静态链表上排序算法的研究

达文姣，任志国，王龙平
（兰州城市学院信息工程学院　甘肃兰州，730070）

摘　要： 排序是计算机操作中的一种常用技术，排序算法在顺序表上有很多实现技术，但在静态链表上的研究却很少见。本文讨论了静态链表上冒泡排序，插入排序和选择排序算法的实现思想，用高级语言实现了这几种算法，最后分析了这些算法的性能。

关键词： 静态链表；冒泡排序；插入排序；选择排序；算法分析

Abstract: Sort is an important technology in computer science. The sort algorithms on sequence list is often discussed, but few on static chain list. The bubble sort algorithm, insertion sort algorithm and choose sort algorithm is disassed in this paper on static chain list. These algorithms is realized on computer by programming, and last, The preformances of these algorithms are analyzed.

Key words: Static chain table ; Bubble sort algorithm ; Insertion sort algorithm ; Choose sort algorithm ; Analysis of algorithms

中图分类号： TP311.12　　**文献标识码：** A　　**文章编号：** 1001-9227(2011)02-0012-03

0　引　言

排序是计算机程序设计中一项经常而又重要的操作，研究排序算法具有重要的理论意义和广泛的应用价值。而对于静态链表上排序算法的研究和实现更加有助于我们掌握静态链表的实质和排序算法的广泛性。所以本文将讨论插入、冒泡、选择三种内部排序算法在静态链表上的实现过程。文中没有提及的概念及术语参见文献[1-3]。

1　冒泡排序算法在静态链表上的实现

定义如下静态链表：

```
# define N 50   // N为链表可能达到的最大长度
typedef int elemtype;
typedef struct node
{ elemtype data; // 数据域
  int next;
}slinklist;
```

在众多的排序方法中，冒泡排序(bubble sort)是最简单的一种排序算法。在顺序表上冒泡排序的实现非常简单，在静态链表上的排序思想也一样。算法思想描述如下：在静态单链表中，也可以进行冒泡排序操作，对于静态链表每一个结点看成竖着排列的“气泡”，然后分别从静态链表的头结点开始向尾节点扫描。在扫描的过程中时刻注意两个相邻元素的顺序，保证前一结点元素的数据域小于后一节点元素的数据域，这样经过一趟扫描后就使较大的元素沉到链尾，然后记住链尾结点的位置。下次又从头结点向后扫描，由于在前一趟扫描过程中最大的元素已经沉到链尾并记住了该元素的位置，所以这次扫描最大的元素不再参加排序，将剩下的元素进行排序，排序的过程中保证使得后一结点元素的数据域大于前一结点元素的数据域。这样反复的扫描，并不断缩小排序空间，直到整个序列有序位置为止。这样看来，在排序中，只需记住最后排好序的元素的位置即可。具体算法设计如下：

具体算法描述如下：

```
void Bubblesort(slinklist s[N])//静态链表上的冒泡排序算法
{ int head=-1,end,p,q;//end用来记录排好序的最后一个元素位置的地址，p，q 分别为前序，后继的关系
   elemtype temp;
   p=s[head].next;
   q=s[p].next;
   end=-1;//排序前end指向链尾NULL
   while(end!=s[head].next) // 如果head所指结点的next成员为end，则结束循环
   {   p=s[head].next;//p结点总是从链表的头结点开始
       q=s[p].next;// q总是指向"p所指结点"的下一结点
       while(s[p].next!=end)// 当s[p].next的值为end时，表示已到链尾
       {   if(s[p].data>s[q].data)//按照数据域从小到大排序
           {   temp=s[p].data;
               s[p].data=s[q].data;
               s[q].data=temp;
           }
           p=q;
           q=s[q].next;
       }
       end=p;// 使end指向"每次排序后的q所指的结点即尾结点"
   }
}
```

收稿日期：2010-12-10

作者简介：达文姣(1976-)，女，讲师，主要从事计算机应用技术的研究与教学工作。

2 插入排序算法在静态链表上的实现

插入排序就是指在要排序的一组数中，假设前面(n-1)[n>=2]个数已经是排好顺序的，现在要把第n个数插到前面的有序数中，使得这n个数也是排好顺序的。如此反复循环，直到全部排好顺序。直接插入排序是稳定的。而在链表中，也可以这样认为，链的前面n-1结点已经有序，现要在把第n个结点插到前面的有链表中，使得这n个结点也是排好顺序的。如此反复循环，直到全部排好顺序。这样，就实现了链表上的插入排序。

定义结点如上文，算法思想描述：在静态单链表中，进行排序操作也可用插入排序，具体操作为：先将原静态链表中的第一个元素不变，然后将其它的所有元素放入备用静态链表中的，接着取备用链表中的第一个元素与原静态链表中的第一元素比较，插入到原静态链表中的，继续从备用静态链表中取一个元素，让它跟原静态链表中的第一个元素作比较找到相应位置插入即完成排序操作，以后同理。

具体算法描述如下：

```
void sortlinklist(slinklist s[N])//静态链表上的插入排序算法
{ int  pre,p,q,newhead,head=-1;//定义变量
  p=s[head].next; //变量p用来存放链L1的头结点地址
  newhead=s[p].next; //newhead用来标记链L2的头结点的地址
  s[p].next=-1;//已用静态链表的尾结点指针指向-1表示结束
  while(newhead!=-1) //判断原链中的元素，如果不少于两个则进行排序操作
  {   q=newhead;
      newhead=s[newhead].next;
      pre=head; // 已用静态链表的前驱
      p=s[head].next; // 已用静态链表的后继
      while(p!=-1&&s[p].data<s[q].data) //已用链表和备用链表中的元素作比较
      {   pre=p;
          p=s[p].next; //依次向后找合适的插入点
      }
      s[pre].next=q;
      s[q].next=p; //将元素链入已用静态链表的中
  }
}
```

3 选择排序在静态链表上的实现

选择排序在单链表上的基本思想和线性表上是一样，都是从记录的无序子序列中“选择”关键字最小或最大的记录，并将它加入到有序子序列中，以此方法增加记录的有序子序列的长度。具体为n个记录的全排序需进行n-1轮次；每轮次，在剩余的记录中直接找出最小（大）的记录，交换到相应位置，重复直至全部排序完毕。

算法思想描述：在静态单链表中，进行选择排序，具体操作为：先将原静态链表中的第一个元素不变，然后将其它的所有元素放入备用静态链表中的，接着取备用链表中的第一个元素与原静态链表中的第一元素比较，插入到原静态链表中的，继续从备用静态链表中取一个元素，让它跟原静态链表中的第一个元素作比较找到相应位置插入即完成排序操作，以后同理。

具体算法描述一下如下：

```
void Selectsort(slinklist s[N])//静态链表上的选择排序算法
{ intp,min,q;//min存放最小元素的地址；p,q分别为前序，后继的关系
  elemtypet;
  for(p=s[head].next;s[p].next!=-1;p=s[p].next)//p从链头遍历到链尾
  {   for(min=p,q=s[p].next;q!=-1;q=s[q].next)//p,q始终是前序和后继关系
  {
  if(s[q].data<s[min].data)//记住最小元素的位置
                  min=q;
      }
      if(min!=p)//将最小元素交换到链的最前面完成选择排序
      {t=s[q].data;
      s[q].data=s[min].data;
      s[min].data=t;
      }
  }
}
```

4 各种排序算法在顺序表和链表上性能比较

在线性表上的冒泡排序，插入排序，选择排序，最好情况下的时间复杂度都是O(n)，最差和平均情况下的时间复杂度是O(n2)，辅助空间为O(1)，算法一般不稳定。在静态链表的冒泡排序，插入排序，选择排序的时间复杂度，空间复杂度，稳定性与线性表上完全相同。所以从实现过程和算法的分析，可以很明显的发现两种算法的设计思想和实现过程基本相同。而且实现过程也非常相似。

5 总 结

通过以上几种排序算法在链表上的实现，让我们了解到排序算法实现的广泛性，不仅仅在顺序表上能实现，而且在链表上同样也能实现。但与顺序表上的排序算法相比，链式结构上的排序有其明显的优越性：首先链式结构上的排序只改变链的指向，而不会改变数据元素所占结点的位置，即不会出现大量数据元素的移动。其次链式结构上每个数据元素占用一个结点，而不会有多余的结点存在，所以数据所占的存储空间不会浪费。静态链表上排序算法的时间开销比顺序表上的更少。

参考文献

[1] 严蔚敏，吴伟民. 数据结构[M]. 北京:清华大学出版社, 1997, 10.

[2] 耿国华. 数据结构(C语言版)[M]. 西安: 西安电子科技大学出版社, 2002.

[3] Robert L. Kruse, Alexander J. Ryba. Data Structures and Program Design in C++[M]. Pearson Education, USA, 2001.5

[4] Shell D L. A high- speed sorting procedure[J]. Communications of the ACM, 1959.

[5] 任志国，蔡小龙等. 插入排序算法的双链表模拟[J]，电脑编程与维护. 2010(06):8-9.

[6] Caire G. Tuto rial fo r Beginners [EB?OL]. h t tp: ?? jade. tilab. com? JADEP rogramm ing Tutorial for Beginners. pdf, 2003-12-01.

（上接第11页）

这种方法的思路是选取多个不同的初值，同时产生多条马尔科夫链，经过一段时间后，若这几条链稳定下来，则说明算法收敛了。在实际操作中，可在同一个二维图中画出这些不同的马尔科夫链产生的后验样本值对迭代次数的散点图，如果经过若干次迭代后，这些散点图基本稳定，重合在一起，则可判断其算法收敛。

3　Gibbs抽样算法在软件可靠性测试中的应用

在软件测试过程中，对于被测软件，基于逻辑覆盖和基本路径测试方法生成的测试用例，覆盖了程序的所有测试，保证了在测试中程序的每个可执行语句至少执行一次，由此可知：这些测试用例构成了一个完整的测试实验样本空间；生成的每个测试用例一定发生并且发生的概率相同。因此，可以把测试用例的生成及其发生概率和马尔可夫链模型联系起来。把一个完全正确的逻辑结构或模块对应的状态集看成在时刻t处于的状态集，则状态集中的每一个状态对应一个测试用例，从而构成一个完整的满足马尔可夫链模型的测试用例集；同样，把接下来要测试的逻辑结构或模块所处的状态看作下一时刻t+1处于的状态。通过以上对模型的分析，可以初步构造一个基于该模型的软件可靠性测试的结果分析准则，描述如下：

（1）s是被测逻辑结构或模块所对应的测试用例集，它应该是有限的、可列的集合或任意非空集。

(2)Q=[q_1, q_2, …, q_n]是时刻t测试用例的发生概率分布，q_i是在t时刻测试用例i的发生概率，满足$\sum_{j=1}^{N} q_{ij}=1$。其中，$q_1=q_2=\cdots=q_n$，即在t时刻，测试用例集中的每个测试用例发生概率相同。

结果分析准则的概率表示形式为Pz=(C，H)。

（1）C是被测逻辑结构或模块所对应的测试用例集，它应该是有限的、可列的集合或任意非空集。

(2)H是时刻t测试用例的发生概率分布。其中H=[h_1、$h_2\cdots h_k$]，P({h_i})表示t时刻第i个测试用例发生的概率，满足：

$$P(\{h_1\})=P(\{h_2\})=\cdots=P(\{h_k\})=1/k \quad (1)$$

$$P(\{h_1\})+P(\{h_2\})+\cdots+P(\{h_k\})=1 \quad (2)$$

即在t时刻，测试用例集中的每个测试用例全都发生，并且发生概率相同。

4　一个简单的实例分析

通过对某一软件在进行了单元测试、集成测试和系统测试后得出的有关各类错误数据资料整理分析得出了错误分类统计表[3]，如表1所示。

表1　错误分类统计表

错误分类	错误数	百分比
需求错误	1317	8.1
功能和性能错误	2624	16.2
结构错误	4082	25.2
数据错误	3638	22.4
实现与编码错误	1601	9.9
软件集成错误	1455	9.0
系统结构错误	282	1.7
测试定义与测试集成错误	447	2.8
其他类型错误	763	4.7
总计错误	16209	100.0

根据表1的资料可设α_1表示需求错误，α_2表示功能和性能错误，α_3表示结构错误，α_4表示数据错误，α_5表示实现与编码错误，α_6表示软件集成错误，α_7表示系统结构错误，α_8表示测试执行错误，α_9表示其它类型错误。由此得出“性质错误”类$\Omega_1=\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9\}$，其总体分布为$P(\alpha_1)=p_1, P(\alpha_2)=p_2, P(\alpha_3)=p_3, P(\alpha_4)=p_4, P(\alpha_5)=p_5, P(\alpha_6)=p_6, P(\alpha_7)=p_7, P(\alpha_8)=p_8, P(\alpha_9)=p_9$。由表1得出“性质错误”样本，其样本$\Omega_1$'总体容量为16209。由大数定理可以得$p_1\approx0.081, p_2\approx0.162, p_3\approx0.252, p_4\approx0.224, p_5\approx0.099, p_6\approx0.09, p_7\approx0.017, p_8\approx0.028, p_9\approx0.047$。

“性质错误”类Ω_1包括9类错误，总计错误的百分比100%，得出：这9类错误构成了完整的测试实验样本空间$\Omega1$。由于每类错误中的测试用例数目不同，所以其错误数的百分比不同，根据测试用例的概念[5]可知：所有类的各个测试用例是相互独立的，并且每个测试用例都得发生，即每个测试用例发生的概率是相同的。所以这个例子满足了提出的Pz=(C，H)这个式子。

5　结束语

MCMC方法依赖于模拟的收敛性，也即其构造的马尔科夫链是否收敛？何时收敛？判断收敛性的方法至今仍没有完全可靠的收敛性诊断方法，这使得收敛性的诊断问题成为MCMC方法实施的难点，有待探索。

参考文献

[1] NATO软件复用标准简介, 2007.

[2] 马瑞芳等. 计算机软件测试方法的研究[J]. 小型微型计算机系统, 2003.

[3] 李兴南，葛　玮，董云卫等. 一种基于大数定律的软件测试方法[J]. 微机发展, 2005.

静态链表上排序算法的研究

作者： 达文姣， 任志国， 王龙平， DA Wen-jiao， REN Zhi-guo， WANG Long-ping
作者单位： 兰州城市学院信息工程学院,甘肃,兰州,730070
刊名： 自动化与仪器仪表
英文刊名： AUTOMATION & INSTRUMENTATION
年，卷(期)： 2011(2)
被引用次数： 6次

参考文献(6条)

1. 严蔚敏;吴伟民 数据结构 1997

2. 耿国华 数据结构(C语言版) 2002

3. Robert L.Kruse;Alexander J.Ryba Data Structures and Program Design in C++ 2001

4. Shell D L A high-speed sorting procedure 1959

5. 任志国;蔡小龙 插入排序算法的双链表模拟[期刊论文]-电脑编程技巧与维护 2010(06)

6. Caire G Tuto rial fo r Beginners 2003

引证文献(6条)

1. 祁建宏. 朱正平. 安容瑾 多个栈共享内存空间通用算法研究[期刊论文]-自动化与仪器仪表 2012(4)

2. 吴超云. 郝庆一 冒泡排序算法在静态链表上的实现[期刊论文]-阜阳师范学院学报（自然科学版） 2013(4)

3. 祁建宏. 朱正平. 岳秋菊. 任志国. 达文姣 基于多条件的单链表快速排序算法研究[期刊论文]-自动化与仪器仪表 2012(3)

4. 高慧. 任志国. 岳秋菊. 达文姣 链式二路插入排序算法研究[期刊论文]-电脑编程技巧与维护 2012(8)

5. 岳秋菊. 达文姣. 瞿朝成. 任志国 链式存储结构上选择排序算法的研究与实现[期刊论文]-电脑编程技巧与维护 2011(18)

6. 达文姣. 朱正平. 任志国. 岳秋菊 链式存储结构上直接插入排序算法的研究与实现[期刊论文]-自动化与仪器仪表 2011(6)