§5.3 快速排序

一．例子：将以下序列排序：3, 1, 4, 1, 5, 9, 2, 6, 5, 3

\***取前两个不同元素的大者作标杆**

左指针从左向右扫描，右指针从右向左扫描，当左边标杆，右边标杆时，交换左、右元素，继续上述过程，直到左指针到右指针的右边。

分成两组之后，左指针左边的是小于标杆的，右边是大于等于标杆的，之后递归这两组

二．算法：

FUNCTION Findpivot (i, j : integer) : integer;（返回标杆的下标）

VAR

firstkey : keytype;

k : integer;

BEGIN

firstkey := A[i].key;

FOR k := i + 1 TO j DO

IF A[k].key > firstkey THEN

RETURN(k)

ELSE IF A[k].key < firstkey THEN

RETURN( i );

RETURN(0);

END;

FUNCTION Partition (i, j : integer; pivot : keytype) : integer;

/\*将A[ i ], , A[ j ]分成两组，左边一组的关键字均小于标杆，右边一组的关键字均大于等于标杆，返回右边一组的第一个元素的下标\*/

VAR

l, r : integer;

BEGIN

l := i ; r := j ;

WHILE l r DO

BEGIN

WHILE A[ l ].key < pivot DO

l := l +1;

WHILE A[ r ].key pivot DO

r := r 1;

IF l < r THEN swap(A[ l ], A[ r ]);

END;

RETURN( l );

END;

PROCEDURE Quicksort (i, j : integer);

/\* 将排序 \*/

VAR

pivot : keytype;

pivotindex : integer;

k : integer;

BEGIN

1. pivotindex := Findpivot(i, j);
2. IF pivotindex 0 THEN

BEGIN

1. pivot := A[pivotindex].key;
2. k := Partition (i, j, pivot);
3. Quicksort (i, k-1);
4. Quicksort (k, j);6

END

END;

三．最坏运行时间

1. Findpivot : 最坏情况下：；

2. Partition：最坏情况下：；

游标指向到中每一个元素时，所花的时间不超过某一个常数，而游标决不会再回到其中任一元素，因为当时，该过程停止。故执行时间不超过。

3. Quicksort

(1) ：

(2), (3) :

(4) :

每次递归调用的时间与该调用要排序的元素个数成比例。总的运行时间等于所有递归调用中要排序的元素个数之和，也等于每个元素出现在不同递归调用中的次数之和。即在一开始例子中，每个元素排好序时所在的层数求和。

在最坏情况下，处理一个已排好序从小到大的数组(见图5.3)。

的层数为，的层数为n，各元素深度之和为：

故最坏情况下，需时间。

四．平均运行时间：

假设中元素各不相同。输入元素的各种顺序概率相同。递归调用时，元素的各种顺序概率也相同，标杆出现在排好序的数组中第2个，第3个，，第个位置的概率相同。

假设标杆出现在第个位置上，Partition后，左边有i个元素，右边有(n-i)个元素。

根据选标杆的方法，或者标杆在第一个位置，比标杆小的i个元素之一在第二个位置；或者比标杆小的i个元素之一在第一个位置, 标杆在第二个位置。

标杆在第一个位置的概率：

比标杆小的i个元素之一在第二个位置的概率：

第一种情况的概率：

同理，第二种情况的概率：

设对个元素排序的平均时间为。

T(1)

上式中为将个元素Partition所需的时间。

因为 (5.2)

故 (5.3)

(5.4)

(5.5)

要解(5.5)式，我们猜测一个结果 ， 然后用归纳法证明。

时，

设时，

(5.6)

(5.7)

取，则。

**故平均时间复杂度为。**

**\*快速排序平均时间比堆排序平均时间少一个常数因子，因而平均时间最少。**

作业8：

1.设是k个整数集合，其中的整数值域从1到, 这k个集合的元素个数之和为n。设计一个时间的算法，将所有排序。

2.证明：快速排序在最好情况下运行时间为。