アルゴリズムとデータ構造
演習第８回
ソート１（初等的なソート）

ここでは、初等的なソート方法 （アルゴリズムＣ 第１巻 p.107） について学びます。

問題１ ［印刷用 PostScript］

(1) 次のように並んでいる数列を、バブルソート、挿入ソート、選択ソート を用いてソートしなさい。ただし、途中の過程も書くこと。

```
9 6 7 1 2
```

(2) 次のように並んでいる数列を、 シェルソートを用いてソートしなさい。ただし、途中の過程も書くこと。

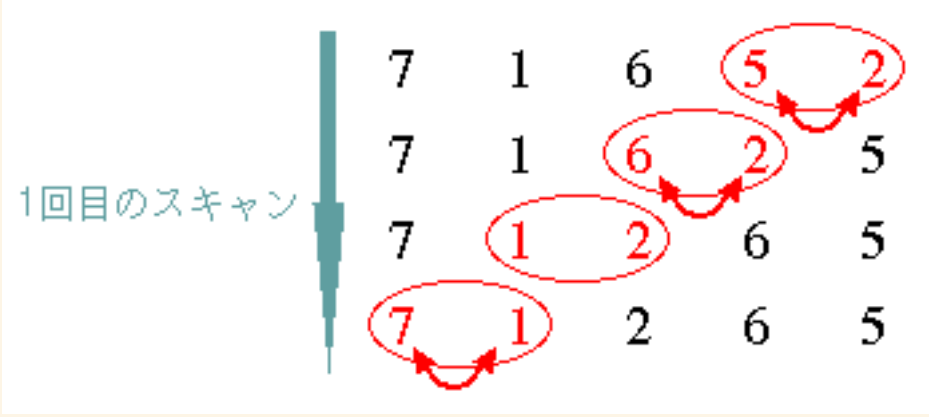
```
5 1 4 3 8 2 6 7
```

(1) の三種類のソート方法は、最も遅いものに分類されます。 どの方法も計算量は $O(n^2)$ です。 以下の図を参考にして、並び換えてください。 縦棒より左側がソート済みであることを表わしています。

バブルソート（アルゴリズムＣ 第１巻 p.116）

バブルソート	7	1	6	5	2
1回目のスキャン後	1	7	2	6	5
2回目のスキャン後	1	2	7	5	6
3回目のスキャン後	1	2	5	7	6
4回目のスキャン後	1	2	5	6	7

一回のスキャンは次のような操作になります。

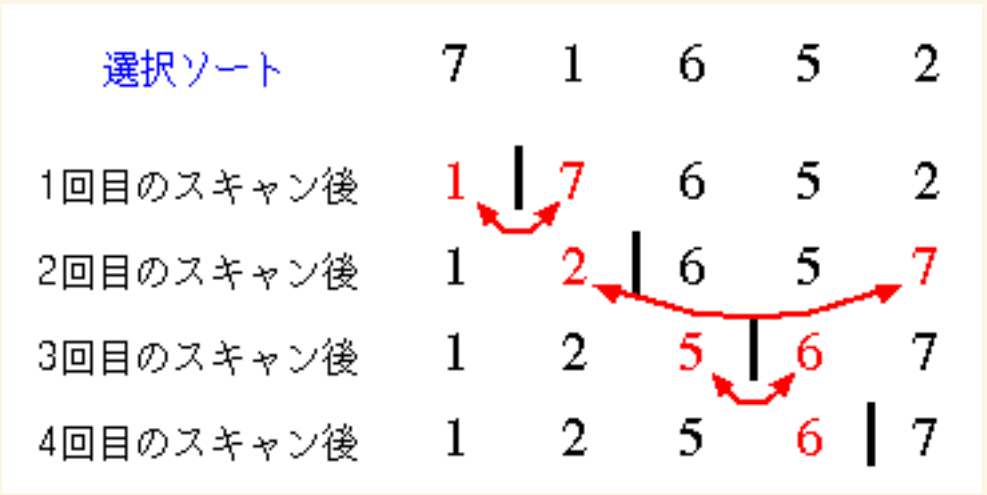


配列の後ろから前に向かって要素を二つずつ比較して、 右の方が小さければ入れ替えます。 小さい要素が泡のように浮いてくるので、バブルソートと呼ばれています。

挿入ソート（アルゴリズムＣ 第１巻 p.113）

縦棒の右隣の要素を左側の適当な場所に挿入していきます。 トランプを並べ替えるとき、大抵の人はこの方法を使っていると思います。

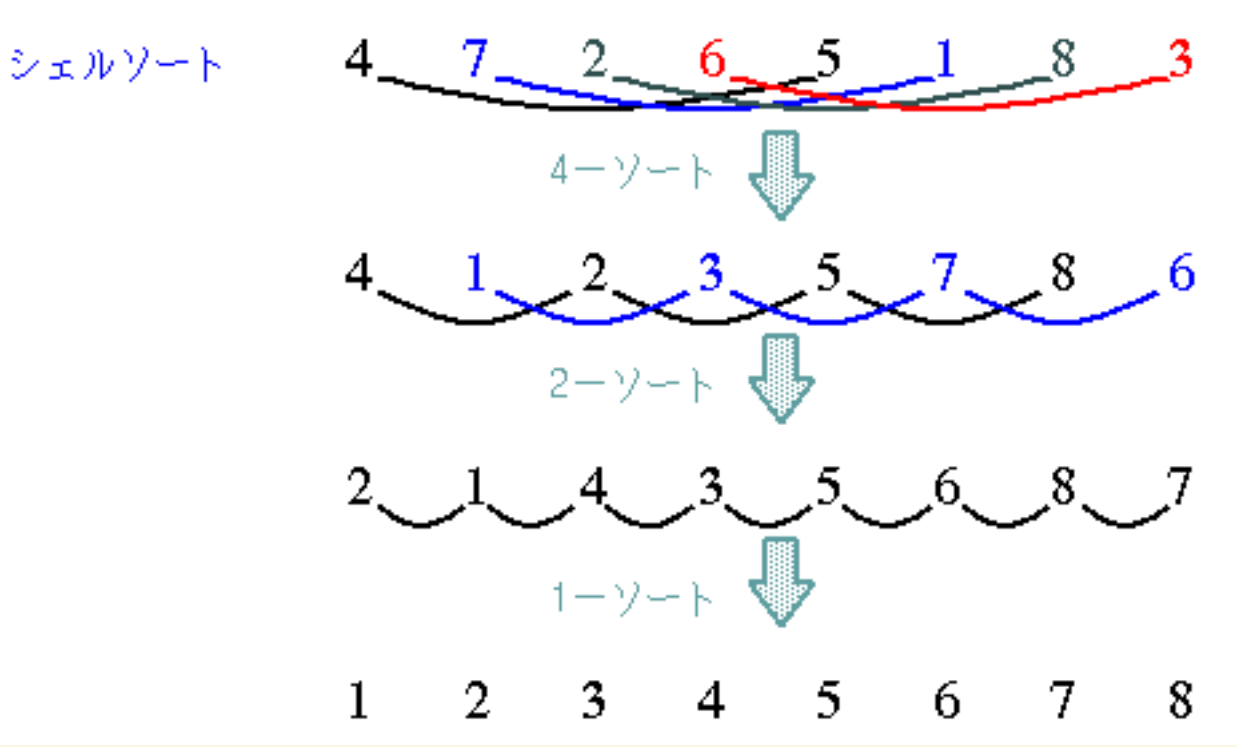
選択ソート（アルゴリズムＣ 第１巻 p.111）



縦棒より右側で一番小さいものを、縦棒の左隣と交換していきます。

(2) 挿入ソートを改良したものが、このシェルソート （アルゴリズムＣ 第１巻 p.123）です。

Algorithms and Data Structures



一定間隔で離れた要素だけで並べ替えを行なって、間隔を狭めていきます。 最後の 1-ソート が挿入ソートにあたります。 シェルソートよりも余計な手間（4-ソート と 2-ソート）がかかって、 むしろ遅くなりそうですが、かなり速いソート方法です。

問題２

バブルソート、挿入ソート、選択ソートでソートを行うプログラムを作成しなさい。プログラムは以下の条件を満たすこと。（ex08-2-skel.c）

- ソートされる整数はプログラム中の配列宣言時に初期化して良い
- 実行時に三種類の中からソート方法を選べるようにする
- ソート途中も含めてソート結果を出力する

実行例：

```
% ./a.out
Before:  7 1 6 5 2

Select a method (1:buble, 2:insertion: 3: selection) >  1
1 7 2 6 5
1 2 7 5 6
1 2 5 7 6
1 2 5 6 7
% ./a.out
Before:  7 1 6 5 2

Select a method (1:buble, 2:insertion: 3: selection) >  2
1 7 6 5 2
1 6 7 5 2
1 5 6 7 2
1 2 5 6 7
% ./a.out
Before:  7 1 6 5 2

Select a method (1:buble, 2:insertion: 3: selection) >  3
1 7 6 5 2
1 2 6 5 7
1 2 5 6 7
1 2 5 6 7
```

ソートの途中でデータにない変な数字が出てきた場合は、 ループの範囲が間違っている可能性が高いので、 そこをもう一度見直してください。

> 注意：教科書に載っている関数について
> 教科書のソート関数は、配列の（0 番目からではなく） 1 番目から データが入っていることを前提に作られており、 変数 N というのはデータの個数を表しています。 また、教科書の *bubble* 関数は前から後ろに向かって スキャンしています。よく理解した上で使ってください。

問題 3

シェルソートを行う関数を作り、 問題 2 で作った挿入ソートと速度を比較しなさい。プログラムは以下の条件を満たすこと。（ex08-3-skel.c）

- 使用するデータの個数は 100000 個とする
- データは乱数を用いて作成する
- 挿入ソートとシェルソートで同じ内容のデータを使用する
- ソートの途中や結果は表示しなくても良い
- 時間計測には、演習第 5 回で使用した gethrtime 関数を使用する

実行例（シェルソートの具体的な数値は伏せてあります）：

```
% ./a.out
insertion sort: elapsed time 316.239708.
shell sort: elapsed time ********.
```

同じ条件で比較するために、使用するデータは同じものを用意してください。 まず、配列 a に 100000 個の乱数を入れて、 そのあと配列 b に配列 a の内容をコピーし、 挿入ソートで配列 a をソートし、 シェルソートで配列 b をソートするとやりやすいでしょう。

まずはデータの個数を 10 個くらいにし、配列 a, b の内容を ソート前とソート後に表示して、うまく動いているかを確認してください。 データの個数が少ないうちは、実は挿入ソートの方が速いはずです。 興味のある人は、データの個数がいくつくらいになるとシェルソートの方が 速くなるか試してみてください。

Written by わかまつなおき