文章编号：1007-757X(2010)10-0048-02

基于JAVA的快速排序

邢素萍

摘 要：阐述了排序对数据处理的重要性。采用流行的跨平台的面向对象程序设计语言java实现了快速排序的算法，并从理论上进行了算法分析。

关键词：信息；数据结构；数据；排序；Java
中图分类号：TP311　　**文献标志码**：A

0 引言

信息是计算机学科的基础，信息必须以数据的方式，按一定的规则存储在计算机的存储器中。对数据的操作方法如增加、插入、删除、查询等既要考虑数据的存储，又要考虑数据操作中数据的处理速度等各方面的因素。因此，怎样更科学的设计算法，采用哪种程序设计语言能更好地实现算法是非常重要的。

著名的瑞士科学家N.Wirth曾指出：算法+数据结构=程序。其中数据结构是指数据逻辑结构和物理结构，算法是对数据运算的描述。由此可见，程序实质是对具体问题选择一种好的数据结构，再设计一个好的算法，而好的算法通常取决于实际问题的数据结构。Java语言作为面向对象的程序设计语言，它提供了许多特性来持封装，每个数据项均封装在某个对象中。每条可执行的语句均由某个对象来完成。每个对象均是某个类的实例或是一个数组。每个类均在一个单继承的层次结构中定义。Java的单继承层次结构是一种树型结构，它的根是object类。Java的面向对象的特点使得它特别适合于数据结构的设计和实现，为程序员提供了一个很好的开发平台。因而，采用当今流行的跨平台程序设计语言java实现数据结构的设计，可以很好地解决对数据存储方式、处理方式的具体化问题。另外，利用java语言程序设计的灵活性及交互性，实现数据的动态输入，自动、分步、循环执行存储处理过程，并可在Internet上发布。

排序（Sorting）又称分类，广泛地应用在事务处理及各种数据加工的过程中，如果数据能够根据某种规则排序，就能大大提高数据处理的算法效率。如，在英文字典中按英文字母顺序单词排列单词，使我们可以很快速地从一本厚厚的字典中找到所要查找的单词。在计算机处理数据之前也要对其先进行排序，它是计算机程序设计中的一种重要操作。

1 排序的概念

所谓排序就是整理文件中的记录，使之按关键字递增（或递减）的次序排列起来。

假设含n个记录的序列为{R1,R2,...,Rn}，其相应的关键字序列为{K1,K2,...,K}，需确定1,2,...,n的一种排列Ri1,Ri2,...Rin，使其相应的关键字满足Ki1≤Ki2≤...≤Kin(或Kn≥Ki2≥...≥Kin)的关系。排序的对象是文件，它由一组记录组成。每条记录则由一个或若干个数据项（或域）组成。排序运算的依据是可用来标识一个记录的一个或多个组合数据项，我们称其为关键字（Key）。当待排序记录的关键字均不相同时，排序结果是惟一的，否则排序结果不惟一。在待排序的文件中，若存在多个关键字相同的记录，经过排序后这些具有相同关键字的记录之间的相对次序保持不变，该排序方法是稳定的；若具有相同关键字的记录之间的相对次序发生变化，则这种排序方法是不稳定的。快速排序被认为是目前基于比较记录关键字的内部排序中最好的排序方法。

2 快速排序的基本思想

快速排序是C.R.A.Hoare提出的一种划分交换排序。此排序方法采用了一种分治的策略，通常称其为分治法（Divide-and-ConquerMethod）。分治法的基本思想是：将原问题分解为若干个规模更小结构与原问题相似的子问题，递归地解这些子问题。然后将这些子问题的解组合为原问题的解。这样在用递归描述的分治算法的每一层递归上，都有以下3个步骤：

分解：将原问题分解为若干个子问题；

求解：递归地解各子问题，若子问题的规模足够小，则直接求解；

组合：将各子问题的解组合成原问题。

若把当前待排序的无序区设为R[low.....high],利用上面方法可将快速排序的基本思想描述为：

1．分解。在R[low.....high]任选一个记录作基准，以此基准将当前无序区划分为左、右两个较小的子区间R[low.....pivotpos-1]和R[pivotpos+1.....high],并使左边子区间中所记录的关键字均小于等于基准记录（设基准记录为pivot）的关键字pivot.key，右边的子区间中所有记录的关键字均大于等于pivot.key，而基准记录pivot则位于正确的位置（pivotpos）上，不用参加后续的排序。因此，划分的关键是要求出基准记录所在的位置pivotpos，划分的结果可以简单地表示为：

R[low.....pivotpos-1].keys≤R[pivotpos].key≤R[pivotpos+1.....high].key

这里low≤pivotpos≤high。应当注意：pivot=R[[pivotpos]。

2．求解。通过递归调用快速排序对左、右子区间R[low.....pivotpos-1]和R[pivotpos+1.....high]排序。

3．组合。因为当"求解"求解步骤中的两个递归调用结束时，其左、右两个子区间已有序，所以由上面的不等式立

作者简介：邢素萍（1957-），女，山东莱阳，南京工业职业技术学院，教授，研究方向：数据库，软件设计，南京 210046

即知道整个数组R已有序。

3 用java实现具体算法

```
package org.rut.util.algorithm.support;
import org.rut.util.algorithm.SortUtil;
public class ImprovedQuickSort implements SortUtil.Sort
{
private static int MAX_STACK_SIZE=4096;
private static int THRESHOLD=10;
public void sort(int[] data) {
int[] stack=new int[MAX_STACK_SIZE];
int top=-1;
int pivot;
int pivotIndex,l,r;
stack[++top]=0;
stack[++top]=data.length-1;
while(top>0){
int j=stack[top--];
int i=stack[top--];
pivotIndex=(i+j)/2;
pivot=data[pivotIndex];
SortUtil.swap(data,pivotIndex,j);
file://partition
l=i-1;
r=j;
do{
while(data[++l]<pivot);
while((r!=0)&&(data[--r]>pivot));
SortUtil.swap(data,l,r);
} while(l<r);
SortUtil.swap(data,l,r);
SortUtil.swap(data,l,j);
if((l-i)>THRESHOLD){
stack[++top]=i;
stack[++top]=l-1; }
if((j-l)>THRESHOLD){
stack[++top]=l+1;
stack[++top]=j; }  }
file://new InsertSort().sort(data);
insertSort(data);
}
private void insertSort(int[] data) {
int temp;
for(int i=1;i<data.length;i++){
for(int j=i;(j>0)&&(data[j]<data[j-1]);j--){
SortUtil.swap(data,j,j-1); }} }
}
```

4 算法分析

快速排序的时间主要耗费在划分操作上，对长度为k的区间进行划分，共需k-1次关键字的比较。

1.最坏的时间复杂度

最坏情况是每次划分选取的基准都是当前无序区中关键字最小（或最大）的记录，划分的结果是基准左边的子区间为空（或右边的子区间为空），而划分所得的另一个非空的子区间中记录数目仅仅比划分前的无序区中记录个数减少一个。

因此，快速排序必须做n-1次划分，第i次划分开始时区间长度为n-i+1,所需的比较次数为n-i($1\le i\le n-1$),故总的比较次数达到最大值：

$C_{max}=n(n-1)/2=O(n^2)$

如果按上面给出的划分算法，每次取当前无序区的第1个记录为基准，那么文件的记录已按递增序（或递减序）排列时，每次划分所取的基准就是当前无序区中关键字最小（或最大）的记录，则快速排序所需的比较次数反而最多。

2.最好的时间复杂度：最好情况下，每次划分所取的基准都是当前无序区的“中值”记录，划分的结果是基准的左、右两个无序子区间的长度大致相等。总的关键字比较次数为：$O(nlgn)$因为快速排序的记录移动次数不大于比较的次数，所以快速排序的最坏时间复杂度应为$O(n^2)$，最好的时间复杂度为$O(nlgn)$。

3.平均时间复杂度：尽管快速排序的最坏时间为$O(n^2)$，但就平均性能而言，它是基于关键字比较的内部排序算法中速度最快者，它的平均时间复杂度为O(nlgn)。

4.空间复杂度：快速排序在系统内部需要一个栈来实现递归。若每次划分较为均匀，则其递归树的高度为O(lgn)，故递归后需栈空间为O(lgn)。最坏情况下，递归树的高度为O（n），所需的栈空间为O(n)。

5.稳定性：快速排序是非稳定的。

目前，随着社会经济的飞速发展，要求计算机处理的信息量越来越大，在各种数据加工的过程中，如果数据能够根据某种规则排序，就能大大提高数据处理的算法效率。

5 结束语

本文分析了快速排序的算法。在实际应用中，若待排序的记录数量大，记录本身除关键字外的其它信息量也较大，而且对排序稳定性要求不很高时快速排序法是较好的选择。掌握排序的方法，可以在实际的应用过程中提高查找的效率。

参考文献

[1] “一种适合Java环境的中文快速排序和模糊检索方法”[J].《电脑知识与技术》2009(7).

[2] “中文数据排序与快速检索方法研究”[J].《微计算机信息》2007(3).

[3] “用Java语言改进插入排序算法”[J].《宜宾学院学报》2006(12).

[4] “自然归并算法的Java语言实现”[J].《濮阳职业技术学院学报》2006(4).

(收稿日期：2010-05-10)