アルゴリズム論（第4回）

2003.10.27
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

- これからやること（アルゴリズム各論）
  - 整列、探索、グラフ
- きょう（整列アルゴリズム（1））
  - バブルソート
  - 挿入法
  - クイックソート
- 原理、アルゴリズム、計算量、正当性、実際問題での利用

3 整列アルゴリズム

- 数値や文字列を一定の順序（小さい順、アルファベット順など）で並べ換える
  - 以下の説明では特にことわらない限り小さい順
- 名簿や辞書の作成、テストの点数による並べ換え、データベースの作成と検索など
- 評価：n個のデータを並べ換えるときの比較回数
  - $\Theta(n \log n)$から$\Theta(n^2)$まで

3.0 最小（大）値法

- n個のデータの最大（小）値を見つけて一番後ろに
- 残りの（nー1）個のデータで同様に
- 以上を最後に1つのデータとなるまで繰り返す
- 比較回数
  - (n-1) + (n-2) + … + 2 + 1 = n(n-1)/2
  - $\Theta(n^2)$のアルゴリズム

最小（大）値法

- 動作例
  - 初期データ: 8 4 3 7 6 5 2 1
    - 4 3 7 6 5 2 1 8 （1回目のループ終了時）
    - 4 3 6 5 2 1 7 8 （2回目のループ終了時）
    - 4 3 5 2 1 6 7 8 （3回目のループ終了時）
    - 4 3 2 1 5 6 7 8 （4回目のループ終了時）
    - 3 2 1 4 5 6 7 8 （5回目のループ終了時）
    - 2 1 3 4 5 6 7 8 （6回目のループ終了時）
    - 1 2 3 4 5 6 7 8 （7回目のループ終了時）

3.1 バブルソート

- 1番目と2番目を比較し、順序が逆であれば入れ換える。次に2番目と3番目を比較して入れ換える。これを最後まで（n番目まで）行うと、最後の数が最大の数として確定する。
- 残りの（n-1)個のデータで同様に繰り返す
- 以上を最後に1つのデータとなるまで繰り返す
- 比較回数
  - (n-1) + (n-2) + … + 2 + 1 = n(n-1)/2
  - $\Theta(n^2)$のアルゴリズム

アルゴリズム3.0 バブルソート（bubble sort）

```
for( i=0; i<n; i++ )
    for( j=1; j<n – i ; j++)
        if( data[j-1] > data[j] )
            swap( data[j-1], data[j] ) ;
```

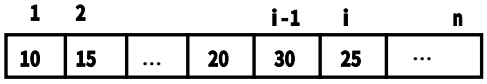
- 動作例
  - 初期データ: 8 4 3 7 6 5 2 1
    - 4 3 7 6 5 2 1 8 （1回目のループ終了時）
    - 3 4 6 5 2 1 7 8 （2回目のループ終了時）
    - 3 4 5 2 1 6 7 8 （3回目のループ終了時）
    - 3 4 2 1 5 6 7 8 （4回目のループ終了時）
    - 3 2 1 4 5 6 7 8 （5回目のループ終了時）
    - 2 1 3 4 5 6 7 8 （6回目のループ終了時）
    - 1 2 3 4 5 6 7 8 （7回目のループ終了時）

バブルソートの利用

- プログラムは簡単
- $\Theta(n^2)$のアルゴリズムは実用的でない
  - » n＝１０００で秒のオーダーでも、
  - » n＝１００００で分、n＝１０００００で時間、n＝１０００００００で日のオーダーに
- 個人的なデータの管理程度

3.2 挿入法

- すでにi-1番目までが整列済みのとき、i番目の要素を適切な位置に挿入し、そのあとのデータを一つずつずらす
- 上の手順をi = 2 からnまで行う



アルゴリズム3.1 挿入法（insertionsort）

- 入力：n個の数値または文字列の組
- 出力：一定の順序で並んだ数値または文字列の組
- 内容：i = 2 からnまで以下を繰り返す
  - (1) w ← i番目の要素；j←i-1
  - (2) （ w < j番目の要素 ）である間、jが１になるまで以下を繰り返す
    - (2.1) j＋１番目の要素 ← j番目の要素；j←j－１
  - (3) j＋１番目 ← w

アルゴリズム3.1 挿入法（insertionsort）

```
for( i=1; i<n; i++ )
    for( j=i; j>0 && data[j-1]>data[j]; j-- )
        swap( data[j-1], data[j] ) ;
```

- 動作例
  - 初期データ: 8 4 3 7 6 5 2 1
    - 4 8 3 7 6 5 2 1 （1回目のループ終了時）
    - 3 4 8 7 6 5 2 1 （2回目のループ終了時）
    - 3 4 7 8 6 5 2 1 （3回目のループ終了時）
    - 3 4 6 7 8 5 2 1 （4回目のループ終了時）
    - 3 4 5 6 7 8 2 1 （5回目のループ終了時）
    - 2 3 4 5 6 7 8 1 （6回目のループ終了時）
    - 1 2 3 4 5 6 7 8 （7回目のループ終了時）

挿入法の計算量（比較回数）

- 最悪：i番目の要素がi-1番目までのすべての要素より小さい
  - » 1 + 2 + … + (n-2) + (n-1) = n(n-1)/2
  - » $\Theta(n^2)$
- 最良：i番目の要素がi-1番目までのすべての要素より大きい
  - » n-1
  - » $\Theta(n)$
  - » データがほぼ整列されているときは効率がよい
- あまり多くない量のデータのとき有効
- 後に述べる「巧妙な」方法よりプログラミングが容易

3.3 シェルソート（Shellsort）

- D. L. Shell による（1959）
- アイデア
  - ほとんど正しく並んだデータでは挿入法が有利
  - きざみ幅 1+h で挿入法を行い、hを減らしていく（最後は1）
- 整列アルゴリズム
  - すべての入力と出力はアルゴリズム3.1（挿入法）と同じ
  - 以下のアルゴリズムの記述では入出力を省略

シェルソート（Shellsort）

- アイデアをより具体的に
  1. 適当な間隔hを決める
  2. 間隔hをあけて取り出したデータ列に挿入ソートを適用する
  3. 間隔hを狭めて、2.を適用する操作を繰り返す
  4. h=1になったら、最後に挿入ソートを適用して終了

アルゴリズム3.2 シェルソート（Shellsort）

- 内容：h<n の適当な h から始めて h＝１までのいくつかのhで以下を繰り返す
  - » i = 1+h から n まで以下を繰り返す
    - (1) w ← i番目の要素; j ← i – h
    - (2)（w < j番目の要素）である間、jが1になるまで以下を繰り返す
      - (2.1) j+h 番目の要素 ← j番目の要素; j ← j – h
    - (3) j＋h 番目 ← w

シェルソートの実行例

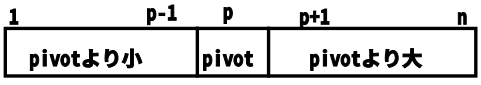
- 3 7 9 0 5 1 6 8 4 2 0 6 1 5 7 3 4 9 8 2
- 3 7 9 0 5 1 6 → 3 3 2 0 5 1 5
  8 4 2 0 6 1 5 7 4 4 0 6 1 6
  7 3 4 9 8 2 8 7 9 9 8 2
- 3 3 2 → 0 0 1
  0 5 1 1 2 2
  5 7 4 3 3 4
  4 0 6 4 5 6
  1 6 8 5 6 8
  7 9 9 7 7 9
  8 2 8 9

シェルソートの計算量と利用

- データ量が n／h のシェルソート：$\Theta((n/h)^2)$
- h回の繰り返し：$\Theta(n^2/h)$
- 正確な計算量の解析は困難
  - » $h = 2^k - 1$ なる h（1, 3, 7, 15, 31,…）を使用
  - » $\Theta(n^{3/2})$
  - » $h_{n+1}=3h_n+1$ なる h（1, 4, 13, 40, 121,…）を使用
  - » $\Theta(n^{1.25})$
  - » $h = 2^p3^q$ なる h（1, 2, 3, 4, 6, 8, 9, 12, 16,…）を使用
  - » $\Theta(n \cdot (\log n)^2)$

3.4 クイックソート（quicksort）

- C. A. R. Hoare による（1962）
- 2分探索と同じ分割統治法のアイデア
  - 直接整列させることをしない
  - ある値のデータ（pivot）より小さいものを左、大きいものを右に集め、それぞれの中で同じことを再帰的に繰り返す



アルゴリズム3.3 クイックソート

- sort(t,r)：tはデータの左端を示し、rは右端を示す。sort(1,n)で利用。
- （1）t<rならば
  - （1.1）tからrの要素からpivotを選ぶ
  - （1.2）pivot以下の要素をtからp-1に集める
  - （1.3）pivot以上の要素をp+1からrに集める
  - （1.4）p番目にpivotを入れる
  - （1.5）sort(t,p-1); sort(p+1,r)
- （2）終了して戻る

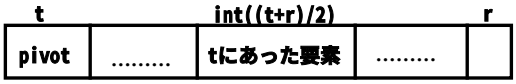
クイックソート

- **動作例（アイデア）**
  - **初期データ：** 8 4 3 7 6 5 2 1

8 4 3 7* 6 5 2 1
4 3 2* 6 5 1 7 8
1 2 4 3* 6 5 7 8
1 2 3 4 6* 5 7 8
1 2 3 4* 5 6 7 8

pivotの位置

- pivotの選択
  - たとえば int((t+r)/2) にある要素
- Pivotを入れる位置 p
  - (1.2)と(1.3)で集めてみないと決まらない
    - » 集めながらpを決定
  - まずpivot と t にあった要素を入れ替える

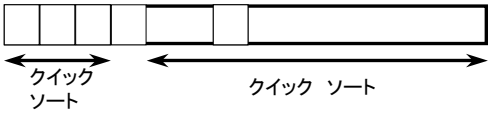


アルゴリズム3.3.1 pivotの位置決定

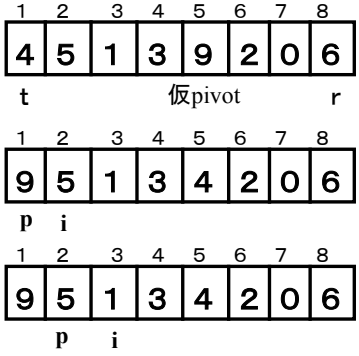
- p=t とし、i = t+1 から始めて r まで、iを増やすごとに以下を繰り返す
  - i番目の要素がpivotより小さいときだけ p←p+1としてから、p にある要素とi にある要素とを入れ換えて、iを1増やす。i番目の要素がpivot以上であれば、何もしないで i を1増やす。
  - 入替前：[p+1]～[i-1], 入替後:[p]～[i] はpivot以上
- 上の操作が終わる
  - p が確定＝pivotが入るべき位置
  - pにあるものとpivotとを入れ換える。
- (1.5)は、この手続き自身の再帰呼び出し

クイックソートの実行

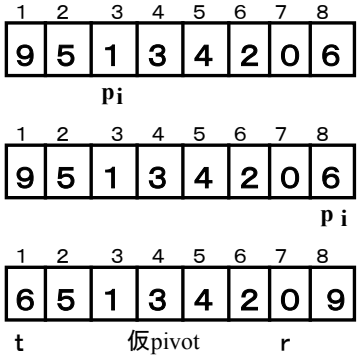
- 仮のピボットを決める
  - とりあえず真ん中にあるもの
  - 一番前にあるものと入れ換え
- ピボットとして正しい位置に
  - 順に比較していき、大きければ後ろにずらす
- ピボットの前と後で再帰的に繰り返す



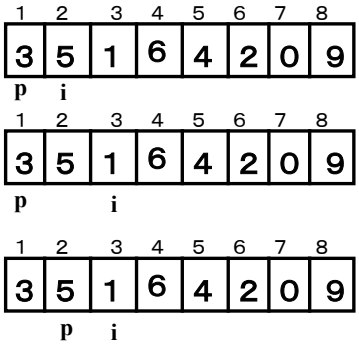
クイックソートの例



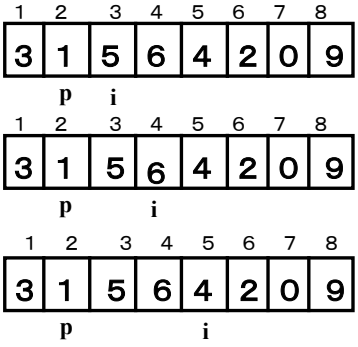
クイックソートの例（2）



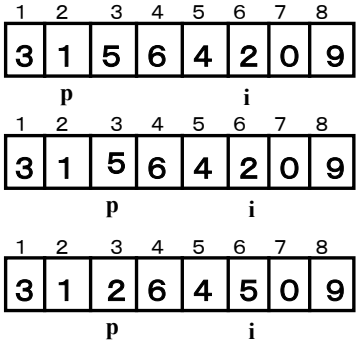
クイックソートの例（3）



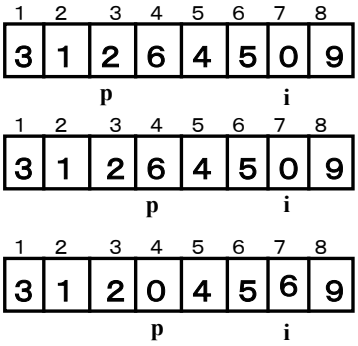
クイックソートの例（4）



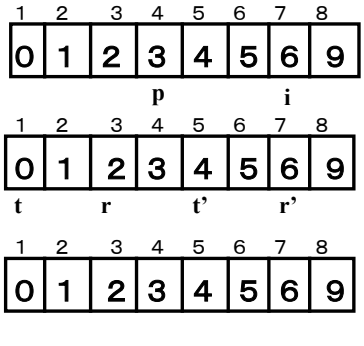
クイックソートの例（5）



クイックソートの例（6）



クイックソートの例（7）



クイックソートの演習

- ここまでに示した例にならって、次のような数列をクイックソートで小さい順に並べ換えなさい。ピボットやパラメータ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

クイックソートの正当性

- クイックソートのアルゴリズムで、もしピボットが入るべき位置に入り、それより大きなものが左になければ、累積帰納法の原理でデータの数にかかわらず正しく整列
- アルゴリズム3.3.1の部分が正しく働くことを確認すればよい
  - 最終的なpが本来ピボットの入る位置
  - ここより左にはピボットを超えるものがない

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
  - $(n-1) + (n-2) + \dots + 2 + 1 = (n(n-1))/2$
  - $\Theta(n^2)$
- 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
  - $\Theta(n \log n)$

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）
  - $Q_n = (n-1) + (1/n) \{ (Q_0 + Q_{n-1}) + (Q_1 + Q_{n-2}) + \dots + (Q_{n-1} + Q_0) \}$
  - $Q_n = (n-1) + (2/n) \Sigma_{k=0,n-1} Q_k \quad (n>1)$
    - » ただし、$Q_0 = 0$, $Q_1 = 0$
  - $Q_n = 2(n+1)(H_{n+1} - 2) + 2$
    - » ただし、$H_n = 1 + 1/2 + 1/3 + \dots + 1/n$
    - » H_n：調和数 $\fallingdotseq \log_e n$
  - $Q_n \fallingdotseq 2n \log_e n \fallingdotseq 1.39n \log_2 n$
  - $\Theta(n \log n)$