

いろいろな変種 Quicksort アルゴリズムの比較について†

梶 原 洋 一†† 有 澤 誠††

Quicksort については既にいろいろな研究がなされているが、どのような特徴をもったサンプルデータに対してどのようなアルゴリズムが有効かは必ずしも明確になっていない。そこで、種々の特徴をもつサンプルデータに対していろいろな Quicksort の変種アルゴリズムの有効性について実験的に検討した。また、そのアルゴリズムにカットオフを導入した場合には効率がどのように変わるかについても検討した。アルゴリズムの効率を比較するための尺度として、データ中の要素の比較回数および交換回数と、1つのデータに対しての分割アルゴリズム適用回数（再帰呼び出し回数）の3つを用いた。個々の特徴をもつある種類のデータに対するこれら3つの尺度をそれぞれ100回ずつ測定した。測定の結果、今回扱ったデータ全体を通して効率が良かったアルゴリズムは、カットオフを用いた擬中央値ソート、カットオフを用いた Dsort である。またすべてソート済み、あるいはすべて逆順のデータでは、カットオフを用いない Bsort, Xsort, Ysort が良いアルゴリズムである。あるデータが与えられた場合、そのデータはどのような特徴をもっているかがあらかじめ分かっている場合には、この実験結果が利用できる。

1. はじめに

Quicksort アルゴリズムは C. A. R. Hoare¹⁾ の提案以来、これまでにいろいろな変種や拡張版が研究されてきた^{2), 3)}。しかし、どのような性格をもつデータにどの変種アルゴリズムが適しているかは、現在のところ必ずしも明確になっていない。

そこで筆者らは、Quicksort のアルゴリズムを基礎とする種々の変種 Quicksort アルゴリズムが、どのようなサンプルデータに対して有効かを特徴づけるために、リスト要素の比較回数、リスト要素の交換回数、リストの分割回数（手続きの再帰呼び出し回数）について、理論的な値と実測値とを比較し考察した。

まず、本論文で使用する用語を説明する。リストとは、ソートの対象となっている配列の全部あるいは一部分のことである。したがって、リストの大きさ（要素数）は再帰呼び出しのたびに小さくなっていく。

データは、リスト中の具体的な要素であり、データの並びかたがある特徴をもっている配列をサンプルデータと言う。

キーは、リストを小さい部分と大きい部分の2つに分割する際の境界値である。キーに等しい値をもつ要素が必ずリストに存在するアルゴリズムもあり、そうとは限らないアルゴリズムもある。

リスト要素の比較回数、交換回数とは、プログラム

実行中に生じるサンプルデータ中のデータとデータの比較、交換の回数である。また分割（再帰呼び出し）回数は、分割アルゴリズム適用回数である。

カットオフ (cutoff) はある整数値であり、リストの大きさがこの値より小さいときには Quicksort 適用を打ち切るための境界値である。

Quicksort のアルゴリズムは再帰呼び出しを行うため、ソートの対象リストが小さいときにはオーバーヘッドが大きい。ある整数値 cutoff よりリストが小さい場合は再帰呼び出しをしないことによってオーバーヘッドが減る。最後に一度だけ insertionsort を適用すると、ソートが完了する。この機構を含めたものを、カットオフを用いた方法と呼ぶ。

本研究では、ランダム、すべてソート済み、すべて逆順などの種々の特徴をもつサンプルデータに対して、あるひとつのアルゴリズムの比較回数、交換回数、再帰呼び出し回数を100回ずつ計測し、その回数の分布のようすを解析した。この実験をカットオフのあるときとないときについて試み、カットオフの有効性などを考察する。この実験をいろいろな変種アルゴリズムに対して行い、アルゴリズムがどのようなサンプルデータに対して有効かを考察する。

以下、まず第2章で、本研究でとりあげるアルゴリズムについてそれぞれのねらいを説明する。第3章で、サンプルデータの性格ごとに、各アルゴリズムの比較回数、交換回数、再帰呼び出しの回数を、カットオフのあるときとないときについて求めたものについて、理論的な値と比較し、個々のアルゴリズムの特徴づけをする。第4章でまとめを述べる。

† A Comparative Study of Variants Quicksort Algorithms by YOICHI KAJIHARA and MAKOTO ARISAWA (Department of Computer Science, Faculty of Engineering, Yamanashi University).

†† 山梨大学工学部計算機科学科

2. いろいろな Quicksort アルゴリズム

ここでは、Quicksort のアルゴリズムを基礎とする合計 9 種類のソーティングアルゴリズムについて説明する。

なお紙面の関係で個々のプログラムは含めない。

2.1 基本となるアルゴリズム (Quicksort)

Quicksort は、リスト中のある要素キー (x) を適当に選び、リストをその値より小さい部分、大きい部分に分け、その分割されたサブリストに対しても同様な処理を再帰的に行うことによってリスト全体をソートする。キーによって分割が終了すると、それぞれのサブリストに含まれる要素の個数を調べ、2 以下の時は要素を直接比較して必要なら要素の交換を行う。

また、このソートにカットオフを導入すると再帰呼び出し回数が少なくなり効率が向上する。

基本的なアルゴリズムには、リストの先頭の値をキーとする方法と、リストの中央の値をキーとする方法がある。

既にすべてソート済み、あるいはすべて逆順のサンプルデータに対しては、リストの中央値をとる方法が最適なキーの選びかたをするが、リストの先頭をとる方法はキーとしてリストの最小値、あるいは最大値を選ぶため効率は非常に悪くなる。

リストの分割には両端から走査していく方法 (Wirth 版⁶⁾) と、左端からのみ走査していく方法 (Bentley 版⁴⁾) の 2 種類がある。それぞれのループアサーションは図 1 のようになる。

Bentley 版アルゴリズムは、未検査の部分の左端の値が x より大きい場合、その場所を指すポインタ p を 1 進めるだけでよいが、 x 以下の場合には要素の交換を行わなければならない。Wirth 版アルゴリズムは、未検査の部分の両端から走査していくが、左から走査しているポインタ q の指す内容が x より大きく、右から走査しているポインタ p の指す内容が x 以下の場合のみ要素の交換が起こる。

この 2 つの分割アルゴリズムの比較回数、交換回数、再帰呼び出し回数をランダムに選んだテスト用データで計数した結果、Bentley 版アルゴリズムは再帰呼び出し回数は Wirth 版アルゴリズムとほとんど同じであるが、比較回数、交換回数は、Wirth 版アルゴリズムの約 2 倍かかることが分かった。以下では分割アルゴリズムは Wirth 版を採用している。

2.2 擬中央値ソート

擬中央値ソートアルゴリズムは前処理として、リストの最小値と最大値を求め、キーとしてリストの最小値と最大値の平均をとる。ランダムサンプルデータに対するキーの選びかたは、基本 Quicksort に比べてより中央値に近いので効率が上がる。しかし前処理に時間がかかり過ぎると、効率はかえって悪くなる。同じ値が連続して現れるようなサンプルデータでは、基本 Quicksort は最悪 ($O(n^2)$) となるが、擬中央値ソートは最小値と最大値を求める前処理 1 回だけ ($O(n)$) でソートが完了する。

擬中央値ソートでは整数データを扱っている場合、分割後のリストの x より大きい部分のリストについては最小値を $x+1$ としているが、真の最小値はそれより大きいかもしれない。このみかけの最小値と真の最小値に大きな差があると効率が落ちる。しかし、みかけの最小値と真の最小値が 2 倍近く離れていたとしても、キーの値はリストの最小値と最大値の平均をとるため、1 回再帰呼び出しを行うことによって真の最小値に近づく。

2.3 meansort

meansort は前処理としてリスト要素の平均値を求め、それをキーとして用いる方法である。meansort は、リストが大きくなればなるほどキーを求めるために時間がかかり、効率が良いとは言えない。しかし、キーが中央値に近いことから分割の回数は基本 Quicksort より少なくなる。また、リストを一度走査しすべての要素がキー以上であったときにはそのリストはすべて同じ値であり、直ちに停止できる。

2.4 Dsort

Dsort (Dijkstra's Dutch Flag Sort) は擬中央値ソートで、リスト要素が整数値でかつリストの最小値と最大値の差が 2 以内のとき、すなわちリストの要素が 3 種類以下の値しか含まないときにはオランダ国旗アルゴリズム^{2),5)}を用いる方法である。次の 3 分割ソートの下準備をかねて含めてある。

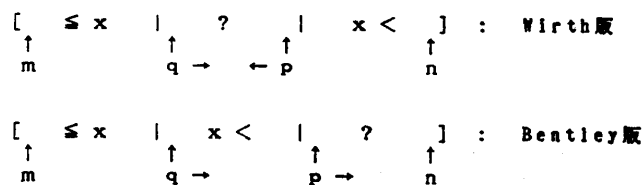


図 1 Wirth 版と Bentley 版アルゴリズムループアサーション
Fig. 1 Loop assertion of Wirth's and Bentley's algorithms.

Dsort にカットオフを導入すると、基本 Quicksort のときと同様に再帰呼び出しの回数が減る。ランダムサンプルデータではあまり重複度は高くなく、カットオフを導入すると要素数が少ないときは再帰呼び出しをしないために、オランダ国旗の問題の部分をあまり使用しなくなる。擬中央値ソートと比べ、重複度の多いサンプルデータに対して効果がある。

2.5 3分割ソート

3分割ソートはオランダ国旗アルゴリズムを一般化して、キーより小さい部分、等しい部分、大きい部分の3つのリストに分割する方法である。もとのオランダ国旗アルゴリズムを、再帰呼び出しを行うためのポインタ位置を調整するように書き換えてある。3分割ソートは、同じ値が頻繁に出現するようなサンプルデータに対してはリストの縮まりかたが速い。

2.6 桁分割ソート

桁分割ソートはデータをビット列とみなし、上位ビットからみていき、あるビットが0か1かでリストを分割する方法である。2の n 乗をキーに選んでいるためキーがリスト内に存在しないこともある。分割をした結果リストが2つにならない場合、すなわちデータのあるビットがすべて0または1のときには、再帰呼び出しはせずに手続きの先頭へジャンプするため再帰呼び出しは起こらない。この改良によって交換回数、再帰呼び出し回数は変化しない。サンプルデータの値の範囲が既知の場合には、キーはとるべきビット範囲が分かる。一般には多目にビット範囲をとらねばならない。

桁分割ソートの最悪の場合はリスト全体でただ1つだけ、ある1ビットがたっている次のようなデータに対してである。[2, 4, 8, 16, 32, 64, ...]

しかし、このようなデータは32ビットマシンの場合高々要素数は31個であり、現実には問題にはならない。

2.7 Bsort, Xsort, Ysort

これら3つは文献 9) に引用されているソーティングアルゴリズムで、大部分の要素が既にすべてソート済み、あるいはすべて逆順のサンプルデータに対して有効である。

Bsort アルゴリズムは、分割するときに左のサブリストの最大値、右のサブリストの最小値をその走査時のサブリストの右端、左端に置く。分割が終るとそれぞれのサブリストの最小値、最大値が分かっているため、基本 Quicksort に比べてリストの縮まる速さが速い。また、分割が終了するまでこれらの値が毎回変わったならそのサブリストがソート済みであると分かる。

Xsort と Ysort は、Bsort を拡張して両側のサブリストの最小値と最大値を覚えておく方法である。Xsort は Bsort と同じように最小値、最大値を swap によって覚えておき、Ysort はそれらの添字を覚えて

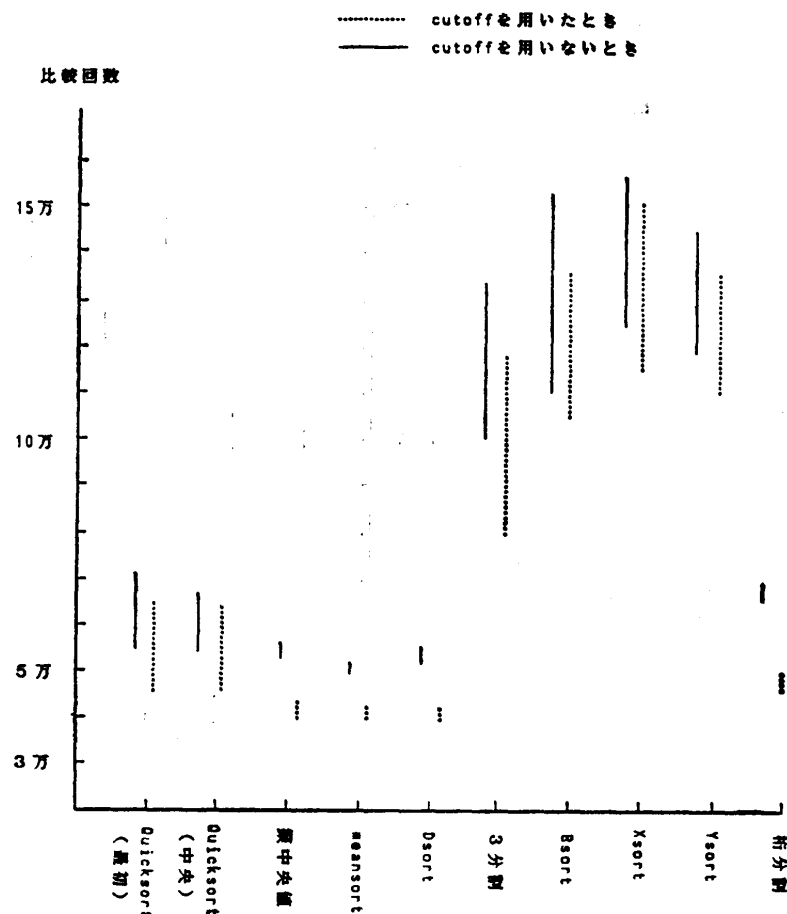


図 2 ランダムサンプルデータに対する比較回数
Fig. 2 The number of comparison for random sample data.

表 1 代表的な実験結果
Table 1 Characteristic experimentation result.

(msec)

サンプルデータ	ランダム		正 順		同 じ 値		Bsort が最悪	
カットオフ	な し	あ り	な し	あ り	な し	あ り	な し	あ り
基本 Quicksort (最初をキー)	250-333	250-350	18283-19866	7366-8034	13384-14600	13716-14634	20333-21700	450-584
基本 Quicksort (中央をキー)	250-350	233-350	200-283	117-184	13967-15467	14266-15650	750-850	717-833
擬中央値ソート	234-350	217-300	200-317	150-200	16-34	17-66	217-300	233-300
meansort	333-450	300-384	350-417	217-300	16-50	17-67	333-417	267-350
Dsort	216-333	200-284	234-316	150-200	16-34	16-67	233-300	233-300
3分割ソート	467-584	466-617	417-517	266-350	16-34	17-67	6750-7067	6717-7133
Bsort	616-833	650-767	17-50	33-83	16-50	33-67	21000-22284	21550-22683
Xsort	700-900	667-834	17-67	50-84	17-67	34-84	1917-2067	1900-2150
Ysort	483-634	517-633	16-50	33-83	16-50	33-67	800-1066	966-1067
桁分割ソート	416-550	316-450	434-550	267-366	433-534	533-633	466-533	400-483

おく。この改良で交換の回数はいくらか減少する。

これら3つのソートにカットオフを導入すると、ランダムサンプルデータの場合については効率が上がるが、既にソートされているサンプルデータについては、効率はかえって下がる。

3. いろいろなサンプルデータに対する比較実験とその結果

今回の実験で用いたサンプルデータはすべて整数であり、ランダムサンプルデータ、すべてあるいは大部分がソート済みまたは逆順のサンプルデータ、重複値のあるサンプルデータ、同一値のみのサンプルデータ、Bsort が最悪になるようなサンプルデータ¹⁰⁾である。サンプルデータの大きさは2000、測定回数はそれぞれ100回ずつ、カットオフ値は15である。代表的な実験結果をまとめたものが表1である。

3.1 ランダムサンプルデータの場合の実験結果

比較回数のばらつきが少なく回数も少ないアルゴリズムは、カットオフを用いた擬中央値ソート、

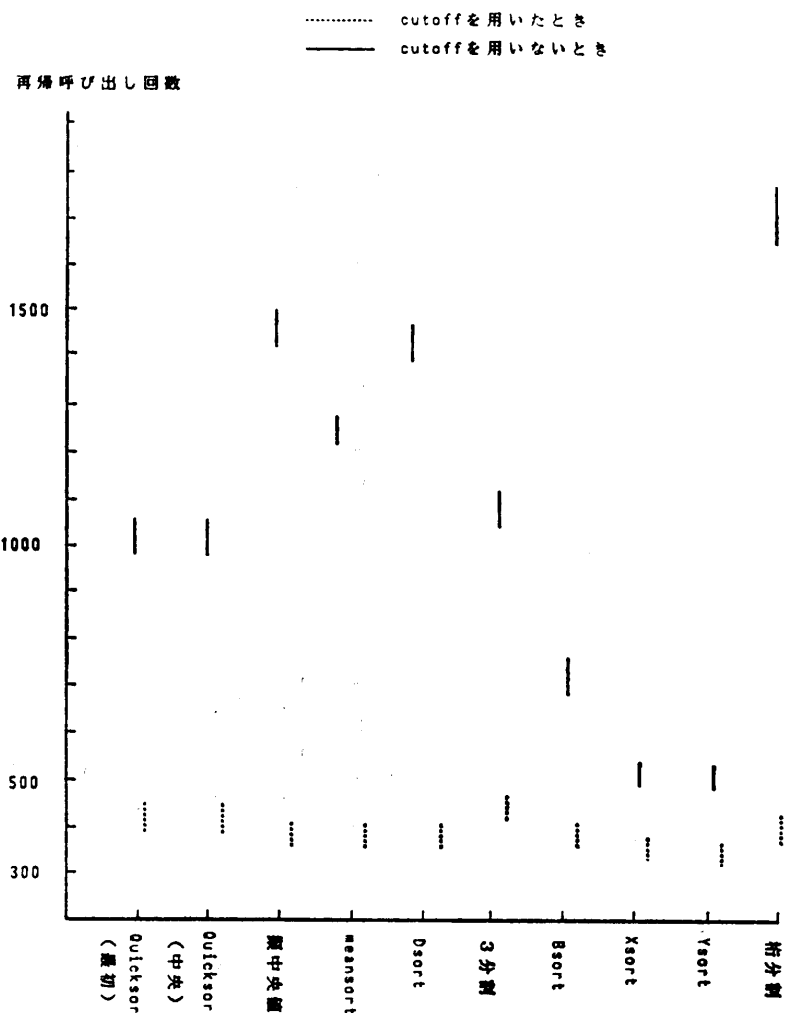


図 3 ランダムサンプルデータに対する再帰呼び出し回数
Fig. 3 The number of recursion for random sample data.

カットオフを用いた meansort, カットオフを用いた Dsort である。カットオフを用いた桁分割ソートもこの3つのアルゴリズムとほとんど同じくらいになる。基本 Quicksort は、カットオフを用いてもあまり改善はみられない。その他のアルゴリズムは、比較回数のばらつきも大きく回数自体も多い (図 2)。

交換回数については Bsort, Xsort, Ysort 以外のソートはカットオフの有無にかかわらずほとんど変わっていない。Bsort, Xsort, Ysort の3つのソートについては、カットオフの有無にかかわらず他のソートよりも3倍から6倍多い。

再帰呼び出し回数についてはカットオフありではすべてのソートが300-500回くらいになる (図 3)。カットオフなしでは、擬中央値ソートと Dsort が基本 Quicksort の約1.5倍であるが、カットオフを用いることにより基本 Quicksort とほぼ同じくらいになる。

この実験結果からランダムサンプルデータの場合、カットオフは非常に有効な手段と言える。また、交換回数、再帰呼び出し回数はカットオフを用いた場合はアルゴリズム間での差はあまりないことから、比較回数だけで効率を比べることができる。実行時間の結果も考慮すると、カットオフを用いた擬中央値ソートとカットオフを用いた Dsort が良いアルゴリズムだと言える。Bsort, Xsort, Ysort はランダムサンプルデータの場合は不利である。

3.2 重複のあるサンプルデータの場合の実験結果

今回は、10種類の値だけが2000個あるようなデータについて実験した。

比較回数については、2種類の基本 Quicksort, Ysort が異常に大きい。また比較回数だけでなく交換回数、再帰呼び出し回数についても、カットオフを用いた効果がない。データの重複度が増すと、基本 Quicksort はある段階からは最悪のケースである同じ

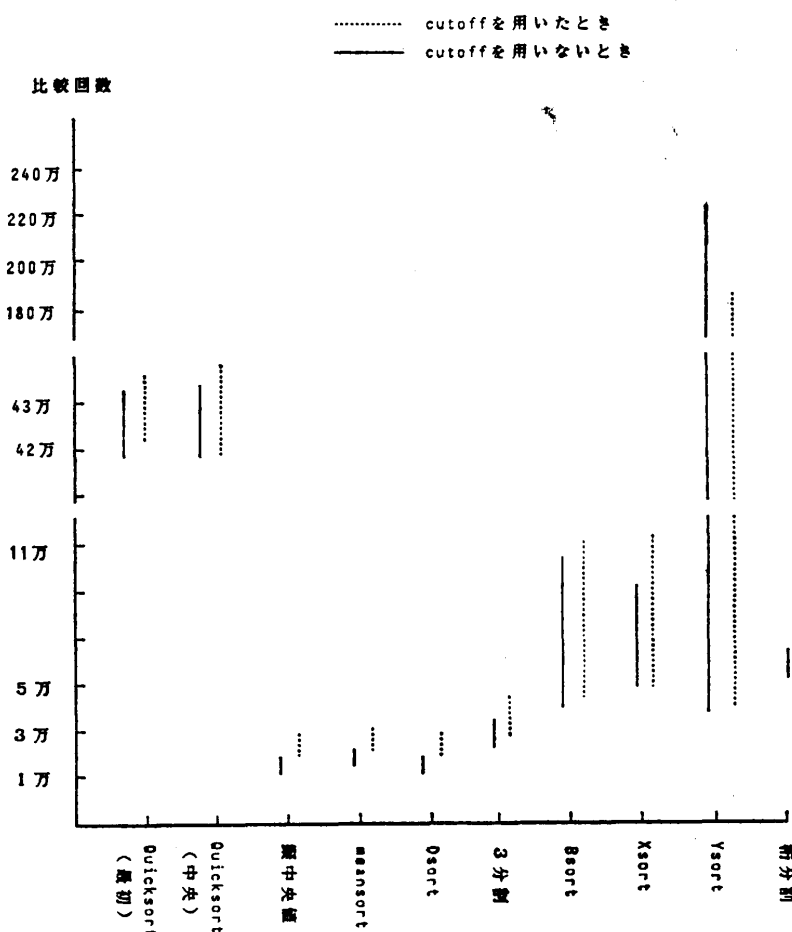


図 4 重複のあるサンプルデータに対する比較回数
Fig. 4 The number of comparison for duplicate sample data.

値の連続するデータとなり $O(n^2)$ となるため、比較回数がきわめて多くなる。Ysort は、Bsort, Xsort の改良版であるにもかかわらず、基本 Quicksort よりも比較回数が多くなる。擬中央値ソート, meansort, Dsort は、リストの中央値により近い値をキーとすることや、リストの最大値、最小値を事前に調べることによって、基本 Quicksort での最悪の場合を避けているため、比較回数は少ない (図 4)。

交換回数は、Ysort は Bsort, Xsort よりも改善されている。ここでは擬中央値ソート, meansort が回数が少なく、ばらつきも小さい。

再帰呼び出し回数では、Bsort, Xsort, Ysort でばらつきがある。Ysort は、Bsort, Xsort に比べきわめて多い。また、基本 Quicksort は、カットオフの有無にかかわらずリスト長だけ再帰呼び出しが起こる。

重複度データの場合、Ysort は比較回数、再帰呼び出し回数が大幅に増える。今回扱ったデータでは各

データの重複度が200とカットオフの値と異なっていたため、カットオフの有無にかかわらず計測結果は変わらないが、重複度が10-20とカットオフの値に近い値をとったときには、両者に差が出てくる。

3.3 同じ値のみからなるサンプルデータの場合の実験結果

基本 Quicksort はキーとして選ぶ要素の位置によらず最悪 ($O(n^2)$) となる。以下の論議では基本 Quicksort は比べる対象から除外する。

比較回数は、カットオフを用いることによって逆に増加しているが、桁分割ソートだけは異様なばらつきをしている。これは、分割のアルゴリズムによるばらつきで、あるビットが0ばかりのときと1ばかりのときとで、その1回の分割における比較回数がリスト長である n だけ違ってくるからである。また、擬中央値ソートと Dsort は、Bsort, Xsort, Ysort の4分の1程度の比較回数で済む。

交換回数と再帰呼び出し回数については、カットオフを用いても用いなくても全く同じ回数である。ただし、桁分割ソートの再帰呼び出し回数は様子が異なり、扱っているデータのビット数だけの再帰呼び出しが起こる。

3.4 大部分がソート済みあるいは逆順のサンプルデータの場合の実験結果

今回は90%程度がソート済みのサンプルデータについて検討した。比較回数は、すべてソート済み、すべて逆順ともカットオフを用いることによってわずかに減っている。リストの最初をキーとする基本 Quicksort はすべてソート済み、すべて逆順のデータの場合同様、他のアルゴリズムに比べ比較回数が非常に多い。これは、何回か再帰呼び出しを行っていくうちにリストの100%がソート済みになる場合があるためである。

再帰呼び出し回数については、カットオフを用いることによってリストの最初の値をキーとする基本 Quicksort 以外のアルゴリズムはだいたい200-400回になり、十分効率が良くなっている。

すべてソート済みのデータの場合のリストの最初の値をキーとする基本 Quicksort の比較回数が異常にばらついている。

3.5 すべてソート済みあるいは逆順のサンプルデータの場合の実験結果

基本 Quicksort は、キーを選ぶ位置がリストの最初であるか中央であるかの違いでアルゴリズムが2種類

ある。すべてソート済みまたはすべて逆順のデータに対しては、キーを中央から選ぶアルゴリズムはそのリストの中央値に近い値をキーとして選び、キーをリストの最初から選ぶアルゴリズムはリストの最小値または最大値をキーとして選ぶことになる。したがって、前者が有利である。

比較回数は Bsort, Xsort, Ysort 以外のソートは、カットオフを用いることによって改善されている。しかし、Bsort, Xsort, Ysort は逆カットオフによって悪い結果になる。

交換回数は、3分割ソートが異常に大きい値をとる。3分割ソートはリストを3つに分割する際、キーと等しい部分が大きければ大きいほど効率は良くなり、この部分が小さいときには効率は悪くなる。ここでのデータは、後者の場合に当たる。

オランダ国旗アルゴリズムでは、キーと等しい部分のすぐ左隣を検査してキーより大きい値ならば、その値とキーより大きい部分のすぐ左の値（キーと等しい部分の右端の値）を交換する。リストの未検査の部分が狭くなっていくときキーより大きい部分、キーと等しい部分はこのようなデータの場合、中央付近まで走査するまで一致している。したがってこの交換は同じ場所の値を交換しているため無駄である。

再帰呼び出しの回数については、リストの最初の値をキーとする基本 Quicksort, Bsort, Xsort, Ysort 以外は、カットオフを用いるとランダムサンプルデータと同じく300-400回になる。Bsort, Xsort, Ysort はカットオフを用いても変わらない。

Bsort, Xsort, Ysort は、比較回数、交換回数、再帰呼び出し回数のどれについても他のアルゴリズムよりも少なく、また実行時間の計測結果からもこのデータの場合に良いアルゴリズムであると言える。

3.6 Bsort が最悪となるようなサンプルデータの場合の実験結果

次の3種類のサンプルデータについて実験した。

1. [2 4 6 8...2000 1 3 5 7...1999]
2. [1 3 5 7...1999 1 3 5 7...1999]
3. [2000 1998...4 2 1 3 5 7...1999]

比較回数は Bsort が桁違いに多く、ついで3分割ソートが多い。3分割ソートは、リストを分割する際キーと等しい部分が大きいとき効率が上がるが、ここで扱っているデータはキーと等しい部分の大きさが常に1であるため、効率は悪い。また、Bsort と3分割ソートに共通して言えることは、カットオフを使って

も再帰呼び出しの回数が他のアルゴリズムとほとんど変わらないことである。カットオフを用いた擬中央値ソート, Dsort, meansort は再帰呼び出し回数が3つのデータともに同じ回数になる。これは今回扱っているデータが, 1 から 2000 までの数の出現する回数が固定しているため, リストの平均値, 中央値がほとんど一致してしまうことによる。

比較回数, 交換回数, 再帰呼び出し回数を通してみると, 擬中央値ソート, meansort, Dsort が3つのデータすべてに対し効率が他のアルゴリズムよりも優れている。

4. おわりに

この研究の結果を要点のみまとめると以下のようになる。

- 擬中央値ソート, meansort, Dsort の3つのアルゴリズムにはあまり差異がみられなかった。分割するキーを計算するための手間を考えると, meansort は擬中央値ソート, Dsort に比べ効率が悪い。

- ランダムサンプルデータの場合, カットオフはオーバーヘッドの大きい再帰呼び出し回数を減らす効果があり, 非常に有効な手段である。

- ランダムサンプルデータの場合は, カットオフを用いた Dsort, 擬中央値ソートを用いるのが良い。

- すべてソート済みまたはすべて逆順のサンプルデータの場合は, カットオフを用いない Ysort が良い。

- データの重複度が多いサンプルデータの場合は, 擬中央値ソート, Dsort が良い。

- 桁分割ソートは最悪のケースがない。

- Bsort, Xsort, Ysort はランダムサンプルデータあるいは重複度の多いサンプルデータに対しては, 効率が悪い。特に Ysort は重複度の多いサンプルデータに対して, 比較回数, 再帰呼び出し回数が異常にばらつき, 最悪となる。

この結果を利用して, ソートすべきデータの性格に応じて適切な Quicksort の変種を選択することが可能になり, 研究の目的を達することができた。

謝辞 本研究に対して親身な助言をしていただいた井内稔技官および有澤研究室の諸氏に厚く感謝いたします。

参 考 文 献

- 1) Hoare, C. A. R.: Algorithm 63, Partition, Algorithm 64, Quicksort, *Comm. ACM*, Vol. 4,

No. 7, p. 321 (1961).

- 2) Dijkstra, E. W.: *A Discipline of Programming*, pp. 111-116, Prentice-Hall, Englewood Cliffs (1976). (浦 昭二, 土居範久, 原田賢一共訳: プログラミング言語論, pp. 128-134, サイエンス社, 東京 (1983)).
- 3) 梶原洋一, 有澤 誠: 種々の Quicksort アルゴリズムの比較研究, 第 32 回情報処理学会全国大会論文集, 1 F-7 (1986).
- 4) Bentley, J. L.: *Programming Pearls, How to Sort*, *Comm. ACM*, Vol. 24, No. 4, pp. 287-291 (1984).
- 5) 真野芳久: オランダ国旗問題とそのアルゴリズムについて, *電子通信学会論文誌*, Vol. J 62-D, No. 11, pp. 774-775 (1979).
- 6) Wirth, N.: *Algorithms + Data = Programs*, Prentice-Hall, London (1976) (片山卓也訳: アルゴリズム + データ構造 = プログラム, 科学技術出版社, 東京 (1979)).
- 7) 王 義孝: On the Efficiencies of Quicksort and Quickselection (Quicksort 及び Quickselection の評価), 山梨大学大学院工学研究科修士論文 (1977).
- 8) Sedgewick, R.: Quicksort, STAN-CS-75-492, pp. 1-24, Stanford University (1975).
- 9) Wainwright, R. L.: A Class of Sorting Algorithms Based on Quicksort, *Comm. ACM*, Vol. 28, No. 4, pp. 396-402 (1985).
- 10) Wainwright, R. L.: Technical Correspondence, *Comm. ACM*, Vol. 29, No. 4, pp. 331-335 (1986).

(昭和 62 年 10 月 8 日受付)

(昭和 63 年 9 月 5 日採録)



梶原 洋一 (正会員)

1963 年生。1986 年山梨大学工学部計算機科学科卒業。1988 年同大学院工学研究科修士課程修了。現在, 日本電気(株)に勤務。在学中の研究テーマは, アルゴリズムの解析と, オブジェクト指向言語システム。ソフトウェア工学全般に興味をもっている。



有澤 誠 (正会員)

1944 年生。1967 年東京大学工学部計数工学科卒業。電子技術総合研究所を経て, 現在山梨大学工学部計算機科学科に勤務。工学博士。ソフトウェア工学, 特にソフトウェアの評価, アルゴリズムの解析, オブジェクト指向システムなどに興味をもっている。