第五章 分类与查找

§5.1 基于比较排序算法的时间下界

一．基于比较排序算法举例

插入排序

PROCEDURE Insertion\_Sort (A);

BEGIN

A[0].key := ;

FOR i := 2 TO n DO

BEGIN

j := i;

WHILE A[ j ].key < A[ j-1 ].key DO

BEGIN

swap(A[ j ], A[ j-1]);

j := j-1;

END

END

END;

二．判定树

基于比较的排序算法，其过程可用一个判定树来描述。

例：插入排序，A[1], A[2], A[3]中存放元素，排序过程的判定树见图5.1。

判定树是一棵二叉树，其每个结点代表算法比较元素后得到的状态。

图5.1

三．基于比较排序算法的时间下界

\*要点：

要将n个元素排序，n个元素的n!种排列都是可能的。

判定树至少有n!个叶子，因为每一种可能的排序是一个叶子。

删掉对应不必要的比较的结点，删掉对应无序的叶子(前面的比较不相容)，恰有n!个叶子。

判定树从根结点到叶子的最长路径的结点数是最坏情况下，排序所需比较次数的下界。

定理5.1：用基于比较的方法把n个元素排序至少需要次比较，其中为某个大于0的常数。

证明：首先证明：高度为h的二叉树有至多个叶子。

当时，二叉树只有根结点，恰好有个叶子。

设当时()，二叉树有**至多个（n!）叶子**。当时，根结点有左、右两个子树，每个子树的高度小于等于。于是二叉树的叶子数。

再证任何将n个不同元素排序的判定树**高度至少为** 。

因为将n个元素排序的结果可以是输入的n个元素的n!个排列中任何一个，故判定树至少有n!个叶子。由上面的结论，它的高度至少为。

最后证明本定理的结论。由于，

因此，。证毕。

结论：

1. 基于比较排序算法最坏情况下时间复杂度为。
2. 基于比较排序算法平均时间复杂度为。

§5.2 堆排序

一．例子：

将下列元素建成堆并排序

从最后一个有孩子的开始，一直做pushdown（把孩子里最小的换上来），之后更新指针指向其前一个元素（注意不是父亲），再接着继续做pushdown。使得堆的顶部为最小元。之后将最小元和最后一个元素互换值，之后从顶开始pushdown到n-1个元素。使得最后的集合是从大到小的排序。

图5.2

二．算法：

PROCEDURE Pushdown (first, last : integer);

VAR

r : integer;

BEGIN

r := first;

WHILE r last DIV 2 DO

IF 2\*r = last THEN

BEGIN

IF A[r].key > A[2\*r].key THEN

swap(A[r], A[2\*r]);

r := last

END

ELSE IF (A[r].key > A[2\*r].key) AND 【last>2r】

(A[2\*r].key A[2\*r+1].key) THEN

BEGIN

swap(A[r], A[2\*r]);

r := 2\*r;

END

ELSE IF (A[r].key > A[2\*r+1].key) AND

(A[2\*r+1].key < A[2\*r].key) THEN

BEGIN

swap(A[r], A[2\*r+1]);

r := 2\*r+1

END

ELSE r := last【父亲不大于小儿子】

END;

PROCEDURE Heapsort (A);

/\*将A[1], A[2], , A[n]排序\*/

VAR

i : integer;

BEGIN

1. FOR i := n DIV 2 DOWNTO 1 DO
2. Pushdown (i, n);【建堆】

(3) FOR i := n DOWNTO 2 DO【第一个元素和最后一的元素互换】

BEGIN

(4) swap(A[1], A[i]);

(5) Pushdown (1, i-1);

END

END;

三．算法分析：

1. 最坏情况下时间复杂度：

\* Pushdown过程：

每次做WHILE循环比较两次，

r初值为first, 经过i次循环，r first\*，

最后r > last/2 , 即first\* > last/2，

即 。

WHILE循环次数。

因为且，

故循环次数，比较次数。

Pushdown过程时间复杂度为。

\*Heapsort过程：

(1)-(2)行：时间复杂度为

(3)-(4)行：时间复杂度为

(3)-(5)行：时间复杂度为

总的时间复杂度：。

2. 平均时间复杂度：。

作业7：

1. 以下算法称为Shell排序

PROCEDURE Shellsort (VAR A : ARRAY [1] OF integer);

VAR

i, j, incr : integer;

BEGIN

incr := n DIV 2;

WHILE incr > 0 DO

BEGIN

FOR i := incr+1 TO n DO

BEGIN

j := iincr;

WHILE j > 0 DO

IF A[ j ] > A[ j + incr ] THEN

BEGIN

swap(A[ j ], A[ j + incr ]);

j := j incr;

END

ELSE j := 0 /\* break \*/

END;

incr := incr DIV 2

END

END;

1. 用Shell排序算法将下列序列排序：22, 36, 6, 79, 26, 45, 75, 13, 31, 62, 27, 76, 33, 16, 62, 47。写出排序过程。

试分析Shell 排序算法的最坏情况下时间复杂性。(选做。提示：最坏情况下时间复杂度为。)

§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。**