葉数適応整列法LOASの最適性

4Ｊ－8　二村良彦　早稲田大学 理工学部　二村夏彦　シラキュース大学

1.はじめに

現実の問題として数列のソーティングを扱う場合には、与えられた数列が殆ど整列済みである場合が多いと言われている[7]。我々は整列済みに近い数列ほど速く整列するソーティングアルゴリズムLOAS(Leaves-Optimal-Adaptive-Sort)を文献[3]で発表した。そこでは数列の整列度を計る為に葉数と呼ばれる事前整列性測度を導入し、その測度に関して数列の整列度が良い時には、LOASが既存のQUICKソートやMERGEソート等よりも高性能であることを示した。またより詳しい性能比較評価、およびLOASの実現方式の詳細については本大会の別講演[4,5]で報告する。本稿ではLOASが葉数に関して最適なアルゴリズムであることを詳しく論じる。

以下では長さnの順列のみを扱う。数列Xの長さおよび集合Sの濃度を各々|X|および‖S‖で表す。対数の底は2、即ちlog n=$\log_2$ nとする。merge(X,Y)は2つの整列済み数列XとYをマージする関数である。例えばX=<1,3,5>かつY=<2,4,6>ならばmerge(X,Y)=<1,2,3,4,5,6>である。適応整列法に関する既存の表記法および概念については[6]に従う。

2.数列の葉数

数列の葉数は数列において隣接する要素よりも小さい要素(葉)の個数であり正確にはLeaves(X)と書く[3]。例えばX1=<7,1,2,6,5,3,4>については、Leaves(X1)=2。何故ならば1と3だけが隣接する要素よりも小さいからである。例えば2は右の隣人より小さいが左の隣人より大きいので葉ではない。葉数を他の事前整列性測度[6]と比べると、例えば、RunsやInv+1以下でありEnc/2以上である(ただしEncは常にLeavesより小さいとは限らない。例えば前述のX1についてはLeaves=2に対してEnc=4)[3]。

一般に事前整列性測度およびその測度で計られたXの整列性を各々PMおよびPM(X)とすると、実用的観点からPMは次の2要件を満足することが望ましい：(1)PM(X)はO(n)で計算できること。(2)PM(X)はn以下であること。

葉数が上記2つの要件を満足する測度であることは明らかである。一方Encの計算にはO(n*log Enc)要す[9]。しかもEnc最適なMELソートと比べて同等以上の性能をLOASが既に達成している[5]ので、Enc≦2Leavesであるにもかかわらず我々はLeavesの実用的価値は大きいと判断する。

3.葉数最適整列法LOAS

LOASは基本的には[6]のマージに基づく整列法の特殊な場合である[3]。それは前述の数列X1を次のように整列する。(1)X1を先ず左からスキャンし下降列と上昇列のペア列(<7,1>, <2,6>) (<5,3>,<4>)に分ける。(2)次に各ペアをマージしLeaves(X1)個の上昇列を作る、即ちmerge(<7,1>,<2,6>)かつmerge(<5,3>,<4>)。(3) 最後に上で得られた2つの数列をマージする、即ちmerge(<1,2,6,7>, <3,4,5>)。これがO(n log m) のアルゴリズムであることは自明である。またメモリー所要量は高々葉数個の部分列を管理するために[(n+1)/2]、そしてマージのためのコピー領域に[n/2]の合計nである。LOASは、数列に関する下記の2つの性質に基づき基本アルゴリズムにおける分割法とマージ法を改良したものである。

性質1： 下り列1、上り列1、下り列2、上り列2、...の

Optimality of Leaves-Optimal-Adaptive-Sort (LOAS)
Yoshihiko Futamura+ and Natsuhiko Futamura*
+School of Science and Engineering, Waseda University
*School of Computer and Information Science, Syracuse University

ように数列を分割した時merge(下り列i, 上り列i)の最小値は下り列iの最後の要素である。

性質2：merge(下り列i, 上り列i)の最後の要素はmerge(下り列i+1, 上り列i+1)の先頭の要素よりも大きい。またこのようなマージの過程で作られる隣接する2つの列に於いても、左列の最後の要素は右列の先頭の要素よりも大きい。

4.LOASの最適性

MANNILA[8]の意味において葉数について最適な整列法の計算量はO(n log Leaves(X))であることの証明の概略を示す。

定義[8]：ある非負の整数値関数M(事前整列性測度と呼ぶ)に対して、整列法が長さnの数列Xを$O(\max\{n, \log \|below(X,M)\|\})$のキー比較で整列できる時に限り、その整列法はM最適であると言う。ただし、$below(X,M)=\{Y \mid M(Y)\leqq M(X)$かつ$|Y|=|X|\}$。

従って葉数に関するLOASの最適性を言うためには次の定理を証明すればよい。

定理：$m^n \leqq \|below(X,Leaves)\|$

この定理の証明には$L(n,m)\geqq m^n-(m-1)^n$を使う。ただしL(n,m)は節数n、葉数mの順列の個数であり、下記の再帰方程式で定義できる[1,2]：

(1) $L(n,m)=0$ if $m<1$ or $n+1<2m$

(2) $L(n,1)=2^{n-1}$

(3) $L(n,m)=2m*L(n-1,m)+(n-2m+2)*L(n-1,m-1)$ if $1<m\leqq(n+1)/2$

ちなみに、節数nの順列が持つ平均葉数は(n+1)/3である。また1とxの間の一様乱数列で長さがnのものにおける平均葉数は$(n+1)/3-(n-2)/(3x^2)$である。ただし等しい値が並んだ時は右側が大きいものとする[1,2]。

LOAS以外の葉数最適整列法としては下記の2つを我々は知っている。(1)Enc≦2Leavesであるので、MELソートは葉数最適である。ただし[5]に示す通りキーの比較回数に関してはLOASより劣っている。(2)対称整列法[2]はO(n*Leaves)のアルゴリズムであるので、Leaves≦log nのとき対称整列法は葉数最適である。対称整列法の抜き出し操作において、勝ちヒープが子供を2人持つ際に、その2人の子供と、負けヒープの3者の中の大小関係をヒープにすることにより、対称整列法の高性能化が図れることを我々は確認している。

5.おわりに

葉数適応整列法LOAS、MELソートおよび対称整列法の葉数に関する最適性を示した。葉数は実用的に有用な事前整列性測度と考えられるのでこの成果も実用的と考えられる。実際、葉数が少ない場合にはLOASはQUICKソートやMERGEソート等の実用的整列法より高性能であることを実験的に確認している[3,5]。一方、現実の世界で大量にソートされているデータの葉数は未調査であるので、それを調べることが今後の課題である。

6.参考文献

[1] 二村、筧、二村：対称ヒープの実現とその応用、情報処理学会アルゴリズム研究会28-7、92年7月. [2]二村、二村、筧：対称整列法、情報処理学会アルゴリズム研究会28-8、92年7月. [3]二村、二村、遠藤、平井：LOAS:葉数に関して最適な適応整列法、日本ソフトウェア科学会11回大会C1-1、1994. [4]二村、平井：葉数最適整列法LOASの実現法、情報処理学会第50回大会4J-9、1995. [5]二村、遠藤、平井、青木：適応整列法の評価、情報処理学会第50回大会4J-10、1995. [6] ESTIVILL-CASTRO AND WOOD:A survey of adaptive sorting algorithms, ACM Computing Surveys, Vol. 24, No.4, December, 1992,441-476. [7]KNUTH：The Art of Computer Programming. Vol.3. Addison-Wesley, Reading, Mass, 1973. [8] MANNILA: Measures of presortedness and optimal sorting algorithms. IEEE Trans.Compt.C-34, 1985, 318-325. [9]SKIENA：Encroaching lists as a measure of presortedness, BIT 28,1992, 755-784.