区間減少ソーティングネットワークの提案

黒田久泰†

本論文では，比較区間を減らしていく区間減少ソートというソーティングネットワークを提案し，0－1原理と最良優先探索を用いる区間減少ソートにおいて最も比較回数が小さくなる場合を計算機上で求める方法について述べる．その結果，n を要素数としたときソーティングネットワークが正しくソートすることを検証する時間が $O(1.618\cdots^n)$ である方法を得た．また，最良優先探索で効率の良いヒューリスティック評価関数を適応させることにより，最適な区間減少ソートではバイトニックソートと大差のない比較回数で正しくソートできることが分かった．

A Proposal of Gap Decrease Sorting Network

HISAYASU KURODA†

This paper proposes sorting network which we call "Gap Decrease Sort" for which the distance between compared elements (the gap) is reduced step by step. The paper also explains how to seek the minimum number of element comparisons in the sorting network. For proving the correctness in the sorting networks, we applied the zero-one principle and succeeded that it can be verified with $O(1.618\cdots^n)$ units of time where n is the number of elements to be sorted. Using more informative heuristic function on best-fast search, Gap Decrease Sort is found to be equal to bitonic sorters with respect to the number of element comparisons.

1. はじめに

ソーティングを行うアルゴリズムには，要素の数値的性質や個数によって様々な方法があるが，要素の数値的性質を利用しない場合には，もっぱら2つの数値間の大小比較でソーティングを行うことになる．本稿では，比較区間を最大幅から1ずつ減らしていく「区間減少ソート」というソーティングネットワークを提案し，その中で最も比較回数の少なくなる最適な区間減少ソートを求める手法について述べる．

n を要素数としたときソーティングネットワークにおいては，すでに，Ajtai, Komlós, そしてSzemerédiらが，$O(n\log n)$ 時間のソーティングネットワークを提唱しているが[1)~3)]，オーダ記法に隠れている係数が巨大であるため，実用的にはあまり有用なものではない．また，$O(n\log^2 n)$ 時間のソーティングネットワークとして，Batcher[4)]が提唱した奇偶マージソートやバイトニックマージソートなどがあげられるが，これらはソーティングの方法が再帰的に定義されているため，実際にプログラミングした場合に再帰呼び出しのために $O(n)$ 時間かかるため，n がそれほど大きくない場合には再帰呼び出しの時間が無視できなくなる．そのため，これらのソーティング法をどのように計算機上で実装していくかなどの研究がなされている[5)]．しかし本論文で提案する区間減少ソートでは，要素数が与えられたとき，どの区間で比較を行えば完全となるかという比較区間数列を知ることができれば，容易にプログラミングを行えるとともに，再帰呼び出しにともなう実行時間の増大を避けることが可能となる．

なお，Pratt[6)]がシェルソートで，最悪の場合でも $O(n\log^2 n)$ 時間となるアルゴリズムを示した．これを利用すれば，最適な区間減少ソートにおいても $O(n\log^2 n)$ 時間のソーティングネットワークであることが示される．

本論文の2章でソーティングネットワークの説明とその性質を示す．3章でソーティングネットワークの正しさを検証する手段として知られている0－1原理の効率の良い計算機上での実装法について述べる．この方法によると従来 $O(2^n)$ の検証法として知られる0－1原理が $O(1.618\cdots^n)$ となることを述べる．4章で筆者の提案する区間減少ソートについて述べる．5章で区間減少ソートの性質などを用い，改善された区間減少ソートを求める方法を述べる．6章で最も比

† 東京大学大学院理学系研究科情報科学専攻
Department of Information Science, Graduate School of Science, University of Tokyo

較回数の少なくなる最適な区間減少ソートを求める方法を示し，要素数が40までの具体的な結果を述べる．7章で最適な区間減少ソートの時間計算量のオーダを求め，他のソーティングネットワークとの比較を行う．

2. ソーティングネットワークについて

2.1 ソーティングネットワーク

ソーティングネットワーク[7)~10)]は，並列ソーティングアルゴリズムを簡潔に表現する便利な計算モデルの1つとして知られている．これは，入力列が与えられたとき，ソーティングを行う前にあらかじめ比較交換の箇所が分かっているからである．

ソーティングネットワークは，データを運ぶ複数のワイヤと多数の比較交換器（compare-exchange，以下では単に比較器と呼ぶ）から構成される．比較器は**図1**のようにそれぞれ2つの入出力端子を持ち，入力 x_1，x_2 に対し，出力 y_1，y_2 は次式で定義されるものである．

$$y_1 = \min(x_1, x_2)$$
$$y_2 = \max(x_1, x_2)$$

たとえば，入力 (x_1, x_2) として $(5, 2)$ を与えると，比較器の上限の端子には2が，下側の端子には5が出力される．

比較器を簡潔に表現するため，**図2**のように表すものとする．

比較器を用いたソーティングネットワークの例を**図3**に示す．

ここで，ソーティングネットワークの基本的性質をあげておく．

性質1・・・比較器の順番を入れ換えることはできない．

性質2・・・正しいソーティングネットワークに比較器を付け加えた場合，それが正しいソーティングネットワークであるとは限らない．

図4における左側の例は3入力の正しいソーティングネットワークである．ところが，右側のように矢印で示す比較器を追加すれば，これは，正しいソーティングネットワークではなくなる．

性質2は正しいソーティングネットワークではないものでも比較器をいくつか削除することで（つまり追加に限らずに），それが正しいソーティングネットワークとなりうることがあるということを述べている．

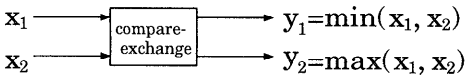


図1 比較交換器
Fig. 1 Compare-exchange equipment.

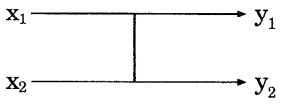
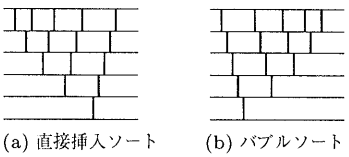
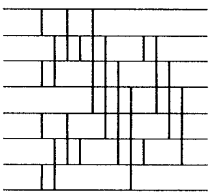


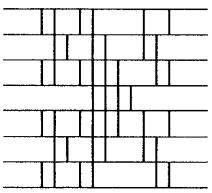
図2 比較交換器の簡略表現
Fig. 2 Simple representation of the compare-exchange equipment.



(a) 直接挿入ソート (b) バブルソート



(c) 奇偶マージソート



(d) バイトニックマージソート

図3 ソーティングネットワークの例
Fig. 3 Some examples of sorting networks.

図4 ワイヤを追加してうまくいかなくなる例
Fig. 4 Example of incomplete sorting network with additional wire.

3. ソーティングネットワークの検証法

3.1 0−1 原理

ソーティングネットワークに限定すれば，正しくソートされることの証明に，0−1原理という効率の良い検証手法が利用できる．

0—1 原理

n 個の入力を持つソーティングネットワーク A が，$\{0,1\}$ 上のあらゆる入力列（2^n 個存在する）を正しくソートできれば，A は整数，実数など線形順序が決まる任意のデータ集合から選ばれた任意の入力列を正しくソートできる．

この方法によれば，n 入力の場合 $O(2^n)$ 時間で検証できる．次節でこれを効率良く計算機上で実行させる方法について考察する．

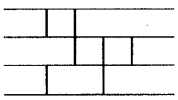
3.2 0—1 原理を利用した検証アルゴリズム

0－1 原理では，$\{0,1\}$ 上のあらゆる入力例を与えるが，最初の状態ではすべて不定の意味を持つ $\{\#\}$ の入力列を与えておく（実際に計算機上で実行させるときには -1 などでよい）．

比較の箇所で入力される 2 つの入力値に基づき次のような処理を行うことで正しく検証が行える．

- (# #) …… (0 0) と (# 1) の 2 通りに分け，それぞれ次のステップにいく．なお (0 #) と (1 1) の 2 通りに分けてもよい．
- (# 0) …… (0 #) として次へいく．
- (1 #) …… (# 1) として次へいく．
- (1 0) …… (0 1) として次へいく．
- (# 1) (0 #) (0 0) (0 1) (1 1) …… 変更せず次へいく．

最終ステップまたは途中の段階で (0…01…1) あるいは，(0…0#1…1) となれば，その分岐については正しくソートされる．そして枝分かれしたすべての分岐についてこれを確認すればよい．



問題 1

問題 1 の検証

```
(# # # #)
  (0 0 # #)
    (0 0 0 0) -
    (0 0 # 1) -
  (# 1 # #)
    (# 1 0 0)(0 1 # 0)(0 0 # 1) -
    (# 1 # 1)
      (0 1 0 1)(0 1 0 1)(0 0 1 1) -
      (# 1 1 1) -
```

この例だと正しくソートされる．

3.3 0—1 原理を利用した検証アルゴリズムの計算ステップ数

入力が (# #) 以外のすべてで分岐は起こらないので，(# #) のときだけを考えればよい．実際，

```
(# # # # …)
  (0 0 # # …)
    ……
  (# 1 # # …)
    ……
```

のように 2 つのブロックに分かれるので，分岐の総数 $T(n)$ は

$$T(n)=T(n-2)+T(n-1) \quad T(1)=1, T(2)=2$$

と書くことができる．これはフィボナッチ数列と呼ばれるもので，

$$T(n)=\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right)$$

となる．ここで，

$$-1<\frac{1-\sqrt{5}}{2}\ (=-0.6180\cdots)<1$$

であるので，

$$T(n)\approx\left(\frac{1+\sqrt{5}}{2}\right)^{n}$$

である．なお，$\frac{1+\sqrt{5}}{2}$ は黄金比と呼ばれる値であり，これより，$T(n)\approx(1.618\cdots)^n$ であることが分かる．

ここで比較器の個数を m として，通過する比較器の総数について少し詳しく解析する．0－1 原理では通過する比較器の総数は正確に $m2^n$ となる．ここで述べた方法で最悪の場合は，最初の $n-1$ 個の比較器ですべての分岐が行われたときである．その場合は，すべての分岐を終了した時点で $2T(n)-1$，その後については分岐の総数 $T(n)$ に比較器の残りの個数をかけた $(m-(n-1))T(n)$ 個の比較器を通過する．結局これらを足し合わせた $(m-n+3)T(n)-1$ が最悪の場合に通過する比較器の総数となる．以上で，本章で述べた方法で $O(1.618\cdots^n)$ 時間の検証となることがいえた．

4. 区間減少ソート

4.1 区間減少ソートの提案

本論文では，**図 5** のようなソーティングネットワークを提案する．これが正しいソーティングネットワークとなることの証明は省略する．

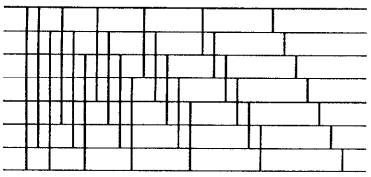


図5 本論文での提案（区間減少ソート）
Fig. 5 Proposal in this paper (gap decrease sort).

区間減少ソート

n 個の入力が与えられた場合，比較区間の最大幅は $n-1$ である．そこで，比較区間 $n-1$ から始め，1ずつ比較区間を減らしていき，最後に比較区間 1，つまり隣あったものを比較する．

比較区間が同じ k のとき，$n-k$ 箇所の比較を行うことになるが，この場合は図5のように上から順序よく比較を行うことにする．

4.2 改善された区間減少ソート

直接挿入ソートやバブルソートではあらかじめ不必要箇所を削除するようなことはできないが，区間減少ソートについてはそれが可能である．そこで，比較器をできるだけ削除することを考える．

ただし，ここでは，同じ区間（要素数 n で比較区間 k のときは $n-k$ 箇所ある）をまとめて削除するものとする．

そこで，区間減少ソートから不必要箇所を1区間（広義には1箇所であるが，ここでは，同じ比較区間をまとめて考えているので1区間とする）でも除いたものを，「改善された区間減少ソート」と呼ぶことにする．

なお，5章でここで述べる改善された区間減少ソートをすべて見つける方法について考察する．

4.3 最適な区間減少ソート

改善された区間減少ソートは，同じ要素数 n でもいくつかの組合せができる．この中で最も比較回数を少なくしたものを「最適な区間減少ソート」と呼ぶことにする．なお同じ要素数 n に対し最適な区間減少ソートは複数個存在する場合もある．

「最適な区間減少ソート」の例を図**6** に示す．この場合，要素数8で，比較区間が7と5，つまり点線で示した比較器の部分を削除したものが，他の改善された区間減少ソートの中でも最も比較回数が少なくなる組合せである．

なお，6章でこの最適な区間減少ソートを見つける方法について考察する．

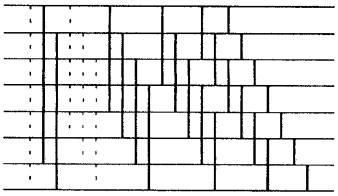


図6 最適な区間減少ソート
Fig. 6 Perfect gap decrease sort.

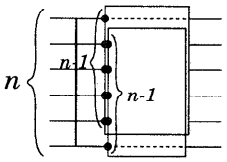


図7 ソーティングネットワークの合成
Fig. 7 Synthesis of sorting networks.

5. すべての改善された区間減少ソートを見つける手法

5.1 区間減少ソートの性質

ここで，さらに区間減少ソートに関する性質を述べる．これらの性質を1つでも多く利用することで，効率良く改善された区間減少ソートを見つけることができるようになる．ここで，$(a\ b\ \cdots)$ というのは比較が必要な区間を表すことにする．

性質3 $\cdots$ $n-1$ 入力の場合に必要な比較区間に，$(n-1)$ を加えたものは必ず n 入力においても正しいソーティングネットワークとなる．

例をあげれば，要素数8で，(6 4 3 2 1) の区間の比較を行えばよいとする．この場合，要素数9のときは，(8 6 4 3 2 1) の区間を比較することで正しくソーティングが行える．

一般に，図**7** に示すように合成されたものがソーティングネットワークであることを証明する．ここで，四角で囲った部分は $n-1$ 入力の正しいソーティングネットワークとする．

証明

証明には0－1原理を利用する．最初の比較器を通過した時点で，最上端が0であるか，または，最下端が1であるかのどちらかになる．最上端が0であれば，下側のソーティングネットワークによりこれは正しくソーティングが行える．また，最下端が1である場合にも，上側のソーティングネットワークにより正しくソーティングが行える．以上のことから図7は正しいソーティングネットワークである． **証明終**

これより，性質3が真であることは明らかである．

ここで，注意すべきことは，最初の最上端と最下端を結ぶ比較器が，必ずしも必要なものではないということである．改善された区間減少ソートを見つけることは，この最初の比較が必要ではない場合を調べていくことにもなる．

性質4 … $(n-1)$ 入力でソーティングの行えなかった比較区間の組と，それに比較区間 $n-1$ を加えた組も，n 入力では正しいソーティングネットワークとはならない．

例をあげると，要素数8で，(7 5 3 2 1) では正しくソーティングが行えないとする．この場合，要素数9のとき，(7 5 3 2 1) と (8 7 5 3 2 1) もまた正しくソーティングが行えない．

証明

証明には $0-1$ 原理を利用する．n 入力のソーティングネットワークを作り，最下端に1を入れてみればよい．最下端を除く $n-1$ 入力のソーティングネットワークが正しくソーティングを行えないものであれば，全体としても正しくソーティングが行えるものではない． **証明終**

これより，n 入力のとき増えるものは（$n-1$ 入力の組）および，それに比較区間として $n-1$ を加えたものに限る．もちろん，すべてではない．

また，性質4を利用すれば次のようなことがいえる．

「比較区間として1，2，3の区間は必ず必要である」

証明

4入力のときに，(3 2 1) とすべての区間の比較が必要であるため（5入力のとき初めて (4 3 2 1) と (3 2 1) の2組が解となる）． **証明終**

5.2 すべての改善された区間減少ソートを見つけるためのアルゴリズム

各 n の値について，削除可能な比較区間の組は次のように見つける．

n 入力のものを調べたいとき，まず $n-1$ 入力のものを調べる．したがって処理は再帰的になる．

性質3を使うことで，$n-1$ 入力の場合に必要な比較区間に，$(n-1)$ を加えたものは必ず n 入力においても正しいソーティングネットワークとなる．あとは，$(n-1)$ を加えないものが，正しいソーティングネットワークであるかどうかを調べていけばよい．逆に性質4よりそれ以外を調べる必要はない．

この方法により，すべての改善された区間減少ソートを効率良く見つけることが可能である．

6. 最適な区間減少ソートを見つける手法

6.1 最良優先探索を用いたアルゴリズム

要素数 n が与えられたとき，最適な区間減少ソートを見つけるだけでよいならば，ヒューリスティック評価関数を用いた最良優先探索を用いることが考えられる[11),12)]．そこで，5.1節の性質3，性質4を使い効率良く最適解を見つける方法について述べる．

ここでは，**図8** のような木構造を考える．

各節点は2つの子を持ち，右側の子は比較区間として自分の持つ要素（深さから1を引いた値）を追加するが，左側の子は追加しないものとする．

ここで，最良優先探索でのヒューリスティック評価関数 f は節点を p とすると，次のように定義される．

$$f(p) = g(p) + h(p)$$

g，h はそれぞれ g^*（初期節点から節点 p までの最小コスト），h^*（節点 p から目標節点までの最小コスト）の推定値である．また，木探索の場合は，つねに $g(p) = g^*(p)$ である．

ここで，最適な区間減少ソートを見つけるには，$g(p)$ には，節点 p での比較回数（要素数はその節点の深さと同じ値），$h(p)$ には，節点 p で決まっている比較区間を深さ n のときに適用させた場合，どれだけ比較回数が増えるのか，その値を入れればよい．

$$g(p) = g^*(p) = \sum_{i=1}^{k} \{\mathrm{depth}(p) - a_i\}$$

$$h(p) = k \times \{\mathrm{depth}(P) - \mathrm{depth}(p)\}$$

k は節点 p での比較の対象となる区間の個数，a_i は i 番目の区間の値である．$\mathrm{depth}(p)$ は節点 p の深さ（あるいは要素数）であり，$\mathrm{depth}(P)$ は n に等しい（P は目標節点）．

最良優先探索では，つねに $f(p)$ が最小となる節点を探し，それが目標節点である（この場合なら $\mathrm{depth}(p)$ が n に等しい）なら，探索が終了する．$\mathrm{depth}(p) < n$ なら，節点 p を展開する．

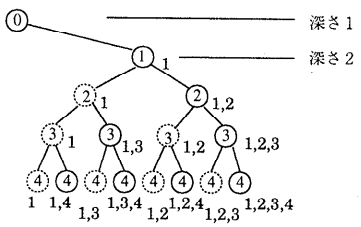


図8 比較区間を要素とする木構造

Fig. 8 A tree structure based on different compared distances.

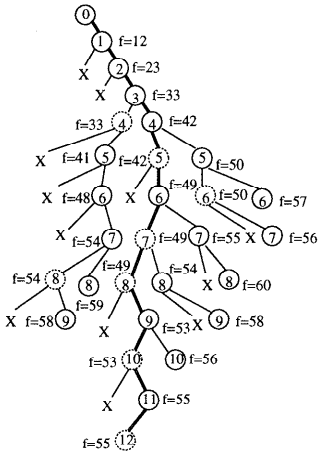


図9 $n = 13$ の場合の探索例
Fig. 9 Example of searching for $n = 13$.

ここでは具体的に $n = 13$ の場合の実行後の木構造の例を図9に示す.

$n = 13$ の場合には，このように区間1 2 3 4 6 9 11の比較を行えばよいことが分かる（実際には大きい順に比較する）．ただし，ここで，注意すべきことは，要素数 n に対する最適な区間減少ソートは得られるが，このとき探索した木構造からは，n より小さい場合の最適な区間減少ソートの解は得られないことである．

6.2 最適な区間減少ソートの結果

要素数が40までについて探索することにより，最適な区間減少ソートは**表1**のようになった．

この表から，実際のプログラミングにおいては，ソーティングの対象となる入力列の要素数に対して，**表2**のように比較区間を設定すれば区間減少ソートとしては最も効率の良いものが得られる．

また，たとえば要素数10に対して39～40の比較区間数列を利用することも可能である（区間9を越えるものについては比較しないようにする）．

6.3 簡単な区間減少ソート

表1の比較区間を見ると，現れる数字に規則性がまったくないようである．**表3**に比較区間をある値から1ずつ減らしていくだけという簡単な場合に，それがどのような要素数のときに正しくソーティングを行えるのかを示す．

7. 考　　察

7.1 最適な区間減少ソートの計算量

n を要素数としたとき，Prattがシェルソートの研究[6]で，n より小さな数で 2^p3^q の形をしたものすべてからなる減少順を用いるなら，最悪の場合でも

表1 最適な区間減少ソートにおける比較区間
Table 1 Different compared distances in the perfect gap decrease sort.

要素数	比較を行う区間の組（総比較回数）
3	2 1 (3)
4	3 2 1 (6)
5	3 2 1 (9)
6	5 3 2 1 (13)
7	6 5 3 2 1 (18)
8	7 6 5 3 2 1 (24)
	6 4 3 2 1 (24)
9	6 4 3 2 1 (29)
10	9 6 4 3 2 1 (35)
11	9 6 4 3 2 1 (41)
12	11 9 6 4 3 2 1 (48)
13	11 9 6 4 3 2 1 (55)
14	13 11 9 6 4 3 2 1 (63)
15	14 13 11 9 6 4 3 2 1 (72)
16	15 9 7 6 4 3 2 1 (81)
17	15 9 7 6 4 3 2 1 (89)
18	17 15 9 7 6 4 3 2 1 (98)
	13 9 8 6 4 3 2 1 (98)
	13 12 6 5 4 3 2 1 (98)
19	18 13 9 8 6 4 3 2 1 (107)
	18 13 12 6 5 4 3 2 1 (107)
20	18 13 9 8 6 4 3 2 1 (116)
	18 13 12 6 5 4 3 2 1 (116)
21	20 18 13 9 8 6 4 3 2 1 (126)
	20 18 13 12 6 5 4 3 2 1 (126)
	18 12 9 8 6 4 3 2 1 (126)
22	18 12 9 8 6 4 3 2 1 (135)
23	18 12 9 8 6 4 3 2 1 (144)
24	23 18 12 9 8 6 4 3 2 1 (154)
25	23 18 12 9 8 6 4 3 2 1 (164)
26	25 23 18 12 9 8 6 4 3 2 1 (175)
	22 18 12 9 8 6 4 3 2 1 (175)
27	26 22 18 12 9 8 6 4 3 2 1 (186)
28	27 26 22 18 12 9 8 6 4 3 2 1 (198)
29	27 26 22 18 12 9 8 6 4 3 2 1 (210)
30	29 27 26 22 18 12 9 8 6 4 3 2 1 (223)
31	29 27 26 22 18 12 9 8 6 4 3 2 1 (236)
32	30 19 18 8 7 6 5 4 3 2 1 (249)
	27 18 13 12 9 8 6 4 3 2 1 (249)
33	27 18 13 12 9 8 6 4 3 2 1 (260)
34	33 27 18 13 12 9 8 6 4 3 2 1 (272)
35	34 33 27 18 13 12 9 8 6 4 3 2 1 (285)
36	34 33 27 18 13 12 9 8 6 4 3 2 1 (298)
37	36 34 33 27 18 13 12 9 8 6 4 3 2 1 (312)
	36 30 27 18 13 12 9 8 6 4 3 2 1 (312)
38	37 36 30 27 18 13 12 9 8 6 4 3 2 1 (326)
39	36 27 24 18 16 12 9 8 6 4 3 2 1 (341)
40	36 27 24 18 16 12 9 8 6 4 3 2 1 (354)

$O(n\log^2 n)$ 時間のアルゴリズムが得られることを発見した．この減少列は，本論文で述べた「改善された区間減少ソート」の一部分をなす．

具体的には，**表4**にある値で要素数 n より小さい値を減少順に並べて比較区間として採用すればよい．

ここでは，最適な区間減少ソートが $O(n\log^2 n)$ で

表2 入力数と最適な比較区間
Table 2 A number of input elements and the perfect compared distances.

要素数	比較を行う区間の組
2 ～ 7	6 5 3 2 1
8 ～ 15	14 13 11 9 6 4 3 2 1
16 ～ 17	15 9 7 6 4 3 2 1
18 ～ 20	18 13 12 6 5 4 3 2 1
21 ～ 26	27 26 25 23 18 12 9 8 6 4 3 2 1
27 ～ 31	29 27 26 22 18 12 9 8 6 4 3 2 1
32 ～ 36	34 33 27 18 13 12 9 8 6 4 3 2 1
37 ～ 38	37 36 30 27 18 13 12 9 8 6 4 3 2 1
39 ～ 40	36 27 24 18 16 12 9 8 6 4 3 2 1

表3 簡単な区間減少ソート
Table 3 Simple gap decrease sort.

比較区間	要素数（総比較回数）
1	2(1)
2~1	3(3)
3~1	4(6) 5(9)
4~1	6(14)
5~1	7(20)
6~1	8(27) 9(33) 10(39) 11(45) 12(51)
7~1	13(63) 14(70) 15(77) 16(84) 17(91)
8~1	18(108) 19(116) 20(124)
9~1	21(144) 22(153)
10~1	23(175) 24(185) 25(195) 26(205)
	27(215) 28(225) 29(235) 30(245)
11~1	31(275) 32(286) 33(297) 34(308)
	35(319) 36(330) 37(341) 38(352)
12~1	39(390) 40(402) 41(414) 42(426)
	43(438) 44(450) 45(462) 46(474)
13~1	47(530) 48(543) ·····

表4 2^p3^q の値
Table 4 Values of 2^p3^q.

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
3^0	1	2	4	8	16	32	64	128
3^1	3	6	12	24	48	96	···	
3^2	9	18	36	72	···			
3^3	27	54	108	···				
3^4	81	···						

あることを示す．なお後で定数部分の比較を行うため，最高次数の係数は残している．

要素数 n が与えられたとき，比較区間数列として 2^p3^q の形をしたものすべてからなる減少順を設定する．最適な区間減少ソートは必ずこのように設定したものよりも総比較回数は小さくなるのは明らかなので，この設定で正しいソーティングネットワークが得られることを証明しさえすればよい．

ここでは，次の定理[7]を利用する．なお，a 順序付きとは，要素列を a 飛びで見たときに（このとき a 個の要素列になる）それぞれについてソートされていることを表す．

定理 $h > k$ のとき，h 順序付きの要素列を k 順序付きに整列しても，その要素列は h 順序付きのままである．

ここで，次の補題を証明する．

補題 比較区間 h を実行するとき，すでに，要素が $2h$ 順序付きでかつ $3h$ 順序付きであれば，挿入ソートなどを用いなくても1回のフェーズで h 順序付きとなる．

シェルソートでは「1回のフェーズ」でできるかどうかは問題とはしていない．しかし，ソーティングネットワークの場合は，事前に比較箇所を決めておくため，この「1回のフェーズ」が重要である．

補題の証明

まず，$2h$ 順序付きであることから，

$$a_0 < a_{2h} < a_{4h} < a_{6h} < \cdots$$

である．また，$3h$ 順序付きでかつ $2h$ 順序付きなので，

$$a_0 < a_{3h} < a_{5h} < a_{7h} < \cdots$$

これらのことから，

$$a_0 < \min\{a_{2h}, a_{3h}, a_{4h}, a_{5h}, \cdots\}$$

これより，a_0 と a_h の大小関係を調べて小さい方を a_0 に代入，大きい方を a_h に代入すればよい．これにより，

$$a_0 < \min\{a_h, a_{2h}, a_{3h}, a_{4h}, \cdots\}$$

が保証される．これを，$a_h, a_{2h}, a_{3h}, \cdots$ についても同じようにやっていけば，必ずそれより右側にある要素の中で最小となるものが代入される．これより h 順序付きの要素列となる． **補題証明終**

ここで，定理より $2h$ 順序列と $3h$ 順序列，そして $l > h$ となる l 順序列が決定されているものすべてについて順序列が保たれたままとなる．結局，2^p3^q の形をしたものすべてからなる減少順を用いることで，最後に $h = 1$ となったとき，1順序列すなわち整列が完了していることになる．

比較区間の個数

次に，n を要素数としたとき，$2^p3^q < n$ となる比較区間の個数のオーダを求めることを考える．

$0 \le p$, $0 \le q$ であることから，

$$0 \le p \le \lfloor \log_2 n \rfloor, \quad 0 \le q \le \lfloor \log_3 n \rfloor$$

である．また，

$$p + q\log_2 3 < \log_2 n \quad (\le \log_2(n-1) \text{ も可})$$

が成り立つ．

そうすると，比較区間の個数は，**図10** で実線で囲

まれた三角形の内部の格子点の数に相当する．それは，p 軸，q 軸，および階段状の太い線で囲まれた部分の面積に等しい．そのため，

$$\text{比較区間の個数} \geq \frac{\log_2 n \cdot \log_3 n}{2}$$

$$\text{比較区間の個数} \leq \frac{(\log_2 n + 2) \cdot (\log_3 n + 2)}{2}$$

が成り立つ．これより，比較区間の個数は $O(\log^2 n)$ であることがいえる．

比較回数のオーダ

ここで，2^p3^q の減少列を用いた場合の比較回数について考察する．

さきほどの p，q の範囲で，比較回数は次のようになる．

$$\text{比較回数} = \sum_q \sum_p (n - 2^p 3^q)$$

これより，$a = \lfloor \log_2 \frac{n}{3^q} \rfloor + 1$ とすれば，

$$\text{比較回数} = \sum_{q=0}^{\lfloor \log_3 n \rfloor} \left(an - 3^q(2^a - 1) \right)$$

この式から正確な比較回数が計算できる．

また，この式から比較回数のオーダを計算すると，$O\left(\frac{n(\log_2 n)^2}{2\log_2 3}\right)$ となることが分かる．

これにより，最適な区間減少ソートが $O(n\log^2 n)$ の比較回数であることが証明された．

しかし，最適な区間減少ソートそのものは，2^p3^q の数字列を使わなければならないということはないため，実際には，Pratt の示した方法より比較回数は小さくなる．

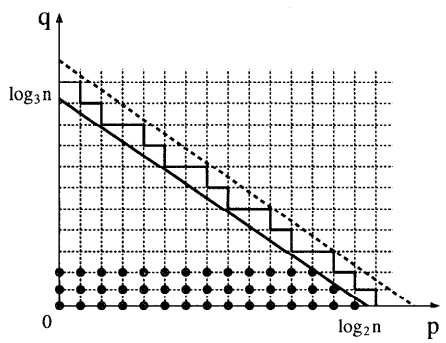


図 10 比較区間の個数
Fig. 10 A number of different compared distances.

7.2 他のソーティングネットワークとの比較

最適な区間減少ソートの比較回数と他のソーティングネットワークの比較回数とを比べてみる（**表 5**）．

これを見ると，比較回数はバブルソートに比べるとずいぶん小さくなることが分かる．また，バイトニックソートと比べても比較回数はそれほど大きな違いがない．

ところで，バイトニックソートの比較回数は

$$\text{比較回数} = m(m+1)2^{m-2}$$

であり，また，奇偶マージソートの比較回数は

$$\text{比較回数} = (m^2 - m + 4)2^{m-2} - 1$$

である（ただし $n = 2^m$）．

これは，どちらも $O\left(\frac{n(\log_2 n)^2}{4}\right)$ である．

そのため，

$$\lim_{n\to\infty} \frac{\text{バイトニックソートの比較回数}}{\text{奇偶マージソートの比較回数}} = 1$$

であるが，さきほどの Pratt の 2^p3^q の形の減少列を用いたならば，

$$\lim_{n\to\infty} \frac{2^p3^q \text{ の区間減少ソート}}{\text{バイトニックや奇偶マージ}} = \frac{2}{\log_2 3}$$

となる．

ここで，$\frac{2}{\log_2 3}$ の値は，およそ 1.26 である．

結局，最適な区間減少ソートでは定数部分の違いが 1.26 倍以下であることが分かる．

7.3 区間減少ソートとコムソート

Combsort（コムソート）[13]は，バブルソートのように隣接する要素を比較するのではなく，1 以上離れた要素を比較するようにしたものである．具体的には，最初の比較区間として要素数を 1.3 で割った商を当てはめる．以下その区間での比較が終了すると前回の比較区間を 1.3 で割った商を新しい比較区間としていく．

表 5 ソーティングネットワーク間の比較
Table 5 Comparison among some sorting netoworks.

n	=	4	5	6	7	8	9	10	11	12	13	14	15	16	···	32
バブルソート	=	6	10	15	21	28	36	45	55	66	78	91	105	120	···	496
区間減少ソート	=	6	9	13	18	24	29	35	41	48	55	63	72	81	···	249
バイトニック	=	6				24								80	···	240
奇偶マージ	=	5	9	12	16	19	26	31	37	41	48	53	59	63	···	191

この値が1未満になると，それ以後の比較区間は1とし，一連の操作中に要素の位置が変化しなくなったら終了する（最後はバブルソートに帰着する）．

最適な区間減少ソートでは，比較区間1での比較の終了時点でソートが完了しているという点でコムソートとは大きく異なる．また，区間減少ソートでは収縮率が一定していない．

8. まとめ

本論文で，区間減少ソートというソーティング方法を考案し，0－1原理や探索手法を用い，その比較回数が最小となる組合せを計算機により求めた．また，比較回数のオーダについても，奇偶マージソートやバイトニックマージソートと同じ $O(n\log^2 n)$ であることも分かった．

この区間減少ソートは，ソーティングの途中でソートが完了していたとしてもその判断は行っていない．しかし，その反面，比較とそれに付随した交換が1ステップで実行されるとすれば，区間減少ソートはどのような入力列に対しても一定の時間でソーティングを終えることができる．区間減少ソートは，平均実行時間ではソーティング終了条件を判断していないため速くはないが，どのような入力列に対しても一定の時間で終わるプログラム（終了条件のないバブルソートなど）としては，効率の良いものだと考えられる．

参考文献

1) Ajtai, M., Komlós, J. and Szmerédi, E.: An $O(n\log n)$ Sorting Network, *Proc. 15th. ACM Symp. on Theory of Comput.*, pp.1–9 (1983).
2) Ajtai, M., Komlós, J. and Szmerédi, E.: Sorting in $C\log N$ Parallel Steps, *Combinatorica*, Vol.3, pp.1–19 (1983).
3) Paterson, M.S.: Improved Sorting Networks with $O(\log N)$ Depth, *Algorithmica*, Vol.5, pp.75–92 (1990).
4) Batcher, K.E.: Sorting Networks and their Applications, *Proc. AFIPS 1968 Spring Joint Comput. Conf.*, pp.307–314 (1968).
5) Kumar, M. and Hirschberg, D.S.: An Efficient Implementation of Batcher's Odd-even Merge Algorithm and Its Application in Parallel Sorting Schemes, *IEEE Trans. Comput.*, Vol.C-32, No.3, pp.254–264 (1983).
6) Pratt, V.: Shellsort and Sorting Networks, Technical Report, Ph.D. Dissertation, Stanford U. Tech. Rep. STAN-CS-72-260 (1972).
7) Knuth, D.E.: *Sorting and Searching*, The Art of Computer Programming, Vol.3, Addison-Wesley (1973).
8) 梅尾博司：超並列計算機アーキテクチャとそのアルゴリズム，共立出版 (1991).
9) 宮野 悟：並列アルゴリズム理論と設計，近代科学社 (1993).
10) 梅尾博司：Batcher のバイトニック・マージ/ソート，*bit*, Vol.25, No.8–10, pp.98–100, 95–97, 100–102 (1993).
11) Korf., R.: Depth-first Iterative-deepening: An Optimal Admissible Tree Search, *Artificial Intelligence*, Vol.27, pp.97–109 (1985).
12) Korf., R.: Search: A Survey of Recent Results, *Exploring Artificial Intelligence*, Vol.27, pp.97–109, Morgan-Kaufmann (1988).
13) Lacey, S. and Box, R.: A Fast, Easy Sort, *BYTE*, Vol.APR., pp.315–320 (1991).

(平成8年 3 月4日受付)
(平成8年12月5日採録)

黒田 久泰（学生会員）

1970年生．1993年名古屋大学理学部物理学科卒業．1995年京都大学大学院工学研究科応用システム科学専攻修士課程修了．現在，東京大学大学院理学系研究科情報科学専攻博士課程在学中．探索アルゴリズムの研究に従事．