Vol. 21 No. 2 情報処理学会論文誌 Mar. 1980

ショートノート

安全クイックソート†

野崎昭弘†† 杉本俊彦†††

「内部ソートのおそらく最も有用な汎用技法」(クヌース) といわれるクイックソートの長所は平均所要時間が短いことであり，短所は最悪の場合の所要時間がひじょうに長い (項目数 n に対して $O(n^2)$) ことである．本論文ではクイックソートを改良して，最悪の場合の所要時間を項目数 n に対して $O(n \log n)$ におさえ，しかも平均所要時間をほとんど損なわないようにできることを示した．

1. はじめに

Hoare[1] の着想に始まるクイックソートの技法は，いろいろな改良案によって洗練され[2]，「内部ソートに対しておそらく最も有用な汎用技法」と見られている (Knuth[3], 5.5 SUMMARY, 7.)．しかしこの技法は，平均実行時間は少ないけれども，最悪の場合の実行時間が異常に大きいという欠点があり，そのため「実用的には使いにくい」という声もある．しかし簡単な工夫で最悪の場合の実行時間をおさえることができ，しかも我々の実験によれば「平均所要時間もそれほど損なわれない」ことがわかったので，報告したい．

2. クイックソートの概要

クイックソートの手順は次のように述べられる．

今，N個のデータ $X(1), \cdots, X(N)$ のソートを行いたいとする．

(1) "境界値" と呼ばれるある値 T を適当に定める．

(2) データを適当に並べかえて，
$X(1), \cdots, X(J)$ はどれも T 以下，
$X(J+1), \cdots, X(N)$ はどれも T 以上となるように "分割" する．

(3) 分割によって生じた左右の "ブロック" を個別にソートする．

それで全体のソートが完成するのは明らかであろう．

境界値Tの選びかたとしては，$X(N/2)$ とか，いくつかのサンプルの中央値[5] あるいは平均値 (実数値データの場合)[2] など，いろいろな方法が提案されている．

(3)では，ブロックのサイズが十分小さいときは直接挿入法 (straight insertion)[3] を使い，それ以外の場合は(1), (2)を再帰的に適用する．

この方法がうまく働くのは，境界値Tがデータ全体の中央値に一致する場合である．その場合分割のネスティング (**図 1**) の深さは高々 $\lceil \log_2 N \rceil$ である．ひとつのブロックを分割するには，境界値とそのブロックの要素のすべてとの比較が必要であるから，分割が1段深まるたびに，N回の比較が必要である．結局最も好適の場合には，全体として高々 $N \log N$ 回程度の比較ですむ．

逆に，**図 2** のように偏った分割が続くと，事情は一変する．分割の深さは約N段に達し，必要な比較回数は $O(N^2)$ になる．しかし一様乱数データに対しては平均比較回数は $O(N \log N)$ であり，所要時間を実測してみると Heap 法の約半分程度ですむ (王[2], TABLE IV および Knuth[3], 5.2.3., プログラムH

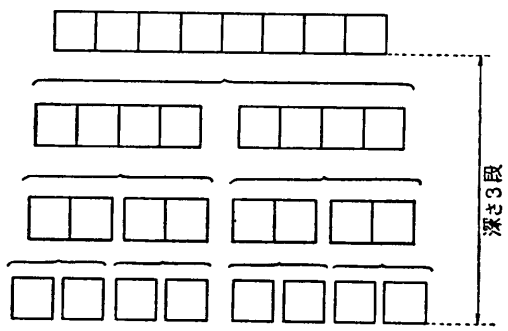

図 1 分割の深さ
Fig. 1 Depth of partitioning.

† Fail-safe Quicksort by AKIHIRO NOZAKI (Natural Science Division, International Christian University) and TOSHIHIKO SUGIMOTO (Department of Computer Science, Faculty of Engineering, Yamanashi University).

†† 国際基督教大学理学科
††† 山梨大学工学部計算機科学科

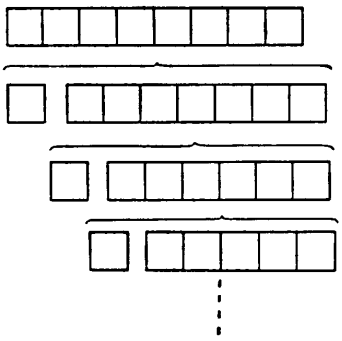

図 2 偏った分割
Fig. 2 Unbalanced partitioning.

の解説参照).

結局，最悪の場合の所要時間が $O(N^2)$ になるのがクイックソートの弱点で，これを $O(N \log N)$ におさえるのが我々の目標である.

3. 処理手順

我々の「安全クイックソート」の手順は次のとおりである.

(1) 準備：与えられたデータサイズ N から

$M=\lfloor \alpha \log_2 N \rfloor$

を計算しておく．M は分割の深さの許容限界で，係数 α は，経験的に 1.5 と定めてみた.

(2) 分割：ブロックのサイズが N_0 以上で，しかも“分割の深さ”が M 未満のときは，適当に境界値 T を定めて分割を行う．N_0 は経験的に 11 と定めた[2].

(3) 直接ソート：サイズが N_0 未満のブロックは，直接挿入法によってソートする．また分割の深さが M に達したときは，ブロックをそれ以上分割せず，Heap 法によってソートする.

分割によって生じた複数個のブロックの管理は，分割の深さをあらわすパラメータ IP と，スタック ISTACK (IP) とによって行う．すなわち，深さ IP のブロック X(L), ⋯, X(R) が

X(L), ⋯, X(J)

および

X(J+1), ⋯, X(R)

に分割されたときは，

```
IP=IP+1
ISTACK (IP)=R
R=J
```

のように右側のブロックの右端の添字をスタックに記録し，左側のブロックのソートを始める．そして左側のブロックが完了したとき，

```
L=R+1
R=ISTACK (IP)
```

によって右側のブロックのソートを始めることができる.

ここでただちに IP=IP−1 とはしない方がよい．Rはもはや不要の情報であるが，分割の深さは減っていない（右側のブロックの深さは左側のブロックの深さと同じ）からである．そのことは図2のような例を考えてみればすぐわかる．そこで我々はあるブロックのソートが終わったとき，次のような処置をとることにした.

```
90  IF (IP. EQ. 0)  RETURN
    IF (ISTACK (IP). NE. 0)
        GO TO 95
    IP=IP−1
    GO TO 90
95  L=R+1
    R=ISTACK (IP)
    ISTACK (IP)=0
    GO TO  (ブロック X(L), ⋯, X(R) のソートの
                入口)
```

ISTACK (IP)≠0 とは，左側のブロックのソートがすんで，右側のブロックのソートに移ることを意味する．そのときスタックの内容はクリアしておくが，分割の深さ IP は変えない．また ISTACK (IP)=0 とは，右側のブロックのソートもすんだこと，したがって深さ IP を1減らしてよいことを意味する.

我々のプログラムの最悪の場合の所要時間は，次のように見積もられる.

(1) 準備：$O(\log N)$

(2) 分割：$O(N \times M)=O(N \log N)$

(3) 直接ソート：直接にソートされる最終ブロックのサイズを $N_1, \cdots, N_t$ とすると，当然

$N_1+\cdots+N_t=N$.

第 i ブロックが直接挿入法によってソートされた場合，その所要時間は $O(N_i^2)$ であるが，$N_i \leqq N_0$ であるから，ある定数 c_1 について $c_1 N_i \log N_i$ でおさえられる．また第 i ブロックが Heap 法でソートされた場合の所要時間も，ある定数 c_2 について $c_2 N_i \log N_i$ でおさえられる．ゆえに $c=\mathrm{Max}\{c_1, c_2\}$ とおくと，(3)の所要時間は

$c\sum N_i \log N_i \leq c(\sum N_i) \log N = cN \log N$

でおさえられる.

このように各段階での所要時間が $O(N \log N)$ であ

表 1 一様乱数データに対する所要時間
Table 1 Required time for uniformly random data.

n	クイックソート	安全クイックソート	ヒープソート
500	0.189	0.190	0.380
1,500	0.656	0.659	1.345
2,500	1.160	1.163	2.419
5,000	2.519	2.533	5.244

一様乱数データに基く，20回の実験の平均（単位秒）．

表 2 タチの悪いデータに対する所要時間
Table 2 Required time for wicked data.

n	クイックソート	安全クイックソート	ヒープソート
100*	0.089	0.064	0.051
500**	0.799	0.426	0.302

* X(I)=4.0 **(50−I)
** X(I)=4.0 **(62−I/2)

るから，全体としても $O(N \log N)$ でおさえられる．

4. 実　験

一様乱数データに対する所要時間の実測値を**表1**に，特にタチの悪いデータ（**表2**の注参照）に対する所要時間の実測値を表2に示した．

注意　このデータが『タチが悪い』というのは，境界値のきめ方に依存している．

これらの表からわかるように，平均所要時間はほとんど損なわれず，タチの悪い場合の所要時間が著しく短縮されている．

我々の計算機の記憶容量および指数部分の桁数の関係から，あまり大きなサイズのデータに対する実験はできなかった．また定数 α, N_0 をどのように選べば最適になるかも，計算機によるであろうが，今後の課題である．

参 考 文 献

1) Hoare, C. A. R.: Algorithm 64 QUICKSORT, CACM, Vol. 4, p. 321 (1961).
2) Wang, Y.: On the Efficiencies of Quicksort and Quickselection, 山梨大学工学研究科修士論文 (1978).
3) Knuth, D. E.: The Art of Computer Programming, Vol. 3, Addison-Wesley (1973).
4) van Emden, M. H.: Increasing the Efficiency of Quicksort, CACM, Vol. 13, pp. 563-567 (1970).
5) Hoare, C. A. R.: Quicksort, Computer J., Vol. 5, pp. 10-15 (1962).

付　録

```
0001            SUBROUTINE FSWS(X,LEFT,RIGHT)
0002            DIMENSION X(5000),ISTACK(20)
0003            DOUBLE PRECISION A,XX,T
0004            INTEGER RIGHT,R
      C FAILSAFE QUICKSORT,3=SAMPLE MEAN METHOD
0005            L=LEFT
0006            R=RIGHT
      C SET A LIMIT TO THE STACK DEPTH
0007            IPMAX=4
0008            NMAX=16
0009            N=R-L+1
0010          2 IF(N.LE.NMAX) GO TO 5
0011            IPMAX=IPMAX+1
0012            NMAX=NMAX+NMAX
0013            GO TO 2
0014          5 IPMAX=3*IPMAX/2
      C NECESSITY OF PARTITIONING
0015            IP=0
0016         10 IF(R-L.LE.10) GO TO 100
0017            IF(IP.EQ.IPMAX) GO TO 200
      C PARTITIONING
0018            M=(L+R)/2
0019            T=(X(L)+X(M)+X(R))/3.0
0020            I=L+1
0021            J=R-1
0022         40 I=I+1
0023            IF(X(I).LT.T) GO TO 40
0024         50 J=J-1
0025            IF(X(J).GT.T) GO TO 50
0026            IF(I-J) 60,65,70
0027         60 XX=X(I)
0028            X(I)=X(J)
0029            X(J)=XX
0030            GO TO 40
0031         65 J=J-1
      C STACKING
0032         70 IP=IP+1
0033            ISTACK(IP)=R
0034            R=J
0035            GO TO 10
      C AFTER SORTING A BLOCK
0036         90 IF(IP.EQ.0) RETURN
0037            IF(ISTACK(IP).NE.0) GO TO 95
0038            IP=IP-1
0039            GO TO 90
0040         95 L=R+1
0041            R=ISTACK(IP)
0042            ISTACK(IP)=0
0043            GO TO 10
      C STRAIGHT INSERTION
0044        100 I=L
0045            J=L-1
0046        110 I=I+1
0047            IF(R.LE.I) GO TO 90
0048            IF(X(I).LE.X(I+1)) GO TO 110
0049            T=X(I+1)
0050            J=I
0051        120 X(J+1)=X(J)
0052            J=J-1
0053            IF(J.LT.1) GO TO 130
0054            IF(T.LT.X(J)) GO TO 120
0055        130 X(J+1)=T
0056            GO TO 110
      C HEAP SORT
0057        200 IL=(R-L+1)/2+L
0058            IR=R
0059        210 IF(IL.GT.L) GO TO 215
0060            T=X(IR)
0061            X(IR)=X(L)
0062            IR=IR-1
0063            IF(IR.GT.L) GO TO 220
0064            X(L)=T
0065            GO TO 90
0066        215 IL=IL-1
0067            T=X(IL)
0068        220 J=IL
0069        225 I=J
0070            J=(J-L)*2+L+1
0071            IF(J-IR)230,235,245
0072        230 IF(X(J).LT.X(J+1)) J=J+1
0073        235 IF(T.GE.X(J)) GO TO 245
0074        240 X(I)=X(J)
0075            GO TO 225
0076        245 X(I)=T
0077            GO TO 210
0078            END
```

(昭和54年 6 月 18 日受付)
(昭和54年 8 月 23 日採録)