基于链表的同构化首尾排序新算法
——“程序设计”与“数据结构”课程的融合创新案例研究

周启海
（西南财经大学信息技术应用研究所　成都 610074）（西南财经大学经济信息工程学院　成都 610074）

摘　要　依据同构化基本原理，研究和发现了基于传统内部首尾排序算法的同构化特点与本质；进而，利用其同构化特点与本质，提出了几种基于链表的首尾排序新算法；从而，进一步深化和推广了首尾排序算法的应用方式与实用范围；同时，也为“程序设计”、“数据结构”的课程融合、教学改革、教育创新提供了重要研究案例。

关键词　同构化，链表，首尾排序，算法，课程融合

New Algorithms for Sorting from Head and Rear Based on Chain List
—— Fused and Innovated Case Study of “Program Design” and “Data Structure” Subjects

Zhou Qi-hai
(Research Institute of Information Technology Application, Southwestern University of Finance and Economics, Chengdu 610074, China)
(School of Economic Information Engineering, Southwestern University of Finance and Economics, Chengdu 610074, China)

Abstract　According to isomorphic fundamental principles, the isomorphic characters and natures of algorithms for sorting from head and rear based on traditional internal sorting were studied and found. Further, several new algorithms for sorting from head and rear based on chain lists were advanced by using their isomorphic characters and natures. Therefore, the application forms and applied ranges of algorithms for sorting from head and rear were deepened and expended. Provided an important studying case for fusing subjects, teaching reform and innovating education of “Program Design” and “Data Structure”.

Keywords　Isomorphic, Chain list, Sorting from head and rear, Algorithm, Fusing subjects

1　引言

排序（即按给定判据使所论数据对象集各元素有序化），分为基于内存的内（部）排序，与面向外存的外（部）排序，且内排序常是外排序的基础。

排序问题，历来是计算机科学中重要基础问题，是教学研究（例如，它可作“数据结构”、“程序设计”的课程融合、教学改革、教育创新的重要研究内容之一）、算法设计、系统开发与应用实践，既密切相关，又屡见不鲜的重要基本内容[1-4]。数组是人们最常见、最常用、最习惯的内排序工具。令人惊叹的是，它使许多人长期来误认为数组似为唯一好用的内排序工具。但对那些采用链表作为主体数据结构的理论研究课题与实际应用问题，而所论课题性质与问题场景又涉及其所论数据集的频繁增删、动态排序、顺序查找与随机检索等基本运算时，倘若再机械地沿袭通常基于数组的内排序方法，则势必将因其“作为目标数据处理场的链表与作为中间数据处理场的数组之间，必定存在且不可避免的数据对倒”所付出的大量时间开销，而严重降低解决所论课题与问题相关处理的算法效率。此时，别开生面、新颖明智地采用的基于链表的排序算法（例如：本文所给出的基于链表的首尾排序算法，或者文献[5]所给出的基于链表的择换排序算法），无疑是有效提高解决所论课题与问题相关处理算法效率的可行之路。

限于篇幅，本文在此仅以递增有序化输出任给 $n(<100)$ 个自然数为例，阐明基于链表的首尾排序新算法，以扩展和深化首尾排序算法的实用范围与应用方式，并兼作“数据结构”与“程序设计”课程的课程融合、教学改革、教育创新案例研究。

2　基于数组的择换排序算法

把找最大数、最小数的算法思想分别应用于排序，可产生简洁明快、效率较高的择换排序算法；而把找最大数、最小数的算法思想同时应用于排序，将催生巧妙简明、效率更高的首尾排序算法[3]。

事实上，采用数组数据结构的首尾排序算法，基于、巧于、高于择换排序算法。它的处理特点是：

(1)其处理数据集的当前处理范围，是分别从其首、尾两端向其对称中心元，同步、相向、同时步进式收缩。

(2)首尾排序算法在其第 $i(=1,2,\cdots,[n/2])$ 遍齐头并进的首、尾双向择换处理中，把从当前处理数据集中同时所获得的第 i 小者、第 i 大者，分别使之到所论数组的对称位置（即第 i、第 $n-i+1$ 位），则可产生排序效率比文献[5]择换法提高近一倍的首尾法。

(3)应特别注意“使第 i 小者、第 i 大者分别到其位操作”的新特点——如果第 i 大者恰在到其位前的第 i 小者位置，就必须“首先，使第 i 小者到其位；然后，重新标记第 i 大者新的所在位置；最后，才能使第 i 大者到其位（即：新的所在位置）”；否则，就会出现“错到其位”的误操作。

据此，首尾排序算法可同构化描述表征如下：

周启海　教授，博（硕）士生导师，主要研究方向为计算几何、算法研究与应用、财经计算、同构化信息处理等。

算法 Eg01 {基于数组的首尾排序算法}

```
{{i,j,k,Min,Max,n,p1,p2:整型;a:整型 数组 [1..100]}};{全局变量定义}
>>> {算法开始}
@:\\ 输出 "输入数据个数:";输入 n //
  直到 0<n 且 n<100;{容错输入}
对 i←1,TRUNC(n/2+0.5) @:\\ {排序遍数控制}
  Min←a[i];p1←i;{标记初始第 i 小者}
  Max←Min;p2←p1;k←n-i+1;{标记初始第 i 大者}
  对 j←i+1,p2 @:{比较次数控制}
    如果 Min>a[j]{当前数据较小?}
      T:\\ p1←j;Min←a[j] //{标记当前最小者}
      F:如果 Max<a[j] {当前数据较大?}
          T:\\ p2←j;Max←a[j] //;{标记当前最大者}
  如果 p1<>i {第 i 小者不在其位?}
    T:\\ a[p1]←a[i];a[i]←Min;{使第 i 小者到其位}
          如果 a[p1]=Max {交换后为第 i 大者? (注:此时第 i 大者,已非 a[p2])}
            T:p2←p1 {标记第 i 大者新位(警告:须使第 i 大者,又是 a[p2])}
          //;
  如果 p2<>k {第 i 大者不在其位?}
    T:\\ a[p2]←a[k];a[k]←Max //;{使第 i 大者到其位}
对 i←1,n-1 @:{数据个数控制}
  输出 a[i];{递增输出各有序化数据}
行输出 a[n];
!!!{算法结束}
```

3 首尾排序算法的根本特点与同构本质

本文作者依据自己提出的同构化基本原理[3],研究并发现了择换排序算法的根本特点与同构本质如下:

①所论数据最小者、最大者的比较择选,只利用了存储这些数据的变量本身(此处为下标变量),而与其所在位置无关(即下标);

②当前最小者、最大者应在位置,则只利用了标记这些最小者、最大者原位置的变量本身(此处为标记其下标位置的标记变量),而与其所具数值无关;

③把当前最小者、最大者分别交换到其应在位置,可不必打乱原变量序列的顺序;

④必须有效解决"占位冲突"问题:在递增有序化中,当"使第 i 小者到其位"的交换处理后,为确保"使第 i 大者到其位",必须切实解决交换处理中可能常的"大(者)占小(者)位(置)"的占位冲突问题;而在递减有序化中,当"使第 i 大者到其位"的交换处理后,为确保"使第 i 小者到其位",必须切实解决交换处理中可能常的"小(者)占大(者)位(置)"的占位冲突问题。

据此,可得出结论:只要能分别同构化地正确处理上述 4 个基本操作(即:当前最小者、最大者挑选;当前最小者、最大者位置标记;当前最小者、最大者分别交换到位;有效解决"占位冲突"),就并非一定要采用数组来实现首尾排序算法。事实上,链表与数组同为线性表的同构化本质属性,就从根本上决定了链表必定应该、而且可以同样用来解决排序问题。

应当指出:

(1)因基于链表的首尾排序处理同构化主要特性,是基于"首→中←尾"模式的双向、同时、同步有序化处理,故其首选数据结构理所当然应为双向链表,方可有利于对"首→中"方向与"中←尾"方向的双向有序化。

(2)原采用数组作数据结构的首尾排序处理时,其数据范围对称中心可用数组下标位置——"首、尾下标值和之半"来直接确定。但这种数据范围对称中心的直接算术计算定位法,显然已不能简单移植到采用链表作数据结构时的首尾排序数据处理(因为此时已无法通过直接算术计算来确定数据范围对称中心的位置),而必须另行采用跟踪标记当前数据处理范围的"首元指针"与"尾元指针"的动态取值情况来描述。

因此,若采用双向链表作为数据结构描述工具,并注意其数据范围对称中心确定处理的特定需要与特殊处理,则文献[5]所论"基于链表的结点插删、结点标记、(结点的数据字段)变量标记的 3 种同构化择换排序新算法",显然应该而且可以对应地推广为基于双向链表的结点插删、结点标记、(结点的数据字段)变量标记的 3 种同构化首尾排序新算法。

4 基于变量标记的双向链表同构化首尾新算法

为省篇幅,基于双向链表的结点插删同构化首尾排序新算法 Eg01、基于双向链表结点标记的同构化首尾排序新算法 Eg02 均予略(事实上,只要借鉴文献[5]及本文算法 Eg03,就已不难给出之),而仅给出其构思最巧、方法最简、效率最高的基于双向链表的(结点的数据字段)变量标记的同构化首尾排序新算法 Eg03。

算法 Eg03 {基于字段变量标记的双向链表首尾排序算法}

```
[[Link<==>指针 ^Nodes]];{双向链表的指针类型定义}
[[Nodes<==>记录 [ Data:整型;{元素体定义:自身数据字段}
                  Precede,Succeed:Link ]{邻元体定义:(直接)前趋、后继指针字段}
                  ]];{双向链表的指针元素类型定义}
{{i,j,Min,Max,n:整型}};{整形变量定义}
{{Head,Rear,Rear1,Visit1,Visit2,Place1,Place2:Link}};{定义链表头、尾、访问、位置等}
>>> {算法开始}
@:\\ 输出 "请输入数据个数:n=?";输入 n {提示下输入数据个数}
//
直到 n>=2;{直到个数 n 已合理且中意为止}
{"用单向链表存放初始数据序列"处理}
生成结点 Head;{生成头结点,以存放第 1 个数据}
输入 Head.Data;{存放第一个节点数据}
Head.Precede←Null;{使头结点前趋指针为空指针}
Head.Succeed←Null;{使头结点后继指针为空指针}
Rear←Head;{头结点此时也是尾结点}
Visit1←Head;{标记初始链表头}
i←2;{结点序号计数器初值}
当 i<=n @:\\ {当链表尚有未生成结点时}
生成结点 Visit1;{生成新尾结点}
输出 "请输入第个 i 数据个数";输入 Visit1.Data;{新尾结点加载数据}
Visit1.Precede←Rear;{使新尾结点前趋指针连通原尾结点}
Visit1.Succeed←Null;{使新尾结点后继指针为空指针}
Rear.Succeed←Visit1;{使原尾结点后继指针连通新尾结点}
Rear←Visit1;{标记新尾结点}
i←i+1 {下一新结点序号生成}
//;
{"用双向链表对初始数据序列进行排序"处理}
Visit1←Head;{排序遍数与始端(结点)控制指针}
```

```
Rear1←Rear;{排序尾端(结点)控制指针}
当 Visit1<>Rear1 @:\\{当排序遍数未完时}
Place1←Visit1;{标记当前范围最小者初始处}
Min←Visit1.Data;{标记当前范围最小者初始值}
Max←Min;{最小者初始值兼作最大者初始值}
Place2←Visit1;{最小者初始处兼作最大者初始处}
Visit2←Visit1.Succeed;{从下一结点起,择选当前最小、最大者}
当 Visit2<>Rear1.Succeed @:\\{当尚有未选结点时}
如果 Visit2.Data<Min{当前结点数据更小吗?}
T:\\ Min←Visit2.Data;Place1←Visit2 //{标记当前最小者及其位置}
F:如果 Visit2.Data>Max{当前结点数据更大吗?}
T:\\ Max←Visit2.Data;Place2←Visit2 //{标记当前最大者及其位置}
Visit2←Visit2.Succeed{指向下一待访结点}
//
如果 Place1<>Visit1{当前最小者不在其位吗?}
T:\\ Place1.Data←Visit1.Data;Visit1.Data←Min //{使当前最小者直接到其位}
如果 Place1.Data=Max{当前最大者恰在交换后处吗?(当前最大者已不在 Place2 处)}
T:Place2←Place1{标记第 i 大者新位(警告:必须使当前最大者又在 Place2 处)}
//;
如果 Place2<>Rear1{当前最大者不在其位吗?}
T:\\ Place2.Data←Rear1.Data;Rear1.Data←Max //{使当前最大者直接到其位}
Visit1←Visit1.Succeed{从头向中(心)步进:指向下一遍排序范围的始端结点}
如果 Visit1<>Rear1{当前排序范围内并非只剩最后两个结点吗?}
Rear1←Rear1.Precede{从尾向中(心)步进:指向下一遍排序范围的尾端结点}
//;
{"输出双向链表中递增有序化各数据"的处理}
Visit1←Head;{指针 Visit1 标记双向链表头结点}
当 Visit1<>Null @:\\{当访问指针 Visit1 非空指针时}
输出 Visit1.Data;{输出当前结点数据字段}
Visit1←Visit1.Succeed{从头向尾步进:指向下一结点位置}
//;
行输出;
!!!
```

结束语 依据同构化基本定理。本文研究和发现了基于传统内部首尾排序算法的同构化特点与本质;利用其同构化特点与本质,逐个更优地构造了基于链表的结点插删、结点标记、(结点数据字段)变量标记的 3 种首尾换排序新算法,有利于进一步深化和推广首尾排序算法的应用方式与实用范围。并且,这些基于链表的首尾排序算法及其编程实现,完全可作"程序设计"与"数据结构"的课程融合、教学改革、教育创新的一个有效案例,而这已被由作者成功主持与主研的 2006 年四川省级精品课程"数据结构"的教学研究与教改实践所证实。

参 考 文 献

[1] Knuth D E. The Art of Computer Programming, Volume 1, Fundamental Algorithms. Addison-Wesley Publishing Company, Inc., 1973

[2] Knuth D E. The Art of Computer Programming, Volume 3, Sorting and Searching. Addison-Wesley Publishing Company, Inc., 1973

[3] 周启海. C++同构化对象程序设计原理. 北京:清华大学出版社,北方交通大学出版社,2004

[4] 严蔚敏. 数据结构(C 语言版). 北京:清华大学出版社,2002

[5] 黄涛. 基于链表的择换排序新算法——"数据结构"与"程序设计"课程的融合创新案例研究[J]. 计算机科学,2008,35(11)

[6] 周启海. C 语言程序设计教程. 北京:机械工业出版社,2004

[7] 周启海,李朔枫,杨祥茂,等. 论程序设计课程教学中的同构化创新思想教育——"对→好→巧→妙→绝"的算法案例[J]. 计算机科学,2007,35(5)

(上接第 163 页)

SV 技术和数据划分技术生成两种形状的初始网格,用随机映射技术定义网格间的距离。实验表明所设计的网格距离定义具有较好的数据特征捕捉能力,ACGD 算法具有较好的聚类性能。

但是合并过程的控制仍是待解决的问题。在一个庞大的合并路径图上通过分析得出合理的簇划分的结果仍是较困难的任务,这也降低了聚类效率。如何评价合并状态的优劣来适时地终止合并操作是下一步的工作方向。

参 考 文 献

[1] Bradley P S, Fayyad U M. Refining Initial Points for k-Means Clustering // Proceedings of 15th International Conference on Machine Learning. San Francisco, USA, 1998:91-99

[2] Lance G N, Williams W T. A general theory of classificatory sorting strategies . 1. Hierarchical systems. The Computer Journal, 1967, 9(4): 373-380

[3] Dan K, Sepandar D K, Christopher D. Manning from Instance-level Constraints to Space-level Constraints: Making the Most of Prior Knowledge in Data Clustering // Morgan K, ed. Proceedings of the Nineteenth International Conference on Machine Learning. San Francisco, USA, 2002:307-314

[4] Ben-Hur A, Horn D, Siegelmann H T. Support Vector Clustering. Journal of Machine Learning Research, 2001, 2:125-137

[5] 丁世飞,史忠植,靳奉祥,等. 基于广义信息距离的直接聚类算法. 计算机研究与发展, 2007(4):674-679

[6] Novak E, Ritter K. The Curse of Dimension and a Universal Method for Numerical Integration // Nürnberger G, et al., eds. Multivariate Approximation and Splines. Germany, 1997:177-188

[7] http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html

[8] Ng A, Jordan M, Weiss Y. On Spectral Clustering: Analysis and Algorithm // Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2002

[9] http://www.uncc.edu/knowledgediscovery