コンピュータを利用した
ソーティング手順の作成

平成 18 年 2 月 22 日

情報電子工学科 竹野研究室
森山 俊

目 次

概要

通常、人間が実際にする並べ替え作業は、人間が並べ替えの手順を求めて、人間が手作業で並べ替える。この作業に関して、コンピュータは一切使っていない。この作業に関して、コンピュータを利用して、並べ替えの手順を求めるのはコンピュータ、実際に並べ替え作業をするのは人間、というようにコンピュータを利用したソーティング手順の作成についての検討を行った。実際には、ひとつのルールを設定して、そのルールの下での最適なソーティング手順の求め方を考察し、その手順と回数、シミュレーションによる平均回数についての検討を行った。

1　はじめに

1〜100 までの数字がランダムに書かれた 100 枚の紙束があるとする。これを順番に並び替えるとして、単純なもので次の方法が考えられる。書かれた数字が 1〜50 と 51〜100 の二つに分けて、分けた二つに対して同じ手順を繰り返して行くというものである。しかしこの作業では人間がソーティングの手順を求めている。この作業にコンピュータを利用して、コンピュータにソーティングの手順を求めさせて、それに従って人間が実際にソーティングを行うというようにするのが、この研究の内容である。コンピュータに最適なソーティングの手順を求めさせて、それを音声で出力させることで、人間は、その音声に従って紙を配るだけで並び替えが終了するという方式を目標とするが、これを応用すればソーティングを行う機械を作成することも可能かもしれない。

まず、一般的なソーティング方法について考察し、次に、コンピュータを利用したソーティング手順の作成について考えることにする。

2　一般的なソーティングについて

2.1　ソーティングについて

ソーティングとは、ある集合に属する要素の有限列が与えられたとき、与えられた順序に従って要素を並べ替えることをいう。数字に限らず、文字列でも人間でも比較の方法さえ決まっていれば整列することができる。

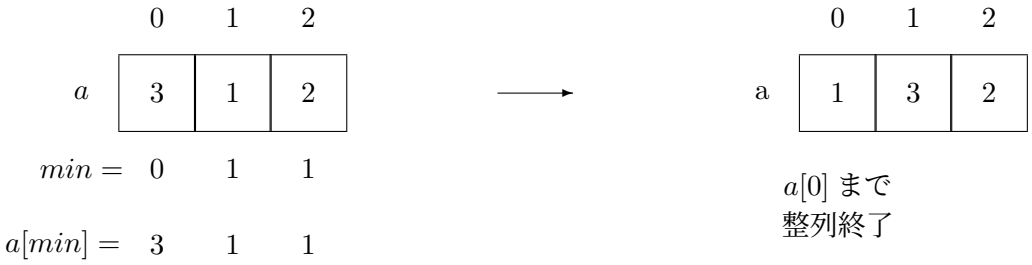
2.2　選択ソート

選択ソートにおいて、n 個の数字を昇順に並べ替えるアルゴリズムは

1. 1 番目の値から n 番目の値までを調べて最小値を見つけ、その値と 1 番目の値を入れ替える。
2. 2 番目の値から n 番目の値までの $(n-1)$ 個の中から最小値を見つけ、その値と 2 番目の値を入れ替える。
3. この処理を $(n-1)$ 回繰り返す。

というもの。

$n = 3$ の例



1. $a[0]$ から順番に見て行って、最小値の場所を min に、最小値を $a[min]$ に入れて行く。
2. 1 番最後に min に入っている値が最小値の場所で、$a[min]$ に入っている値が最小値ということになるので、$a[0]$ の値と $a[min]$ の値を入れ替える。
3. 上の図のように $a[0]$ まで整列完了となる。
4. 1. から 3. までを、$(n-1)$ 回繰り返せば終了。

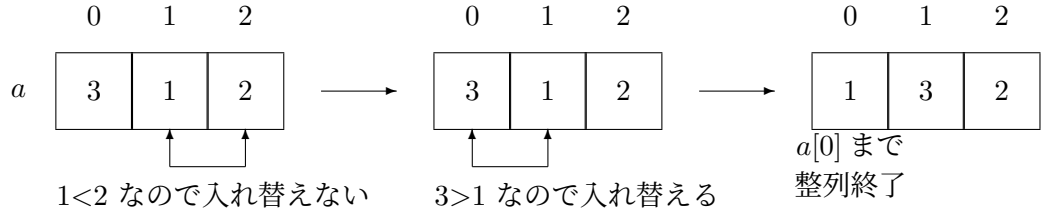
2.3 バブルソート

選択ソートは最小値を探すとき、仮の最小値と、その場所の 2 つの情報を覚えている必要がある。それらを覚えずにすむように改良したものがバブルソートである。

n 個の数字を昇順に並べ替えるアルゴリズムは

1. n 個の値の中から最小値を 1 番目にする。そのために以下のような作業をする。
   (a) n 番目と $(n-1)$ 番目を比較し、$n < (n-1)$ ならば入れ替える。
   (b) $(n-1)$ 番目と $(n-2)$ 番目を比較し、$(n-1) < (n-2)$ ならば入れ替える。
   (c) 同様に $(n-1)$ 回繰り返すと、1 番目に最小値がくる。
2. 2 番目の値から n 番目の値までの $(n-1)$ 個について、1 の作業を行う。
3. 同様の処理を $(n-1)$ 回繰り返す。

$n = 3$ の例



1. $a[1]$ の値と $a[2]$ の値を比較。$a[1]$ の値 $< a[2]$ の値なので入れ替えない。
2. $a[0]$ の値と $a[1]$ の値を比較。$a[0]$ の値 $> a[1]$ の値なので入れ替える。
3. この処理を $n - 1 = 2$ 回繰り返す。

2.4 マージソート

ソーティングされた複数のデータをまとめ、1 つのソーティングしたデータにすることをマージと呼ぶ。このマージを利用したソーティングをマージソートという。

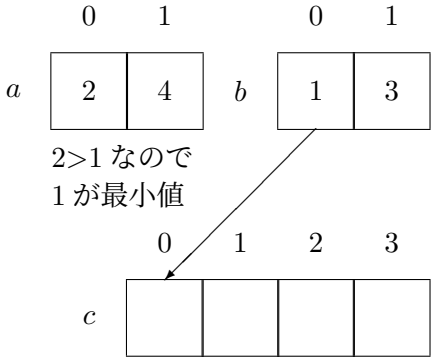
n 個の数字を昇順に並べ替えるアルゴリズムは

1. 与えられた配列データを小さい配列へと分割する。このとき各配列がほぼ均等 (要素数が 6 なら 3 と 3) になるように分割する。
2. 1 の手順を要素数が 1 になるまで繰り返す。
3. 分割された配列を順番に併合していく。このとき、2 つの配列の先頭の値の小さい方を取り出して新しい配列とする。

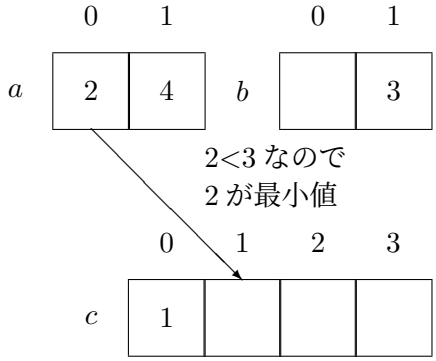
分割された配列の併合手順を具体例で説明する。

$a[0] = 2, a[1] = 4$ と $b[0] = 1, b[1] = 3$ の 2 つのソートされている配列があるとする。

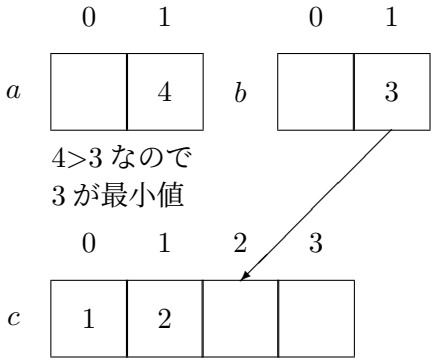
1.



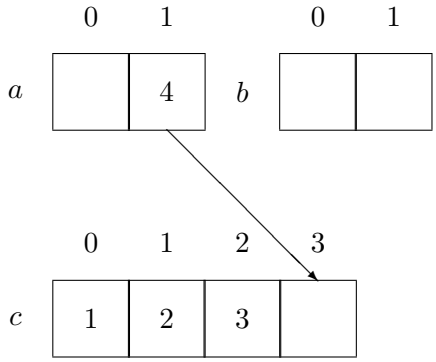
2.



3.



4.



1. 作業用の配列 $c[4]$ を用意する。

   各配列の先頭の値 ($a[0] = 2$ と $b[0] = 1$) を比較して、小さい方が最小値なので $b[0]$ の値を $c[0]$ に入れる。

2. $a[0] = 2$ と $b[1] = 3$ を比較。$a[0]$ の値を $c[1]$ に入れる。

3. $a[1] = 4$ と $b[1] = 3$ を比較。$b[1]$ の値を $c[2]$ に入れる。

4. $a[1]$ の値を $c[3]$ に入れる。終了。

2.5 挿入ソート

挿入ソートとは、データ列を整列済部分と未整列部分に分け、未整列部分の要素を整列済部分の適切な位置に入れていく方法である。

n 個の数字を昇順に並び替えるアルゴリズムは、以下のようなものである。

1. 1 番目の値と 2 番目の値を比較して、1 番目の値 $<$ 2 番目の値ならば、そのままにする。そうでないなら 1 番目の値と 2 番目の値を入れ替える。
2. 次に 3 番目の値を、整列済の 2 つの値と比較して適切な位置に入れる。
3. 2 を n 番目の値まで続ける。

整列済部分に未整列の要素を入れる方法を以下の例で説明する。

$a[0]$ の値と $a[1]$ の値までは整列済で、未整列の $a[2]$ の値を入れるとする。$a[0] = 3$,$a[1] = 5$,$a[2] = 4$

1. 未整列の $a[2]$ の値 (=4) を別の変数 tmp に入れる。
2. tmp の値 (=4) と $a[1]$ の値 (=5) を比較。

   tmp の値 $\geq a[1]$ の値ならば $a[1]$ の右に tmp の値を入れてソート完了。tmp の値 $< a[1]$ の値ならば $a[1]$ の値を右に 1 つずらす。この例では、tmp の値 (=4) $< a[1]$ の値 (=5) なので、$a[1]$ の値を右に 1 つずらす。
3. tmp の値 (=4) と $a[0]$ の値 (=3) を比較。

   tmp の値 $\geq a[0]$ の値ならば $a[0]$ の右に tmp の値を入れてソート完了。tmp の値 $< a[0]$ の値ならば $a[0]$ の値を右に 1 つずらして、$a[0]$ に tmp の値を入れてソート完了。この例では、tmp の値 (=4) $\geq a[0]$ の値 (=3) なので、$a[0]$ の右に tmp の値を入れてソート完了。

挿入ソートの具体例を以下に示す。

$a[0] = 20, a[1] = 15, a[2] = 32, a[3] = 25, a[4] = 22$ という配列があるとする。これを昇順に並び替えるには

1. $a[0]$ の値と $a[1]$ の値を比較。$a[0]$ の値 (=20)$> a[1]$ の値 (=15) なので入れ替える。
2. tmp に $a[2]$ の値 (=32) を入れる。

   tmp の値 (=32) と $a[1]$ の値 (=20) を比較。tmp の値 $> a[1]$ の値なので $a[2]$ に tmp の値を入れる。
3. tmp に $a[3]$ の値 (=25) を入れる。

   tmp の値 (=25) と $a[2]$ の値 (=32) を比較。tmp の値 $< a[2]$ の値なので $a[2]$ の値を $a[3]$ に入れる。

tmp の値 (=25) と $a[1]$ の値 (=20) を比較。tmp の値 $> a[1]$ の値なので $a[2]$ に tmp の値を入れる。

4. tmp に $a[4]$ の値 (=22) を入れる。

   tmp の値 (=22) と $a[3]$ の値 (=32) を比較。tmp の値 $< a[3]$ の値なので $a[3]$ の値を $a[4]$ に入れる。

   tmp の値 (=22) と $a[2]$ の値 (=25) を比較。tmp の値 $< a[2]$ の値なので $a[2]$ の値を $a[3]$ に入れる。

   tmp の値 (=22) と $a[1]$ の値 (=20) を比較。tmp の値 $> a[1]$ の値なので $a[2]$ に tmp の値を入れる。

5. ソート完了。

2.6 シェルソート

全体の要素を、ある数 (以下 H とする) だけ離れた要素同士を 1 つのブロックとして分割する。分割したブロックごとに挿入ソートを行う。次にブロックのサイズを大きく (H を小さく) して、各ブロック毎に再度挿入ソートを行う。これを繰り返し、最後に全体を 1 つのブロックとして挿入ソートを行う。これをシェルソートという。

ブロックへの分割の仕方を以下の例で説明する。

要素数が 8 の配列があるとする。ここで $H = 4$ としてブロックに分割する場合を考える。

4 つ離れた要素同士をひとまとめにすることなので、

- $a[0]$ と $a[4]$
- $a[1]$ と $a[5]$
- $a[2]$ と $a[6]$
- $a[3]$ と $a[7]$

のそれぞれが 1 つのブロックということになる。

シェルソートの具体例を以下に示す。

$a[0] = 4, a[1] = 2, a[2] = 6, a[3] = 5, a[4] = 1, a[5] = 3, a[6] = 8, a[7] = 7$ の配列を昇順に並び替える場合を考える。ただし H の初期値は、(要素数) / 2 とし、以降は $H/2$ とする。

1. $H = 4$

   4 つ離れた要素同士を 1 つのブロックとしてソート。

   - $a[0]$ の値 (=4) と $a[4]$ の値 (=1) を比較。入れ替え。
   - $a[1]$ の値 (=2) と $a[5]$ の値 (=3) を比較。そのまま。
   - $a[2]$ の値 (=6) と $a[6]$ の値 (=8) を比較。そのまま。
   - $a[3]$ の値 (=5) と $a[7]$ の値 (=7) を比較。そのまま。

2. $a[0] = 1, a[1] = 2, a[2] = 6, a[3] = 5, a[4] = 4, a[5] = 3, a[6] = 8, a[7] = 7, H = 2$

   2 つ離れた要素同士を 1 つのブロックとしてソート。

   - $a[0], a[2], a[4], a[6]$ をソート
     (a) $a[0]$ の値 (=1) と $a[2]$ の値 (=6) を比較。そのまま。
     (b) tmp に $a[4]$ の値 (=4) を入れる。tmp の値 (=4) と $a[2]$ の値 (=6) を比較。$a[2]$ の値を $a[4]$ に入れる。tmp の値と $a[0]$ の値 (=1) を比較。tmp の値を $a[2]$ に入れる。
     (c) tmp に $a[6]$ の値 (=8) を入れる。tmp の値 (=8) と $a[4]$ の値 (=6) を比較。$a[6]$ に tmp の値を入れる。
   - $a[1], a[3], a[5], a[7]$ をソート
     (a) $a[1]$ の値 (=2) と $a[3]$ の値 (=5) を比較。そのまま。
     (b) tmp に $a[5]$ の値 (=3) を入れる。tmp の値 (=3) と $a[3]$ の値 (=5) を比較。$a[3]$ の値を a[5] に入れる。tmp の値と $a[1]$ の値 (=2) を比較。tmp の値を $a[3]$ に入れる。
     (c) tmp に $a[7]$ の値 (=7) を入れる。tmp の値 (=7) と $a[5]$ の値 (=5) を比較。$a[7]$ に tmp の値を入れる。

3. $a[0] = 1, a[1] = 2, a[2] = 4, a[3] = 3, a[4] = 6, a[5] = 5, a[6] = 8, a[7] = 7, H = 1$

   (a) $a[0]$ の値 (=1) と $a[1]$ の値 (=2) を比較。そのまま。
   (b) tmp に $a[2]$ の値 (=4) を入れる。tmp の値 (=4) と $a[1]$ の値 (=2) を比較。$a[2]$ に tmp の値を入れる。
   (c) tmp に $a[3]$ の値 (=3) を入れる。tmp の値 (=3) と $a[2]$ の値 (=4) を比較。$a[2]$ の値を $a[3]$ に入れる。tmp の値と $a[1]$ の値 (=2) を比較。tmp の値を $a[2]$ に入れる。
   (d) tmp に $a[4]$ の値 (=6) を入れる。tmp の値 (=6) と $a[3]$ の値 (=4) を比較。$a[4]$ に tmp の値を入れる。
   (e) tmp に $a[5]$ の値 (=5) を入れる。tmp の値 (=5) と $a[4]$ の値 (=6) を比較。$a[4]$ の値を $a[5]$ に入れる。tmp の値と $a[3]$ の値 (=4) を比較。tmp の値を $a[4]$ に入れる。

(f) tmp に $a[6]$ の値 (=8) を入れる。tmp の値 (=8) と $a[5]$ の値 (=6) を比較。$a[6]$ に tmp の値を入れる。

(g) tmp に $a[7]$ の値 (=7) を入れる。tmp の値 (=7) と $a[6]$ の値 (=8) を比較。$a[6]$ の値を $a[7]$ に入れる。tmp の値と $a[5]$ の値 (=6) を比較。tmp の値を $a[6]$ に入れる。

4. ソート完了。

3　コンピュータを利用したソーティング手順の生成

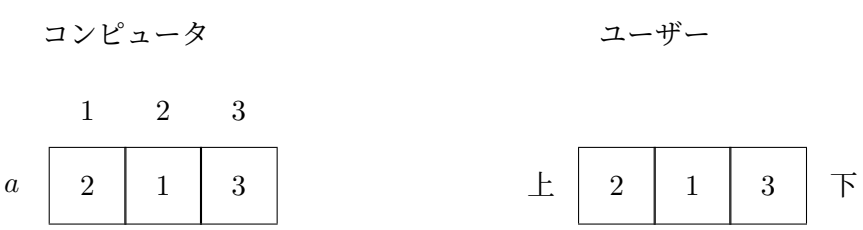
3.1　コンピュータを利用したソーティング手順の生成について

前章までは実際にあるソーティング方法について説明してきたが、この章からはコンピュータを利用したソーティングについて考えて行くことにする。

コンピュータを利用したソーティングの前提条件は以下のようにする。

- ユーザーの手元に、ランダムに並んだ n 枚の紙束がある。紙にはそれぞれ数字が書かれている。紙の並び順と数字は、データとしてコンピュータに入っている。

例



- コンピュータの作業は、データの判断とユーザーへの指示など。
- ユーザーの作業は、紙を配る、配られた紙の山を整えるなど (但し、ユーザーはコンピュータの指示通りにしか行動できない)。
- 計算量は、ユーザーの一つの作業を 1 とし、コンピュータの作業は基本的には 0 とする。

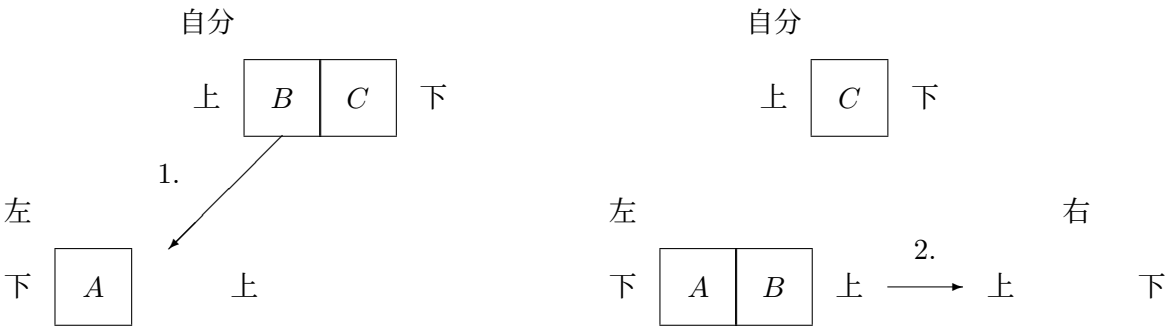
コンピュータを利用したソーティングをするには、まずコンピュータに最適な並び替えの手順を求めさせなければならない。次章からは、並び替えのルールを設定し、そのルールにおける最適な並び替え手順について考えて行く。

3.2　並べ替えに関するルール

並び替えのルールを以下のようにする。

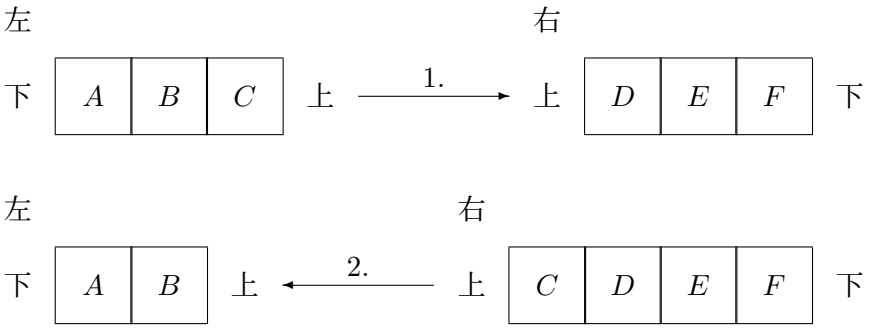
- 最初は、紙束は自分の山にある。
- 紙は 1 枚ずつ配る。
- 山は自分の山を含めて 3 つ。但し、自分の山には紙は配れない。
- 整列が終了する場所は、どこの山でもよい。
- 自分の山からは、1 番上の紙を左右どちらかの山に配ることができる。但し、自分の山に紙がある間だけとする。
- 左の山からは、1 番上の紙を右の山にだけ配ることができる。但し、左の山に紙がある間だけとする。また、自分の山から左の山に紙を配った後と、右の山から左の山に紙を配った後もできないこととする。

  理由は以下に示す。



1. 自分の山の 1 番上の紙を左に配る。
2. 左の山の 1 番上の紙を右の山に配る。

結果として、1 回目に自分の山の 1 番上の紙を右に配った場合と同じになるので、1 回無駄な手順を踏んだことになる。



1. 左の山の 1 番上の紙を右の山に配る。
2. 右の山の 1 番上の紙を左の山に配る。

結果として、元の状態に戻るので意味がないことになる。

- 右の山からは、1 番上の紙を左の山にだけ配ることができる。但し、右の山に紙がある間だけとする。また、自分の山から右の山に紙を配った後と、左の山から右の山に紙を配った後もできないこととする。理由は、左の山の 1 番上の紙を右の山に配る場合と同様である。
- 計算量は、いずれの動作も 1 とする。

以下の章では、実際に最適な並び替え手順を求める方法について考えて行くことにする。

3.3 ソーティング方法 1

設定したルールにおける昇順に並び替える最適な手順を、

1. その場合毎に可能な手順を全て調べる。取り得る全ての並び替えの手順を見つける。
2. 次に、並び替えが終了するパターンから逆に順番を辿って行き、取り得る全ての並び替えの手順を見つける。
3. 見つかった並び替えの手順を、1 番回数が少なくて済むものから順に、ランダムに並んだ紙束に対して実際に試して、並び替えが可能だったものが最適な並び替え手順になる。

という方法で求めた。

以下の例は、紙の枚数を 3 枚とした場合のものである。

手順の後ろに書かれている記号 (例：l) は、その手順を記号に置き換えるという意味。

1. 1 回目に可能な手順は、自分の山の紙を、左右どちらかの山に配る。しかし、ここでは左右どちらの山に紙を配っても同じなので、とりあえず、
   自分の山の 1 番上の紙を左の山に配る。...l
   ことになる。
2. 2 回目に可能な手順は、
   自分の山の 1 番上の紙を左の山に配る。...l
   自分の山の 1 番上の紙を右の山に配る。...r
   左の山の 1 番上の紙を右の山に配る。...L (l の後なので×)
   右の山の 1 番上の紙を左の山に配る。...R (右の山に紙がないので×)
   なので、l か r のどちらかということになる。
3. 3 回目に可能な手順は、
   - l、l の場合
     自分の山の 1 番上の紙を左の山に配る。...l (昇順になっていれば終了)
     自分の山の 1 番上の紙を右の山に配る。...r
     左の山の 1 番上の紙を右の山に配る。...L (l の後なので×)
     右の山の 1 番上の紙を左の山に配る。...R (右の山に紙がないので×)
     なので、l か r のどちらかということになる。
   - l、r の場合
     自分の山の 1 番上の紙を左の山に配る。...l
     自分の山の 1 番上の紙を右の山に配る。...r
     左の山の 1 番上の紙を右の山に配る。...L
     右の山の 1 番上の紙を左の山に配る。...R (r の後なので×)
     なので、l、r、L のどれかということになる。
4. 4 回目に可能な手順は、
   - l、l、r の場合
     自分の山の 1 番上の紙を左の山に配る。...l (自分の山に紙がないので×)
     自分の山の 1 番上の紙を右の山に配る。...r (自分の山に紙がないので×)
     左の山の 1 番上の紙を右の山に配る。...L
     右の山の 1 番上の紙を左の山に配る。...R (r の後なので×)
     なので、L ということになる。

- l、r、l の場合
  自分の山の 1 番上の紙を左の山に配る。...l (自分の山に紙がないので×)
  自分の山の 1 番上の紙を右の山に配る。...r (自分の山に紙がないので×)
  左の山の 1 番上の紙を右の山に配る。...L (l の後なので×)
  右の山の 1 番上の紙を左の山に配る。...R (昇順になっていれば終了)
  なので、R ということになる。
- l、r、r の場合
  自分の山の 1 番上の紙を左の山に配る。...l (自分の山に紙がないので×)
  自分の山の 1 番上の紙を右の山に配る。...r (自分の山に紙がないので×)
  左の山の 1 番上の紙を右の山に配る。...L (昇順になっていれば終了)
  右の山の 1 番上の紙を左の山に配る。...R (r の後なので×)
  なので、L ということになる。
- l、r、L の場合
  自分の山の 1 番上の紙を左の山に配る。...l
  自分の山の 1 番上の紙を右の山に配る。...r (昇順になっていれば終了)
  左の山の 1 番上の紙を右の山に配る。...L (左の山に紙がないので×)
  右の山の 1 番上の紙を左の山に配る。...R (L の後なので×)
  なので、l か r のどちらかということになる。

5. 5 回目に可能な手順は、

- l、l、r、L の場合
  自分の山の 1 番上の紙を左の山に配る。...l (自分の山に紙がないので×)
  自分の山の 1 番上の紙を右の山に配る。...r (自分の山に紙がないので×)
  左の山の 1 番上の紙を右の山に配る。...L (最初の並びに戻るので×)
  右の山の 1 番上の紙を左の山に配る。...R (L の後なので×)
  なので、可能な手順は無し。
- l、r、L、l の場合
  自分の山の 1 番上の紙を左の山に配る。...l (自分の山に紙がないので×)
  自分の山の 1 番上の紙を右の山に配る。...r (自分の山に紙がないので×)
  左の山の 1 番上の紙を右の山に配る。...L (l の後なので×)
  右の山の 1 番上の紙を左の山に配る。...R
  なので、R ということになる。

6. 6 回目に可能な手順は、

- l、r、L、l、R の場合
  自分の山の 1 番上の紙を左の山に配る。...l (自分の山に紙がないので×)
  自分の山の 1 番上の紙を右の山に配る。...r (自分の山に紙がないので×)
  左の山の 1 番上の紙を右の山に配る。...L (R の後なので×)
  右の山の 1 番上の紙を左の山に配る。...R (昇順になっていれば終了)
  なので、R ということになる。

終了になっている所から逆に順番に辿って行くと、並び替えの手順のパターンは全部で、以下の 5 つということになる。

最初の並び替える前の並び方を A, B, C とすると、

l,l,l...C, B, A の並び方で終了。

l,r,l,R...B, C, A の並び方で終了。

l,r,r,L...A, C, B の並び方で終了。

l,r,L,r...C, A, B の並び方で終了。

l,r,L,l,R,R...B, A, C の並び方で終了。

もしルール 1 に従って並び替えが可能とするならば、上の並び替えのパターンのどれかで並べ替えることができるはずである。つまり、回数が少なくて済むパターンから試して行き、並び替え可能だったパターンが最適な並び替え手順ということになる。

以下の表は、上の例の方法で求めた昇順に並び替える最適な手順と回数をまとめたものである。表の見方は以下の通りである。

- 並び順の数字：具体的な値ではなく、何番目に小さい値かを示す。
- 必要なし：最初から昇順に並んでいるので、並び替えの必要がない。
- 左：自分の山から左の山に紙を配る。
- 右：自分の山から右の山に紙を配る。
- 左から右：左の山から右の山に紙を配る。
- 右から左：右の山から左の山に紙を配る。

紙の枚数 $n = 2$

並び順	並び替え手順の例	回数
1,2	必要なし。	0
2,1	左。左。	2

紙の枚数 $n = 3$

並び順	並び替え手順の例	回数
1,2,3	必要なし。	0
1,3,2	左。右。右。左から右。	4
2,1,3	左。右。左から右。左。右から左。右から左。	6
2,3,1	左。右。左から右。右。	4
3,1,2	左。右。左。右から左。	4
3,2,1	左。左。左。	3

紙の枚数 $n = 4$

並び順	並び替え手順の例	回数
1,2,3,4	必要なし。	0
1,2,4,3	左。左。右。右。左から右。左から右。	6
1,3,2,4	左。右。左。右から左。右。左から右。左から右。左から右。	8
1,3,4,2	左。右。右。右。左から右	5
1,4,2,3	左。右。左。右。左から右。左から右。	6
1,4,3,2	左。右。右。右。左から右。	5
2,1,3,4	左。右。左から右。右。左。右から左。右から左。右から左。	8
2,1,4,3	左。右。左から右。左。左。右から左。右から左。	7
2,3,1,4	左。右。左から右。左。右から左。右から左。右。左から右× 3。	10
2,3,4,1	左。左。右。左から右。左から右。右。	6
2,4,1,3	左。右。左から右。左。右から左。右。左から右。左から右。	8
2,4,3,1	左。右。右。左から右。右。	5
3,1,2,4	左。右。右。左から右。左。右から左。右から左。右から左。	8
3,1,4,2	左。右。左から右。左。右から左。左。右から左。	7
3,2,1,4	左。左。右。左から右。左から右。左。右から左× 3。	9
3,2,4,1	左。右。左から右。左。右から左。右から左。左。	7
3,4,1,2	左。右。左から右。左。右。左から右。	6
3,4,2,1	左。右。左から右。右。右。	5
4,1,2,3	左。右。右。左。右から左。右から左。	6
4,1,3,2	左。右。左。左。右から左。	5
4,2,1,3	左。左。右。左から右。左。右から左。右から左。	7
4,2,3,1	左。右。左。右から左。左。	5
4,3,1,2	左。左。右。左。右から左。	5
4,3,2,1	左。左。左。左。	4

この方法の問題点は、以下のようなことが考えられる。

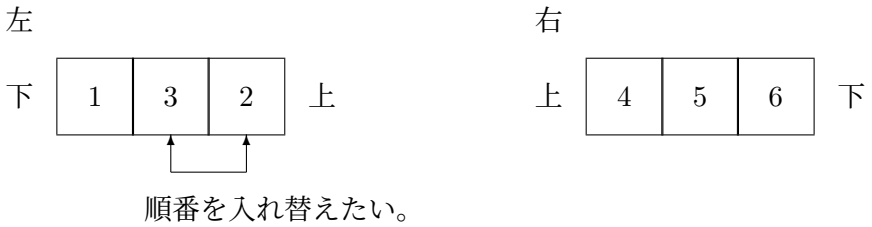
上の手順の考え方は、最悪の場合全ての並び替えの手順のパターンを試さないといけなくなる。紙の枚数を n とすると、並び替えの手順のパターンは全部で $(n! - 1)$ 個出て来る。それを全て試すのは紙の枚数が少ないときには可能でも、多いときには不可能なのが問題である。

3.4　ソーティング方法 2

ソーティング方法 1 の問題点から、確実に並び替え手順が求められる方法はないか考えてみた。そのために、まずルール 1 について考察してみた。

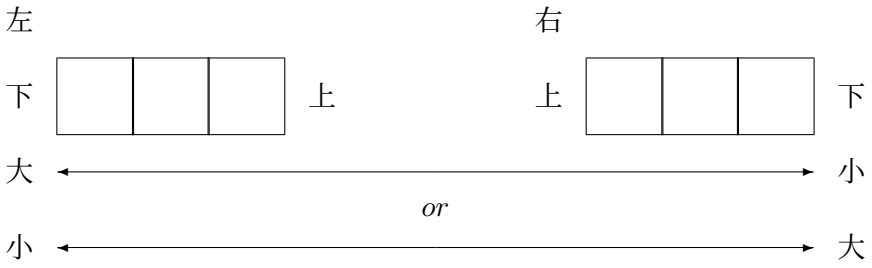
左の山からは、1 番上の紙を右の山にしか配れない。逆に、右の山からは、1 番上の紙を左の山にしか配れないので、左右の山に配られた紙は 2 度と順番を入れ替えることはできない。以下の例を参照。

例



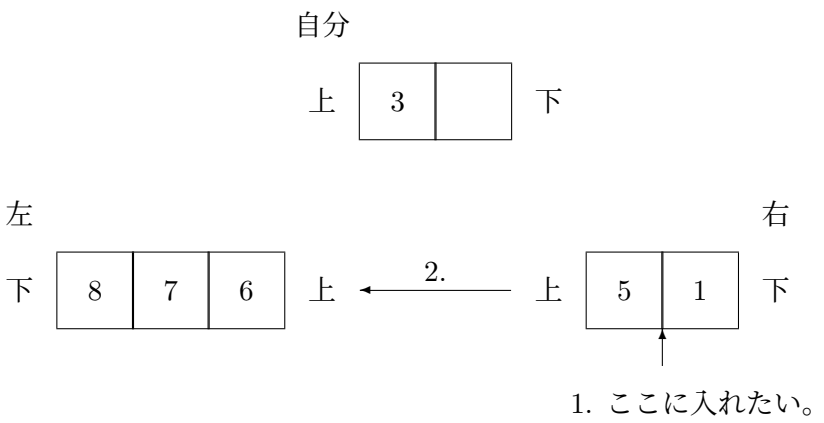
左の山の 1 番上の紙を右の山に配った後に、右の山の 1 番上の紙を左の山に配ると元の状態に戻ってしまう。逆の手順も同様。なので、左の山の 1 番上の紙を右の山に配り続ける、または、その逆の手順しか選択肢が無いことになる。しかし、それでは紙の順番を入れ替えられないのは明らか。

以上のことから、左右の山に配られた紙は 2 度と順番を入れ替えることができないため、自分の山から紙が配られた時点で、以下の図のようになっていないといけないことになる。



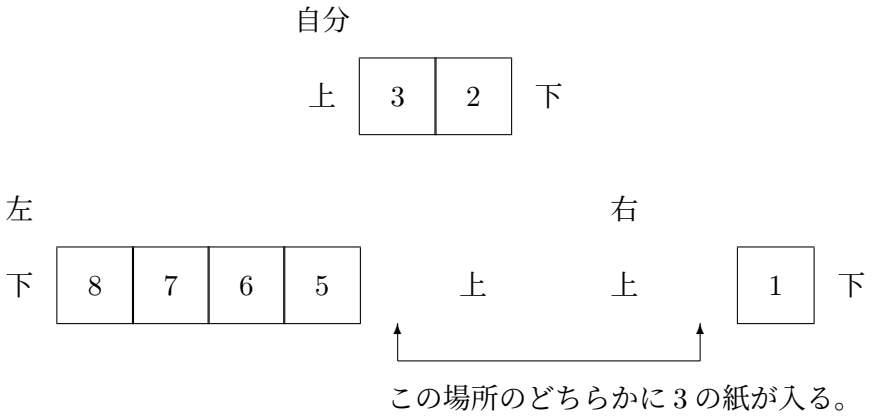
自分の山の 1 番上の紙が配られたときに、上の図の状態になるようなら、自分の山の 1 番上の紙を左右どちらかの山に配り、そうでないなら、自分の山の 1 番上の紙が入るべき場所を見つけ、その場所を空けるために左右の山の間で紙を移動させる。以下の例を参照。

例



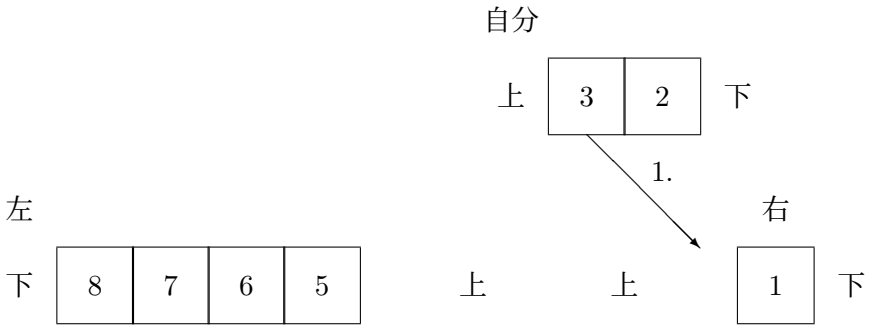
1. 条件を満たすためには、5 と 1 の紙の間に 3 の紙を入れなければならない。しかし、このままでは入れられない。
2. 3 の紙が入る場所を空けるために、5 の紙を右の山に配る。

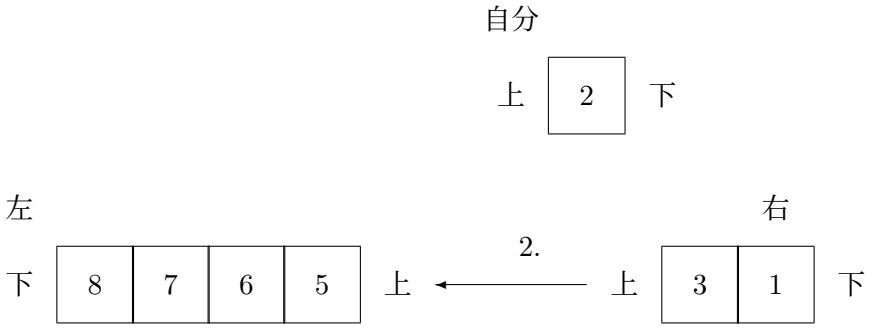
そうすると、以下の図のようになる。



上の図のどちらに 3 の紙が入るかは、自分の山の 3 の紙の後ろの紙の数字で決まる。以下の図を参照。

3 の紙の後ろの紙の数字が 3 より小さい場合





1. 3 の紙を右の山に配る。

2. 2 の紙を 3 と 1 の紙の間に入れるために、3 の紙を左の山に配る。

上の図のように、3 の紙を右の山に配っても、最終的には左の山に配ることになるので、この場合には 3 の紙を左の山に配った方がいいことになる。

このように、自分の山の 1 番上の紙の数字より自分の山の 2 番目の紙の数字の方が小さい場合は、自分の山の 1 番上の紙を、自分の山の 1 番上の紙の数字より大きい数字の紙が置かれている山の方に配った方がいいことになる。自分の山の 1 番上の紙の数字より自分の山の 2 番目の紙の数字の方が大きい場合は、上の図と全く逆に考えればいいので、自分の山の 1 番上の紙を、自分の山の 1 番上の紙の数字より小さい数字の紙が置かれている山の方に配った方がいいことになる。

まとめると、

1. 自分の山の 1 番上の紙を配った時点で、左の山の 1 番下の紙から 1 番上の紙、右の山の 1 番上の紙から 1 番下の紙 (または、その逆) までを順番に見て行ったとき、紙の数字が順番に並んでいるようにしなければいけないため、自分の山の 1 番上の紙の入る場所を見つけ、その場所を空ける。

2. 自分の山の 1 番上の紙を左右どちらの山に配るかは、自分の山の 2 番目の紙の数字を考慮して決める。

この 2 つが、ルール 1 の並び替え手順を考える上で、重要なことである。

3.5 アルゴリズム

前節のことを考慮すると、以下のようなアルゴリズムに従って手順を求めていけば、最適な並び替え手順が求まることがわかった。

1. 自分の山の 1 番上の紙を右の山に配る。

2.
   - 左右の山のどちらかに、紙が 1 枚もないとき
     (a) 自分の山の 1 番上の紙、自分の山の 2 番目の紙、紙がある方の山の 1 番上の紙の 3 枚の紙の数字を比較。
     (b) 自分の山の 1 番上の紙の数字が最小値か最大値なら、自分の山の 1 番上の紙を、紙が 1 枚もない方の山に配る。
     そうでないなら、自分の山の 1 番上の紙を、紙がある方の山に配る。

- 左右の山に紙が 1 枚以上あるとき
  (a) 自分の山の 1 番上の紙、左右の山の 1 番上の紙の 3 枚の紙の数字を比較。
  (b) 左の山の 1 番上の紙の数字が最小値でも最大値でもなければ、左の山の 1 番上の紙を右の山に配る。
  そうでないなら次へ。
  (c) 右の山の 1 番上の紙の数字が最小値でも最大値でもなければ、右の山の 1 番上の紙を左の山に配る。
  そうでないなら次へ。
  (d) – 左の山の 1 番上の紙の数字が最小値のとき
    i. 自分の山の 1 番上の紙、自分の山の 2 番目の紙の 2 枚の紙の数字を比較。
    ii. 自分の山の 1 番上の紙の数字の方が小さければ、自分の山の 1 番上の紙を左の山に配る。
    そうでないなら、自分の山の 1 番上の紙を右の山に配る。
  – 左の山の 1 番上の紙の数字が最大値のとき
    i. 自分の山の 1 番上の紙、自分の山の 2 番目の紙の 2 枚の紙の数字を比較。
    ii. 自分の山の 1 番上の紙の数字の方が小さければ、自分の山の 1 番上の紙を右の山に配る。
    そうでないなら、自分の山の 1 番上の紙を左の山に配る。

3. 2. の作業を自分の山の紙が 1 枚になるまで繰り返す。

4. - 左右の山のどちらかに、紙が 1 枚もないとき
  (a) 自分の山の 1 番上の紙、紙がある方の山の 1 番上の紙の 2 枚の紙の数字を比較。
  (b) 自分の山の 1 番上の紙の数字の方が小さければ、自分の山の 1 番上の紙を、紙がある方の山に配って終了。
  そうでないなら、自分の山の 1 番上の紙を、紙が 1 枚もない方の山に配る。
- 左右の山に紙が 1 枚以上あるとき
  (a) 自分の山の 1 番上の紙、左右の山の 1 番上の紙の 3 枚の紙の数字を比較。
  (b) – 自分の山の 1 番上の紙の数字が最大値のとき
    i. 左右の山の 1 番上の紙の 2 枚の紙の数字を比較。
    ii. 左の山の 1 番上の紙の数字の方が小さければ、右の山の 1 番上の紙を左の山に配る。
    そうでないなら、左の山の 1 番上の紙を右の山に配る。
  – 自分の山の 1 番上の紙の数字が最小値のとき
    i. 左右の山の 1 番上の紙の 2 枚の紙の数字を比較。

ii. 左の山の 1 番上の紙の数字の方が小さければ、左の山の 1 番上の紙を右の山に配る。
そうでないなら、右の山の 1 番上の紙を左の山に配る。

– 上のどちらでもないとき

i. 左右の山の 1 番上の紙の 2 枚の紙の数字を比較。

ii. 左の山の 1 番上の紙の数字の方が小さければ、自分の山の 1 番上の紙を右の山に配る。
そうでないなら、自分の山の 1 番上の紙を左の山に配る。

5. 4. の作業を自分の山の紙が 0 枚になるまで繰り返す。

6. (a) 左右の山の 1 番上の紙の 2 枚の紙の数字を比較。

(b) 左の山の 1 番上の紙の数字の方が小さければ、左の山の 1 番上の紙を右の山に配る (左の山の紙が 1 枚もなくなるまで)。
そうでないなら、右の山の 1 番上の紙を左の山に配る (右の山の紙が 1 枚もなくなるまで)。

7. 終了。

また、このアルゴリズムに従って、紙の枚数が 3 枚と 4 枚の場合の全てのパターンのランダムに並んだ紙束に対して並び替え作業をやったところ、全てのパターンについて並び替え手順と回数が一致 (但し、最初に配るのが左か右かの違いがあったので、全ての手順が左右逆になっていた) ので、このアルゴリズムで並び替えることができると言える。

3.6 並び替え手順の回数についての考察

最悪な場合の回数と平均回数について考えてみた。

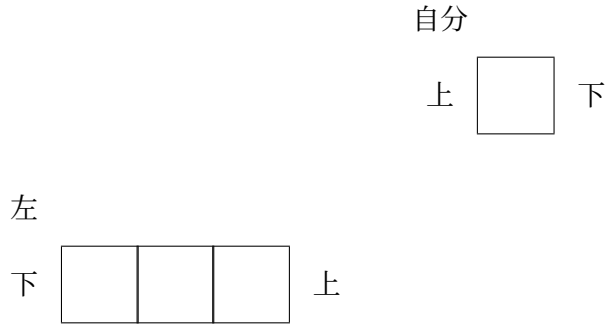
まずは、最悪な場合の回数について考えてみる。

最悪な場合の並び替え手順を、紙の枚数 n が 4 枚のときの例で説明する。

並び順	並び替え手順の例	回数
2,1,3	l,r,L,l,R,R	6
2,3,1,4	l,r,L,l,R,R,r,L,L,L	10

まず 3 枚目までの最悪な場合の並び替え手順は、$n = 3$ のときの最悪な場合の並び替え手順と同じになると考えられる。つまり回数は 6 回と考えられる。実際に、$n = 3$ のとき

と $n = 4$ のときの最悪な場合の並び替え手順を比較してみると、3 枚目までの並び替え手順は上の表のように一致している。なので 6 回目終了時点で、以下の様に 4 枚目の紙だけが手元にある状態になる。



7 回目に可能な手順は、

自分の山の 1 番上の紙を左の山に配る。...l

自分の山の 1 番上の紙を右の山に配る。...r

左の山の 1 番上の紙を右の山に配る。...L (6 回目の手順が R なので×)

右の山の 1 番上の紙を左の山に配る。...R (右の山に紙がないので×)

l か r ということになる。

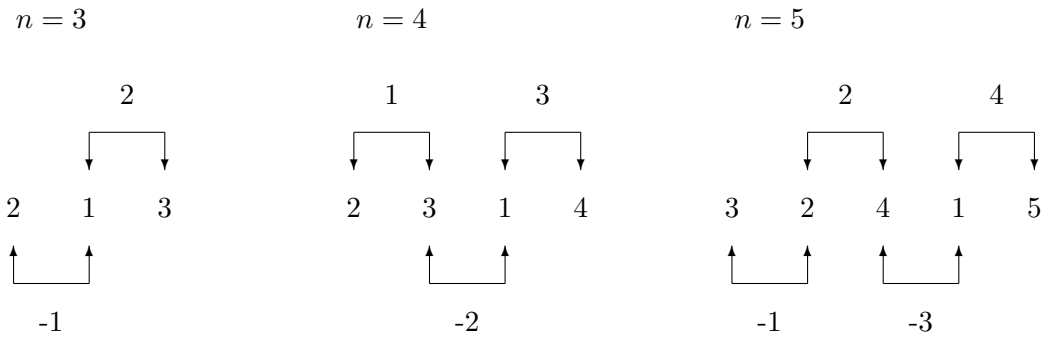
l の場合は、その時点で終了となる。なぜなら、そうでないと次に可能な手順が L(l の後にはできない) しか無いため矛盾してしまう。よって、回数は (6+1) 回ということになる。

r の場合は、左の山から順番に右の山に紙を配るしかないので、回数は (6+4) 回となる。

よって、紙の枚数 $n = 4$ のときの並び替え手順は、最悪でも (6+4) 回となる。これは、$(n - 1)$ のときの最悪な場合の手順に n を足したものになっている。

以上のことから考えると、最悪な場合の手順は $n = 5$ のときは (10+5) 回 (全ての並び方について調べたが、実際に回数は 1 番多くて 15 回だった)、$n = 6$ のときは 15+6 回、$n = 7$ のときは (21+7) 回、..... と考えられる。このことから、紙の枚数が $n(n \geq 3)$ の場合、最悪でも $(n/2) * (n + 1)$ 回で並び替えができると言える。

ちなみに、1 番回数が多い並び方は、以下の図のような並びだった。



つまり、後ろから、

最大値、最小値、2 番目に大きい値、2 番目に小さい値、.....

というような並び方が 1 番回数が多くなると考えられる。

次に、平均回数について考えてみる。

紙の枚数を n とした場合の正確な平均回数を求めようとすると、$n!$ 通りある全ての並び方について、手順の回数を求めなければいけなくなる。しかし、それは不可能なので平均回数を以下の手順で求めてみた。

1. プログラム 1 で乱数を使って 1〜A までの順列を B 行生成する。
2. プログラム 2 で、プログラム 1 で生成されたファイルから行ごとに読み込み、並び替えの回数を求めさせ、その回数を合計に加算させる。
3. 2. を B 回繰り返させる。
4. 最終的な合計を B で割って平均を求めさせ、B、合計、平均、改行を指定したファイルに追加で出力させる。
5. 4. までを csh で $B = 100$〜10000 まで 100 刻みで繰り返させる。

その結果は、以下のグラフのようになった。

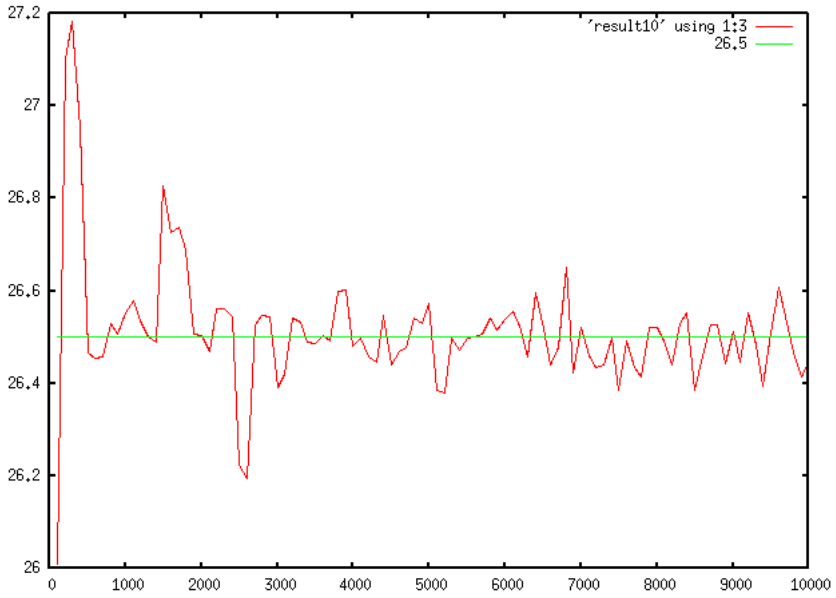


Fig. 1　$n = 10$

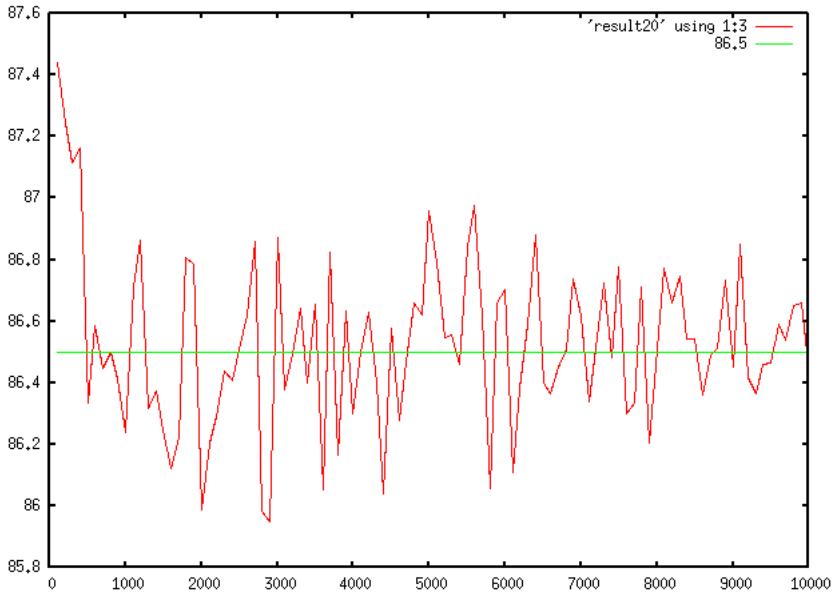


Fig. 2　$n = 20$

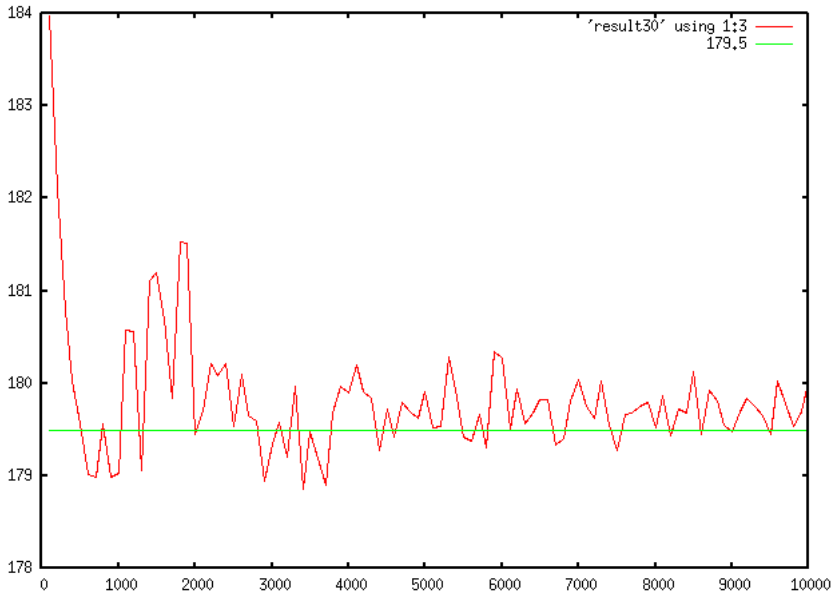


Fig. 3 $n = 30$

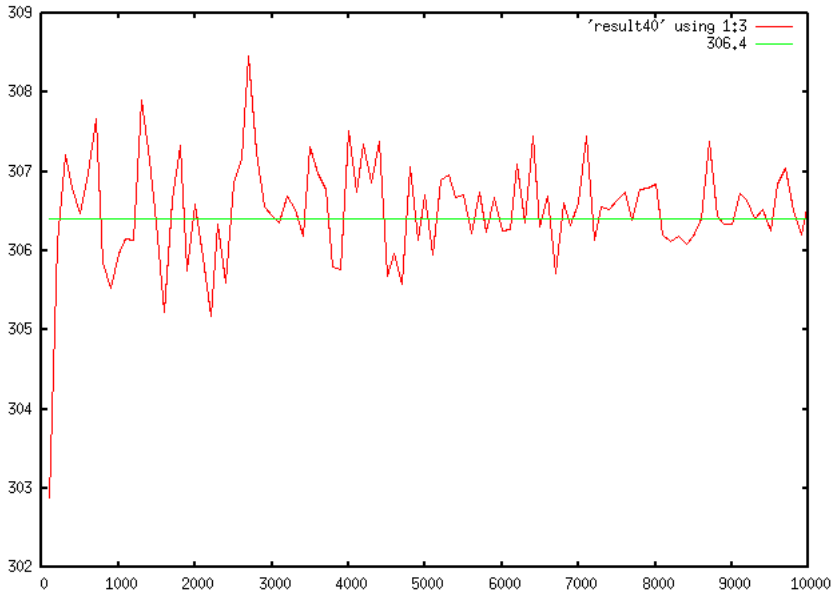


Fig. 4 $n = 40$

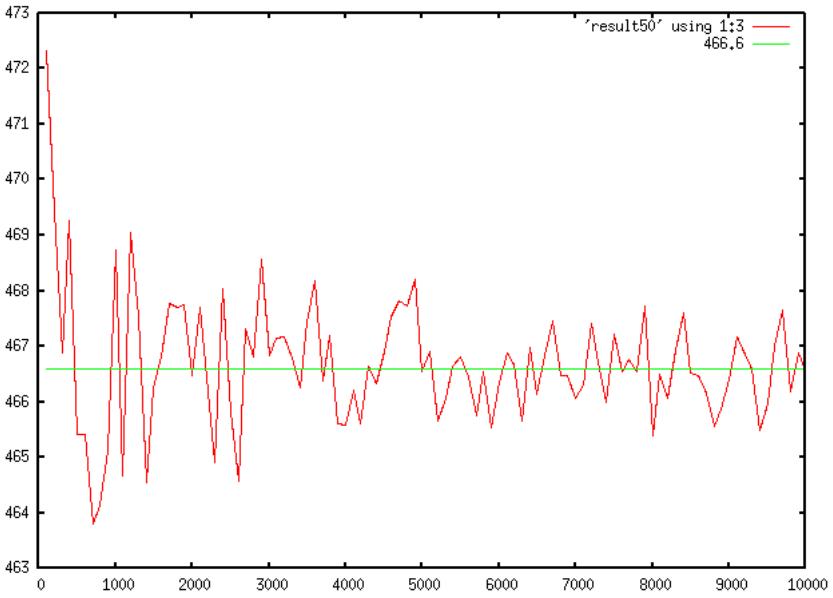


Fig. 5　$n = 50$

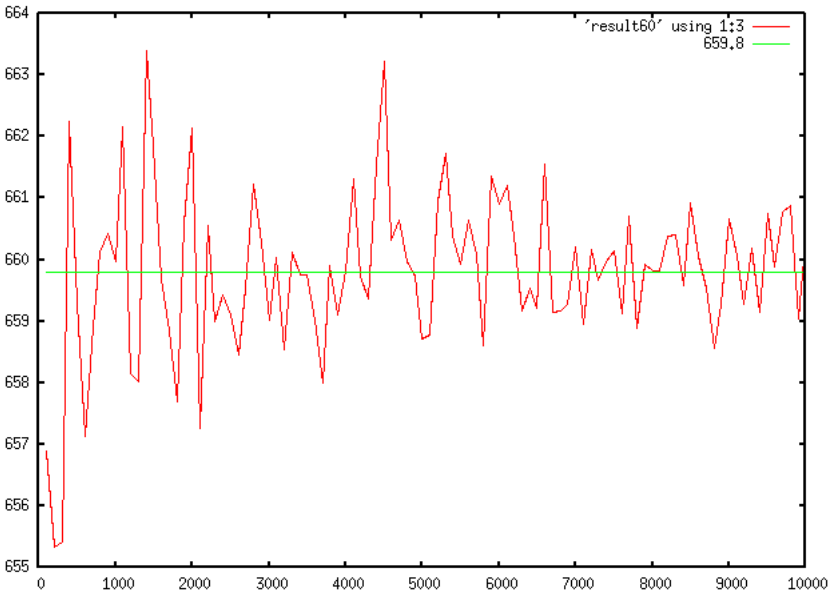


Fig. 6　$n = 60$

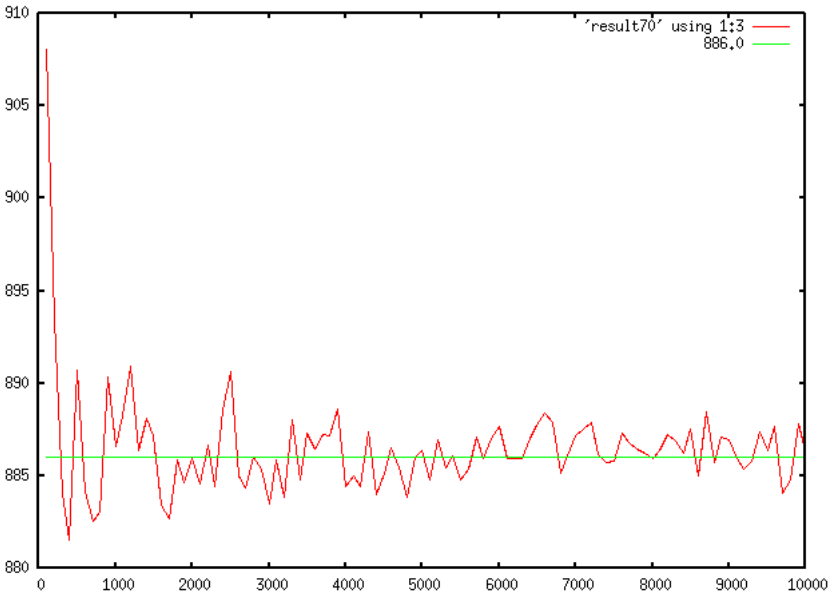


Fig. 7 $n = 70$

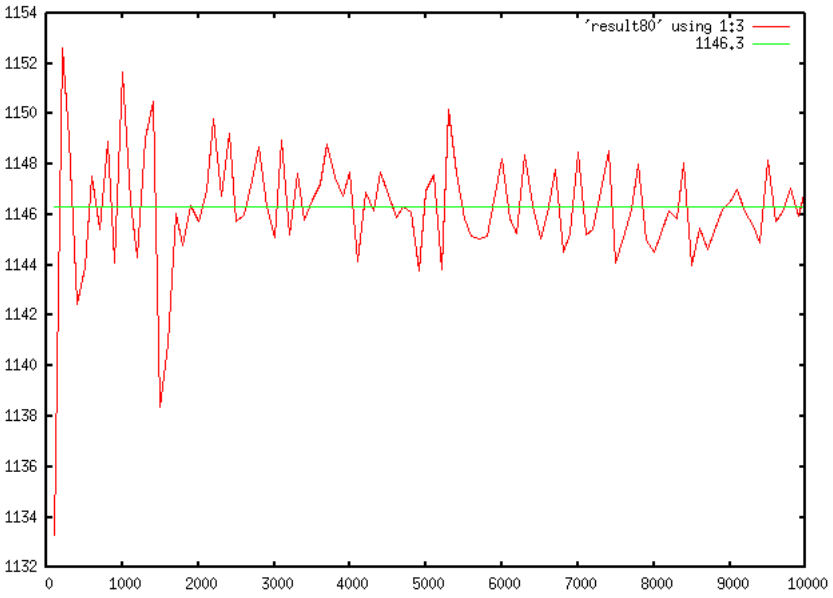


Fig. 8 $n = 80$

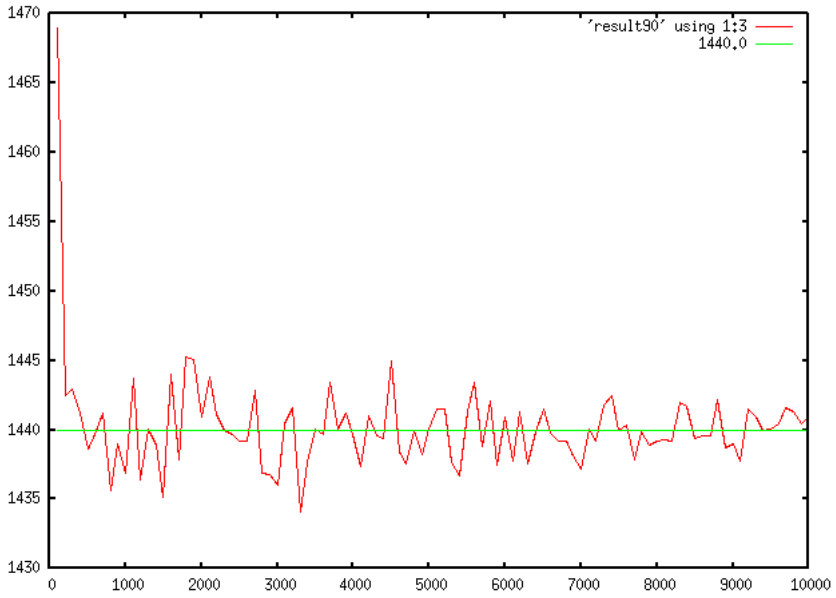


Fig. 9　$n = 90$

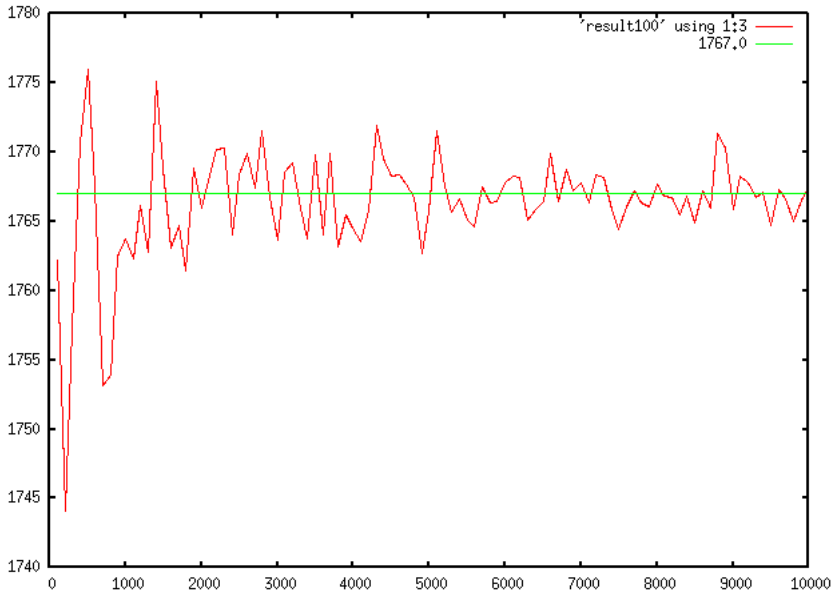


Fig. 10　$n = 100$

紙の枚数	平均回数
10	26.5 ± 0.1
20	86.5 ± 0.2
30	179.5 ± 0.2
40	306.4 ± 0.3
50	466.6 ± 0.5
60	659.8 ± 1
70	886.0 ± 1
80	1146.3 ± 1
90	1440.0 ± 1
100	1767.0 ± 1

紙の枚数 $n = 10$〜100 のそれぞれについて求められた平均回数をグラフにすると以下のようになった。

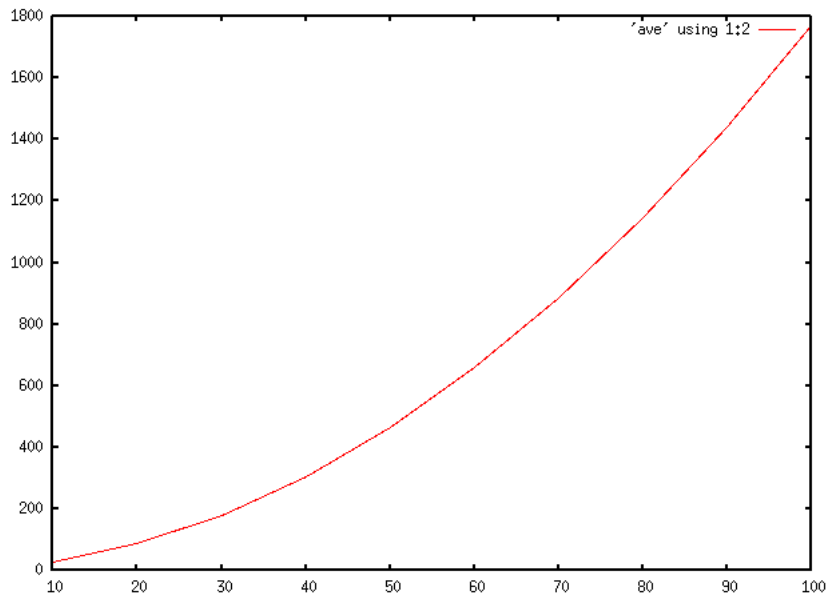


Fig. 11 n の増加による平均回数の推移

4 まとめ

本稿では、コンピュータ利用したソーティングについて、ひとつのルールを設定して、そのルールの下での最適なソーティングの手順の求め方、その手順と回数、シミュレーションによる平均回数を考察してきた。その結果、今回設定したようなルールの下では、

ソーティングするのに、かなりの回数の手順がかかることが解った。少なくともこのルールの設定のままではあまり実用にはならないと考えられる。しかし、ルールの設定を変えてやれば、もっと少ない回数でのソーティングも可能となるはずなので、人間が実際に行う並べ替え作業に、コンピュータを利用してソーティングのサポートをさせることには、十分意味があると考えられる。

今回の研究では、コンピュータ利用したソーティングについて、ルールを設定して、そのルールの下での最適なソーティングの手順を、コンピュータに求めさせるところまでしかできなかった。なので、今後の課題として、最適なソーティングの手順を音声で出力出来るようにすること、そのうえで、実際に人間が並べ替え作業をしてみて、実際のやり易さなどの考察が考えられる。他に、平均回数の理論的な導出や、並べ替えの回数が少なくて済むようなルール設定の考察、種々の方法の比較なども今後の課題である。

参考文献

[1] R.Sedgewick : アルゴリズム *C* 第 *1* 巻 基礎・整列 (近代科学社,1996)

[2] 柴田 望洋・辻 亮介 ： 新版 *C* 言語によるアルゴリズムとデータ構造(ソフトバンクパブリッシング株式会社,2005)

[3] ソートのプログラムの流れ
http://www.dais.is.tohoku.ac.jp/~shioura/teaching/s-info2/sort.ppt

[4] ソートアルゴリズムの種類
http://www.dais.is.tohoku.ac.jp/~shioura/teaching/s-info2/quick.ppt

[5] ソート (整列)
http://www.rsch.tuis.ac.jp/~ohmi/software-basic/sort.html

[6] シェルソート
http://www1.cts.ne.jp/~clab/hsample/Sort/Sort4.html