第七章 快速排序

```
QUICKSORT(A, p, r)
if p < r
    then q ← PARTITION(A, p, r)
         QUICKSORT(A, p, q − 1)
         QUICKSORT(A, q + 1, r)
```

快速排序是基于分治模式的：

分解：数组$A[p..r]$被划分成两个（可能空）子数组$A[p..q-1]$和$A[q+1..r]$，使得$A[p..q-1]$中的每个元素都小于等于$A[q]$，而且，小于等于$A[q+1..r]$中的元素。下标q也在这个划分过程中进行计算。

解决：通过递归调用快速排序，对子数组$A[p..q-1]$和$A[q+1..r]$排序。

合并：因为两个子数组使就地排序的，将它们的合并不需要操作：整个数组$A[p..r]$已排序。

数组划分：

```
PARTITION(A, p, r)
1  x ← A[r]
2  i ← p − 1
3  for j ← p to r − 1
4       do if A[j] ≤ x
5             then i ← i + 1
6                  exchange A[i] ↔ A[j]
7  exchange A[i + 1] ← A[r]
8  return i + 1
```

在第 3 到 6 行中循环的每一轮迭代的开始，对任何数组下标k，有：

1. 如果$p \leqslant k \leqslant i$，则$A[k] \leqslant x$
2. 如果$i+1 \leqslant k \leqslant j-1$，则$A[k] > x$
3. 如果$k = r$，则$A[k] = x$

证明：

初始化：在循环的第一轮迭代前，有$i = p - 1$和$j = p$。在p和i之间没有值，在$i + 1$和$j - 1$之间也没有值。

保持：在第四行中，如果$A[j] > x$，那么j增加 1，在j增加 1 后，条件 2 对$A[j-1]$成立，并且其他性质保持不变；如果$A[j] \leqslant x$，那么将i增加 1，交换$A[i]$和$A[j]$，再将j增加 1。因为进行了交换，现在有$A[j] \leqslant x$，因而条件 1 满足。类似地，还有$A[j-1] > x$，因为根据循环不变式，被交换进$A[j-1]$的项目是大于x的。

终止：当终止时，$j = r$。于是，数组中的每个元素都在循环不变式所描述的三个集合的某一个之中，亦即，我们已将数组中的所有元素划分成了三个集合：一个集合中包含了小于等于 x 的元素，第二个集合中包含了大于x的元素，还有一个只包含了x的集合。在$PARTITION$过程的最后两行中，通过将主元与最左的、大于x的元素进行交换，就将他移动到了它在数组中间的位置上。此时，$PARTITION$的输出满足分解步骤所做规定的要求。

$PARTITION$在之数组$A[p\,.\,.\,r]$上的运行时间为$\Theta(n)$，其中$n = r - p + 1$

练习

7.1-1

$A =< 13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11 >$

1. $A =< 9, 19, 13, 5, 12, 8, 7, 4, 21, 2, 6, 11 >$
2. $A =< 9, 5, 13, 19, 12, 8, 7, 4, 21, 2, 6, 11 >$
3. $A =< 9, 5, 8, 19, 12, 13, 7, 4, 21, 2, 6, 11 >$
4. $A =< 9, 5, 8, 7, 12, 13, 19, 4, 21, 2, 6, 11 >$

5. $A =< 9, 5, 8, 7, 4, 13, 19, 12, 21, 2, 6, 11 >$

6. $A =< 9, 5, 8, 7, 4, 2, 19, 12, 21, 13, 6, 11 >$

7. $A =< 9, 5, 8, 7, 4, 2, 6, 12, 21, 13, 19, 11 >$

8. $A =< 9, 5, 8, 7, 4, 2, 6, 11, 21, 13, 19, 12 >$

7.1-2：

$r-1$

修改：

```
PARTITION(A, p, r)
x ← A[r]
i ← p − 1
for j ← p to r − 1
        do if A[j] ⩽ x
              then i ← i + 1
                    exchange A[i] ↔ A[j]
exchange A[i + 1] ← A[r]
if i = r − 1
    then return ⌊p + r⌋/2
    else  return i + 1
```

7.1-3：

$$MAX : 1+1+[(r-1-p+1)+1]+[(r-1-p+1)]\times 3+1+1$$
$$MIN : 1+1+[(r-1-p+1)+1]+[(r-1-p+1)]+1+1$$
$$\because n = r-p+1$$

所以有：

$$MAX : 1+1+[(n-1)+1]+[(n-1)]\times 3+1+1 = 4n+1$$
$$MIN : 1+1+[(n-1)+1]+[(n-1)]+1+1 = 2n+3$$

故$PARTITION$的运行时间是$\Theta(n)$

7.1-4：

$\text{PARTITION}(A, p, r)$

$x \leftarrow A[r]$
$i \leftarrow p - 1$
for $j \leftarrow p$ **to** $r - 1$
    do if $A[j] \geqslant x$
        then $i \leftarrow i + 1$
            exchange $A[i] \leftrightarrow A[j]$
exchange $A[i+1] \leftarrow A[r]$
return $i + 1$

快速排序的性能：

如果（$PARTITION$过程）划分是对称的，那么本算法从渐进意义上讲，与合并排序算法一样快（$\Theta(n \lg n)$）；如果划分是不对称的，那么本算法渐进上就和插入算法一样慢(插入算法最佳$\Omega(n)$，最差$O(n^2)$)。

最坏情况：

快速排序的最坏情况发生在划分过程产生的两个区域分别包含$n-1$个元素和 1 个 0 元素（空的）的时候。假设算法每次递归调用中都出现了这种不对称划分。划分的时间代价为$\Theta(n)$。因为对一个大小为 0 的数组进行递归调用后，返回$T(0) = \Theta(1)$，故算法的运行时间可以递归地表示为：

$$T(n) = T(n-1) + T(0) + \Theta(n) = T(n-1) + \Theta(n)$$

得到$T(n) = \Theta(n^2)$

最佳情况：

当$PARTITION$过程中两个子问题规模分别为$\lfloor n/2 \rfloor$和$\lceil n/2 \rceil - 1$时，快速排序运行达到最佳。其运行时间可以递归地表达为：

$$T(n) \leqslant 2T(n/2) + \Theta(n)$$

解得$T(n) = O(n \lg n)$

因此，快速排序的最佳运行时间是$\Omega(n \lg n)$，最差运行时间是$O(n^2)$。

平衡的划分：

快速排序的平均情况运行时间与其最佳情况运行时间很接近，而不是非常接近于其最坏情况运行时间。

简单的验证：

假设划分过程很不平衡（就是趋向于出现最坏情况，但还没到最坏情况），划分比例为$(k-1):1$（k的值使得划分明显不平衡，比如使$k=10$时划分比例为$9:1$），因此有

$$T(n)\leqslant T\left(\frac{k-1}{k}n\right)+T\left(\frac{1}{k}n\right)+cn$$

构造一棵递归树，在这棵递归树上，每层的代价是一样的（因为递归过程中T前的系数是 1），都是cn。

树的最长路径沿着$n\to(k-1/k)n\to(k-1/k)^2n\to\cdots\to1$，最短路径沿着$n\to(1/k)n\to(1/k)^2n\to\cdots\to1$，因此得到两个路径的长度分别为$\log_{\frac{k}{k-1}}n$和$\log_k n$。

这里先介绍一个结果：$\log_\alpha n=\Theta(n)$，简证：

$$\begin{aligned}&c_1\lg n\leqslant\log_\alpha n\leqslant c_2\lg n\\ \Longleftrightarrow\ &c_1\lg n\leqslant\frac{\lg n}{\lg\alpha}\leqslant c_2\lg n\\ \Longleftrightarrow\ &c_1\leqslant\log_\alpha 2\leqslant c_2\end{aligned}$$

因此，上面两个路径的长度都是$\Theta(\lg n)$。因此总代价都是$cn\times\Theta(\lg n)=\Theta(n\lg n)$。

这里，我们整个分析过程都是假设等号成立，因此取得的是上界，故总的运行时间应当是$O(n\lg n)$。

这个分析过程说明，快速排序一般情况下的运行状况都接近于最佳状况。

练习

7.2-1

$n-1=m$做替换后得到$T(m+1)=T(m)+\Theta(m)$，假设$T(m)=\Theta(m^2)$，则有:

$$c_1m^2+\Theta(m)\leqslant m^2\leqslant c_2m^2+\Theta(m)$$

假设匿名函数$\Theta(m)=m$，则$c_1m^2+m\leqslant m^2\leqslant c_2m^2+m$

因此$c_1\leqslant\dfrac{m}{m+1},c_2\geqslant\dfrac{m}{m+1}$

取$c_1=\dfrac{1}{2},c_2=1$即可满足。

故$T(n)=T(m+1)=\Theta(n^2)$

7.2-2

这个就是快速排序的最坏情况，因此是$\Theta(n^2)$

7.2-3

每一步都是分成了 0 个元素加上$n-1$个元素，因此是最坏情况，利用 7.2-1 的递归式可以得到运行时间为$\Theta(n^2)$

7.2-4

```
INSERTION-SORT(A)
for j ← 2 to length[A]
    do key ← A[j]
       ▷ Insert A[j] into the sorted sequence A[1 . . j − 1].
       i ← j − 1
       while i > 0 and A[i] > key
           do A[i + 1] ← A[i]
              i ← i − 1
       A[i + 1] ← key
```

根据两种排序算法的性质，对一个具有较好的已排序程度的数组，$INSRTION-SORT$的运行时间是$O(n)$，即接近于$INSRTION-SORT$的最佳运行时间，而$QUICKSORT$恰好相反，对一个具有相当排序程度的数组，$QUICKSORT$的运行时间更趋向于最坏情况，即$\Theta(n^2)$，因此在这类问题上插入排序往往比快速排序具有更好的性能。

7.2-5

递归树是$T(n)=T\left((1-\alpha)n\right)+T\left(\alpha n\right)+cn$，因此最短和最长两个路径（先不考虑哪

个长、哪个短）是$n \to (1-\alpha)n \to (1-\alpha)^2 n \to \cdots \to 1$和$n \to \alpha n \to \alpha^2 n \to \cdots \to 1$

算得两个路径的长度是$-\lg n/\lg\alpha$，$-\lg n/\lg(1-\alpha)$

因为$0 < \alpha \leqslant 1/2$，因此$\alpha \leqslant 1-\alpha$，所以$-1/\lg\alpha < -1/\lg(1-\alpha)$，故最大深度是

$-\lg n/\lg(1-\alpha)$，最小深度是$-\lg n/\lg\alpha$。

快速排序的随机化版本：

$\text{RANDOMIZED-PARTITION}(A,p,r)$

$i \leftarrow RANDOM(p,r)$
$exchange\ A[r] \leftrightarrow A[i]$
return $PARTITION(A,p,r)$

$\text{RANDOMIZED-QUICKSORT}(A,p,r)$

if $p < r$
  then $q \leftarrow RANDOMIZED-PARTITION(A,p,r)$
    $RANDOMIZED-QUICKSORT(A,p,q-1)$
    $RANDOMIZED-QUICKSORT(A,q+1,r)$

练习

7.3-1

因为前面已经有过结论，快速排序的平均运行时间与最佳运行时间很接近，而不是非常接近于最坏情况运行时间。

7.3-2

最坏情况：$\Theta(n)$，最佳情况$\Theta(1)$