DOI:10.19304/j.cnki.issn1000-7180.2002.06.002

快速排序算法研究

A Study on Quicksort Algorithm

西安电子科技大学 霍红卫 (西安 710071)
华 中 理 工 大 学 许 进 (武汉 430074)

摘 要：排序是计算机科学中最重要的研究问题之一。2000 年被列为 20 世纪对科学和工程计算的研究与实践影响最大的 10 大问题之一。文章介绍了基本的快速排序算法及三种枢轴元素的选取方法，全面深入地分析了快速排序算法最坏情况下的时间复杂度、平均情况下的时间复杂度、随机情况下的时间复杂度。并对快速排序算法和堆排序算法进行了比较，理论和实验结果表明，快速排序算法仍然是目前最好的排序算法之一。

关键词：快速排序，时间复杂度，枢轴元素，比较

1 引言

排序是计算机科学中最重要的研究问题之一。由于它的应用广泛和固有的理论上的重要性，2000 年它被列为对科学和工程计算的研究与实践影响最大的 10 大问题之一[1]。因此对于排序的研究既有理论上的重要意义，又有实际应用价值。它在计算机图形、计算机辅助设计、机器人、模式识别、及统计学等领域具有广泛应用[2]。基本的排序问题是重排一个给定的数据项集，使它按递增 (或递减) 排列。数据项可以是具有线性顺序的任意对象。例如，在典型商业数据处理应用中数据项是记录，每一个记录都包含有一个称为关键字的特殊标识符域，并且记录按关键字进行排序。排序能够大大简化查找或更新一个记录的过程。本文主要讨论内部排序问题，即所有待排序的数据项都一次性的装入主存。到目前为止，尽管研究人员已经设计了多种排序算法。但快速排序仍然是实际应用中最常用的一种排序算法。对它的复杂度分析方法和数据结构的研究是研究许多应用问题的基础。快速排序中使用的分治策略是设计有效计算几何和组合问题算法的典型设计方法。

Tony Hoare 在 1962 年首次提出了快速排序算法[3]，在过去 40 年里，对快速排序方法的研究表明，至今快速排序算法仍然是流传久远的最实用的排序算法。只在输入数据项有更详细的信息时，其它排序算法才可能胜过快速排序算法。快速排序算法的一个特性就是其复杂度分析和数据结构的丰富性，这些特性指导我们为各种组合应用问题研制相应的算法设计技术。尽管在最坏情况下，快速排序算法的性能不佳，但严格的证明表明，它的平均情况下的时间复杂度相当好。事实上，在某一排序复杂度模型之下，快速排序的平均复杂度是最可能的一种情况。更具体的说，如果输入有序序列进行排序，快速排序性能最差，而对于几乎所有随机的输入，快速排序性能极好。

快速排序采用了分治策略，它是设计有效组合优化问题算法的一种常用技术。Hoare 在他最初的论文中提到的快速排序的另一种形式是利用随机数产生器的随机排序。研究人员以后逐渐建立了这种思想，并极大地丰富了随机算法，这是理论计算机科学中最有趣的研究领域之一。本文介绍了基本的快速排序算法及三种枢轴元素的选取方法，全面深入地分析了快速排序算法最坏情况下的时间复杂度、平均情况下的时间复杂度、随机情况下的时间复杂度。并对快速排序算法和堆排序算法进行了比较，理论和实验结果表明，快速排序算法仍然是目前已有的最好的排序算法之一。

2 快速排序算法及枢轴元素的选取

快速排序算法是利用分治技术的典型例子。分治策略分为三步。首先将问题分成若干大小基本相等的子问题；独立地解决这些子问题；最后将子问题归并成原问题的解。因此方法的有效性取决于对初始问题划分的过程和最后一步归并解的过程。如同快速排序一样，FFT 也沿用这种方法。

2.1 快速排序算法

假设待排序的 n 个元素存储在数组 A[0. n－1] 中。快速排序算法的高级描述如下：

(1) 从 A[0. n－1]中选取枢轴元素。

(2) 重排 A 中元素，并将其划分成左右两部分。使得数组中所有比枢轴元素小的元素在左部分中，比它大的元素在右部分中。

(3) 对左、右部分进行递归排序。

如果先不看实现的细节和算法的正确性证明，

收稿日期：2002－02－22
基金项目：国家“十五”预研资助项目

不难看出算法使用了分治策略。在这种情况下，第一、二步相应于划分步，第三步求解归约的子问题，实现对整个数组的排序，从而无需归并步骤。在快速排序算法的描述中，忽视了两个关键的问题：

(1) 选择枢轴元素的方法；

(2) 如枢轴元素被选择后，使用的划分方法。

2.2 枢轴元素的选取

由于有效分治算法的一个主要假设是将输入分成几乎大小相同的部分，因此所选择的枢轴元素应能够将数组 A 分成大小几乎相等的部分。假设 A 由 n 个不同元素组成，最好的选择方法是选择 A 中元素的中值作为枢轴元素。尽管有些理论上的好的算法可以找到中值无需排序，但由于开销过大使得快速排序无法在实际中得到实用。

在实现中，有三种基本的方法选择枢轴元素。第一种是从 A 的固定位置选择枢轴元素，比如第一个作为枢轴元素。这种选择方法，除非输入元素随机，否则效果不会好。因为如果输入几乎有序，在每次迭代过程中，输入将被极不均匀的划分，导致算法性能差。第二种是通过计算 A 中元素子集的中值近似作为 A 的中值。普遍使用的方法是计算 A 中第一、中间和最后元素的中值作为 A 的中值。实践表明，采用三元素取中值的规则可大大改善快速排序在最坏情况下的性能。尽管对于某种输入仍然有可能遇到不均匀的划分。第三种方法是用随机数产生器选择枢轴元素。在这种情况下，可以严格的证明迭代的每一步以非常大的概率导致均匀划分，与初始输入分布无关。

2.3 输入元素的划分过程

由于是递归算法，因而描述划分的过程由子数组 A[l..r]开始，其中 l < r。过程利用两个指针 i 和 j，i 从位置 l 开始从左向右移动，j 从位置 r 开始从右向左移动，指针 i 不断增加直到遇到大于枢轴元素的元素 A[i] 为止，指针 j 不断减小直到遇到小于枢轴元素的元素 A[j] 为止，此时交换 A[i] 和 A[j]，过程继续直到两个指针相遇，表明一趟划分过程的结束。过程详见图 1，假设 A 中元素互不相同，最左边的元素作为枢轴元素。j 的初始值为 r + 1 以满足第二步 While 循环从一开始就可以检查到子数组的最右边的元素。第二步的最后两个赋值语句保证枢轴元素放到 A 中排序的正确位置。划分过程的复杂度为线性时间，有一很小的常量因子。

快速排序调用划分过程的初始参数值是 l = 0 和 r = n - 1，接着递归调用子数组 A[l, j - 1] 和 A[j + 1, r]。可以用栈保存要被排序的划分。

```
Procedure partition(A, l, r)
begin
Step1.  i = 1; j = r + 1; pivot = A[1];
Step2.  while (true) {
          while (A[ + + i] < pivot);
          while (A[ - - j] > pivot);
          if i <j then A[i] <- -> A[j];
            else break; }
          A[1] = A[j];
          A[j] = pivot;
end
```

图 1　A[i, r]的划分过程

3 算法复杂度分析

可以用几个测度作为衡量排序算法的性能。由于元素比较是排序算法中所进行的主要操作，因此可用算法中所进行的元素比较次数作为衡量算法性能的标准。算法中出现的元素交换数也是重要的参量。另一个主要因素是数据移动和它的空间局部性，这在使用分级存储器，典型情况下是二级高速缓存器和主存时显得尤为重要。本文以元素比较次数作为衡量算法性能的标准，来导出排序的非线性下界。

3.1 最坏情况时间复杂度分析

假设 T(n) 是快速排序算法所做的元素比较次数，则有：

$$T(n) = \max_{1 \le i \le n-1} \{T(i) + T(n-1)\} + cn$$

$$T(1) = \Theta(1)$$

其中 cn 是输入元素首次划分时所需的元素比较次数的上界。参数 i 是第一次迭代后左划分部分的元素个数。由于是分析算法的坏条件情况。显然，最坏情况是当 i = 1(或 i = n - 1) 时，因此 $T(n) = \Theta(n^2)$。然而从复杂度分析的角度来看，快速排序算法还在于它的极好的平均时间复杂度。

3.2 平均情况时间复杂度分析

尽管进行算法的坏条件复杂度分析通常较容易，但其结果的性能估计常常令人失望。如果知道平均情况，则可以进行比较。但导出平均情况复杂度更为困难。一种方法是假设输入符合某一概率分布来建立复杂度。这种方法不仅导出界限困难，而且不能给出输入概率分布的足够情形。

另一种方法是算法采用随机数产生器，这种情况进行算法分析不需对输入分布做任何假设。结论对任何输入分布成立。这种方法即所谓的随机算

法。这里,考虑数组的第一个元素作为枢轴元素,输入元素随机。因此 A 中任一元素作为枢轴元素的概率相等。平均复杂度可由下面递推方程给出:

$$T(n)=\frac{1}{n}\left(\sum_{i=0}^{n-1}(T(i)+T(n-i-1))\right)+cn \quad (1)$$

$$T(1)=\Theta(1)$$

方程(1)等价于:

$$T(n)=\frac{2}{n}\left(\sum_{i=0}^{n-1}T(i)\right)+cn \quad (2)$$

方程(2)两边乘 n,得:

$$nT(n)=2\left(\sum_{i=0}^{n-1}T(i)\right)+cn^2 \quad (3)$$

$$nT(n)-(n-1)T(n-1)=2T(n-1)+2cn-c$$

整理并省略 c(渐进无关),得:

$$nT(n)=(n+1)T(n-1)+2cn \quad (4)$$

两边除以 n(n+1),得:

$$\frac{T(n)}{n+1}=\frac{T(n-1)}{n}+\frac{2c}{n+1}$$

$$\frac{T(n-1)}{n}=\frac{T(n-2)}{n-1}+\frac{2c}{n}$$

$$\frac{T(2)}{3}=\frac{T(1)}{2}+\frac{2c}{3}$$

上述方程相加,且利用:

$$\sum_{i=3}^{n+1}\frac{1}{i}=\log_e(n+1)+\gamma-\frac{3}{2}$$

其中 $\gamma=0.577$ 为欧拉常数。可得:

$$T(n)=O(n\log n)$$

在一般条件下,没有一种基于比较的排序算法能突破这一下界。这一分析证实了快速排序在大多数情况下的良好性能。

3.3 随机情况时间复杂度分析

考虑枢轴元素随机选择的情况。不论初始分布是什么,快速排序算法的复杂度以概率 $1-n^{-c}$ 达到 $O(n\log n)$,其中 c 为某一正常数。我们将快速排序算法看作迭代的序列,第一次迭代将输入划分成两个桶,第二次迭代将上次的两个桶划分成四个桶,重复这一过程,直到每个桶足够小(比如,小于 30 个元素)。然后用插入排序方法对桶内元素进行排序。可从另一角度看这一问题。将递归展成一棵划分树,树的每一层对应迭代划分。每次迭代过程的时间复杂度为 O(n)。因此只需证明迭代次数以极大概率为 $O(\log n)$。

对于任一确定的元素 e,包含元素 e 的任意两个连续桶的大小以某一概率下降一常量因子(定理 1)。经过 $O(\log n)$ 次迭代后,以概率 $1-n^{-7}$,包含元素 e 的桶大小将是 30 或更小,由 Boole 不等式,可得 $O(\log n)$ 次(定理 2)迭代后,每个桶的大小以概率 $1-n^{-6}$ 小于等于 30。

令 e 是数组 A 中的任一元素。n_j 为第 j 次划分后,包含元素 e 的桶的大小,$j\geq 1$。设 $n_0=n$,则有:

定理 1 $\Pr\{n_{j+1}\geq 7n_j/8\}\leq 1/4$,对任意 $j\geq 0$。

证明:元素 α 将第 j 个桶划分成两个桶,其中之一大小至少为 $7n_j/8$,当且仅当 $\mathrm{rank}(\alpha:B_j)\leq n_j/8$ 或 $\mathrm{rank}(\alpha:B_j)\geq 7n_j/8$。随机元素在 B_j 中最小(或最大)的 $n_j/8$ 个元素中的概率至多为 1/4。定理 1 成立。

我们固定 A 中的元素 e,在各种划分步中,考虑包含 e 的桶的大小。称第 j 次划分步是成功的,如果 $n_j\leq 7n_{j-1}/8$。因为 $n_0=n$,在 k 次成功划分后,包含 e 的桶的大小小于 $(7/8)^k n$。因此,e 至多参与 $c\log(n/30)$ 次成功划分步,这里 $c=1/\log(8/7)$,对于证明的其余部分,所有对数都是以 8/7 为底;于是常数 c 等于 1。

定理 2 在 $20\log n$ 次的划分中,元素 e 经过 $20\log n-\log(n/30)$ 次不成功的概率为 $O(n^{-7})$。

证明:各划分中所做的随机选择相互独立,因此组成成功划分步的事件相互独立。这些事件可模型化为 Bernouli 试验[4,5]。令 X 是一个随机变量,表示 20 次 $\log n$ 步中不成功的划分步的数目。于是:

$$\Pr\{X>20\log n-\log(n/30)\}(\Pr\{X>19\log n\}$$

$$\leq\sum_{j>19\log n}\binom{20\log n}{j}\left(\frac{1}{4}\right)^j\left(\frac{3}{4}\right)^{20\log n-j} \quad (5)$$

假定:$\binom{n}{k}\leq\left(\frac{en}{k}\right)^k$

则由(5)式,可得:

$$\leq\sum_{j>19\log n}\left(\frac{20e\log n}{j}\right)^j\left(\frac{1}{4}\right)^j=\sum_{j>19\log n}\left(\frac{20e\log n}{j}\right)^j$$

$$\leq\sum_{j>19\log n}\left(\frac{5e\log n}{19\log n}\right)^j=\sum_{j>19\log n}\left(\frac{5e}{19}\right)^j=O(n^{-7})$$

因此,定理 2 成立。

由 Boole 不等式,A 中一个或多个以上元素经历 $20\log n-\log(n/30)$ 次不成功步的概率至多为 $O(n\times n^{-7})=O(n^{-6})$。因此,算法以概率 $1-O(n^{-6})$ 在 $20\log n$ 次迭代后终止。由此可得,以极大概率算法时间复杂度为 $O(n\log n)$。

4 快速排序算法与堆排序算法的比较

尽管快速排序在原数组中重排数据元素,但不

能保证对问题进行均匀划分，因此导致它的最坏情况 $O(n^2)$。归并排序却能保证均匀划分，并且几乎有最优的最坏情况，但它不能在原数组中进行排序，它需要相当大的辅助空间。堆排序在原数组中进行，最坏情况下的时间复杂度为 $\Theta(n\log n)$。在某种意义上说，堆排序结合了快速排序和归并排序的优点。那么，堆排序能否替代快速排序呢？回答是否定的。原因有两方面：一方面，较之快速排序与归并排序，堆排序的复杂度具有较高的常量因子。因为在快速排序中，它需要较少的元素比较次数和交换次数。另一方面，利用快速排序算法和堆排序算法，对 n = 100，200，500 个元素的排序结果如图 2 和图 3 表明了快速排序的优势。

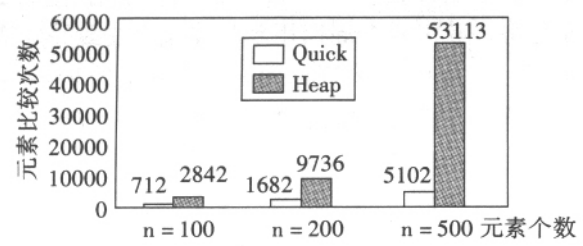

图 2　快速排序与堆排序元素比较次数图

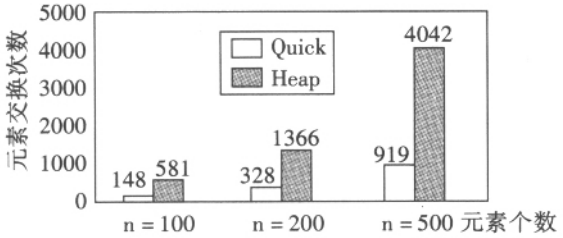

图 3　快速排序与堆排序元素交换次数图

快速排序的基本策略——随机分治策略，是设计组合优化与计算几何一类问题有效算法的基础。尤其是对于设计计算几何中的高维问题和组合优化中的问题的并行算法具有重要意义。

参考文献

[1] J Dongarra. The Top 10 Algorithms. IEEE Computing in Science & Engineering, 2000, 2(1): 22 ~ 23.

[2] T H Cormen, C E Leiserson, R L Rivest. Introduction to Algorithms. MIT Press, September, 2001, II Sorting and Order Statistics.

[3] C A R Hoare. Quicksort. The Computer J., 1962, 15(1): 10 ~ 15.

[4] K Mulmuley. Computational Geometry: An Introduction through Randomized Algorithms. Prentice Hall, Upper Saddle River, N. J., 1994.

[5] D Helman, D Bader, and J Jala. A Randomized Parallel Sorting Algorithm with an Experimental Study. J Parallel and Distributed Computing, 1998, 52(1): 1 ~ 23.

HUO Hong-wei
(School of Computer Science, Xidian University, Xi'an 710071)
XU Jin (Department of Control Science and Engineering, HUST, Wuhan 430074)

Abstract: Sorting is arguably the most studied problem in computer science, because of both its use in many applications and its intrinsic theoretical importance, one of the ten algorithms with the greatest influence on the development and practice of science and engineering in the 20th century. This paper introduces the basic quicksort algorithm and three methods of choice of the pivot and gives a broad of its worse – case、average – case and randomized complexity analysis. A comparison on the quicksort and heapsort is made. Some empirical data show that quicksort is still one of the best sorting algorithms.

Key words: Quicksort, Time complexity, Pivot, Comparison

霍红卫　博士，教授。主要研究方向为算法、并行与分布式计算、DNA 算法及遗传算法。
许　进　教授，博士生导师，华中科技大学特聘教授。主要研究方向为图论、DNA 计算机、神经网络、遗传算法。