い節では、$\{ \log_2 (n+1) - 1\} \times 2$
      - このレベルで、$((n+1)/2) - 1$ 回の繰り返し
      - $[\{ \log_2 (n+1) - 1\} \times 2] \times \{((n+1)/2) - 1\}$
    - 対象とするヒープの高さが一つ減った節では、$\{ \log_2 (n+1) - 2\} \times 2$
      - このレベルで、$(n+1)/4$ 回の繰り返し
      - $[\{ \log_2 (n+1) - 2\} \times 2] \times \{(n+1)/4\}$

ヒープソートの計算量（4）

- （2）根のデータを最後の節と入れ替えながらヒープの再構成を繰り返す（つづき）
  - 以上の操作の比較回数の上限（計算のため、一番高い節での繰り返し回数を(n+1)/4とおく）
    - » $2(n+1) \log_2 (n+1) - 4(n+1)$

$$\{\log_2 (n+1) -1\} \times 2 \times (n+1)/4 + \{\log_2 (n+1) -2\} \times 2 \times (n+1)/8 + \{\log_2 (n+1) -3\} \times 2 \times (n+1)/16 + \cdots\cdots$$
$$= (n+1) \log_2 (n+1) \{ 1/2 + 1/2^2 + 1/2^3 + 1/2^4 + \cdots\cdots \}$$
$$- (n+1) \{1/2 + 2/2^2 + 3/2^3 + 4/2^4 + \cdots\cdots \}$$

- （1） ＋ （2）
  - » $2n \log_2 (n+1)$
  - » $\Theta(n \log_2 n)$

3.6 ソートアルゴリズムのまとめ

アルゴリズム名	計算量	実装の難度
最小（大）値法	$\Theta(n^2)$	最も簡単
挿入法	$\Theta(n) \sim \Theta(n^2)$	簡単
シェル	$\Theta(n^{4/3})$程度	やや簡単
クイック	$\Theta(n \log n) \sim \Theta(n^2)$	やや難
ヒープ	$\Theta(n \log n)$	難