排序算法总结及习题

一、概述

排序是最基础和常用的算法之一，一般情况下，排序不开比较、数据交换，怎样降低算法的时间及空间复杂性是算法设计的目标，尽管经典算法已有不少，但研究一直不断，2001年还有综合性能很好的新算法出现。

**为了对n个元素的线性表进行排序，至少必须扫描一遍以获取n各元素，因此排序问题的计算复杂性下界为：**

**Ω(n)**

**如果对输入的数据不做任何要求，则仅能通过比较来确定输入序列<a1,a2,…,an>各元素间的顺序。**

**无论算法采用怎样的比较策略/顺序，总能对应一个两两比较序列，考察所有可能则可对应一棵决策树。例如：**

**a1:a2**

**(<=) / \(>)**

**(a1a2) (a2a1)**

**树的非叶子结点表示一次比较，叶子结点对应一个可能的结果队列。**

**显然树高度为比较次数。**

**可以为任意的输入，叶子结点数目为n!**

**高度最小情况为满二叉树，则2h =n！**

**一般情况下：2h>=n!, 则h>=log2n!>log2(n/e)n=nlogn-nloge**

**则任意分布数据，算法复杂性下界：Ω(nlogn)**

二、常用基本算法及思想

名次排序、冒泡排序、选择排序、插入排序、快速排序、归并排序、堆排序。

1.名次排序

**（1）计算名次**

void Rank(T a[], int n, int r[])

{ //计算a [0:n-1]中n个元素的排名

for (int i = 0; i < n; i++)

r[i] = 0; //初始化

//逐对比较所有的元素

for (int i = 1; i < n; i++)

for ( int j = 0; j < i; j++)

if (a [j] <= a[ i]) r[i]++;

else r[j ]++;

}

**（2）按名次排序**

void Rearrange (T a[], int n, int r[])

{ //按序重排数组a中的元素，使用附加数组u

T \*u = new T[n+1];

//在u中移动到正确的位置

for (int i = 0; i < n; i++)

u[r[i]] = a[i];

//移回到a中

for (int i = 0; i < n; i++)

a[i] = u[i] ;

delete [ ]u;

}

**2.冒泡排序**

（1）采用一种“冒泡策略”把最大元素移到右部。在冒泡过程中，对相邻的元素进行比较，如果左边的元素大于右边的元素，则交换这两个元素。

（2）在一次冒泡过程结束后，可以确信最大的元素肯定在最右边的位置上。

**3.选择排序**

思想：首先找出最大的元素，把它移动到a[n-1]，然后在余下的n-1个元素中寻找最大的元素并把它移动到a[n-2]，如此进行下去。

**4.插入排序**

（1）因为只有一个元素的数组是一个有序数组，所以可以从包含n个元素的数组的第一个元素开始。通过把第二个元素插入到这个单元数组中，可以得到一个大小为2的有序数组。插入第三个元素可以得到一个大小为3的有序数组。

（2）按照这种方法继续进行下去，最终将得到一个大小为n的有序数组。

**5.堆排序**

（1）将要排序的*n*个元素初始化为一个最大(小)堆.

（2）每次从堆中提取（即删除）元素。

* 如果使用最大堆,各元素将按非递减次序排列。
* 如果使用最小堆,各元素将按非递增次序排列。

**6.归并排序**

将n 个元素按非递增顺序排列.

若*n* 为1，算法终止；

否则，将这一元素集合分割成两个或更多个子集合，对每一个子集合分别排序，然后将排好序的子集合归并为一个集合。

**7.快速排序**

（1）n 个元素被分成三段（组）：左段left，右段right和中段middle。中段仅包含一个元素。

（2）左段中各元素都小于等于中段元素，右段中各元素都大于等于中段元素。因此left和right中的元素可以独立排序，并且不必对left和right的排序结果进行合并。

（3）middle中的元素被称为支点(pivot)。

**8.箱子排序**

**同数值数据先放入一个箱子中。（与数据范围相关）**

**9.基数排序**

**按基数多遍。（与数据范围相关）**

三、复杂度及分析

|  |  |  |
| --- | --- | --- |
| 方法 | 最坏复杂性 | 平均复杂性 |
| 冒泡排序 | n2 | n2 |
| 基数排序 | n | n |
| 插入排序 | n2 | n2 |
| 选择排序 | n2 | n2 |
| 堆排序 | nlogn | nlogn |
| 归并排序 | nlogn | nlogn |
| 快速排序 | n2 | nlogn |

* 快速排序时间复杂性分析

（1）最坏情况

O（n2）

（2）最好情况

O(nlog2n)

（3）平均情况

O（nlog2n）

设t(n) 为分而治之排序算法在最坏情况下所需花费的时间.

d n<k

t(n)=

t(n/k)+t(n-n/k)+cn n≥k



可归纳证明。

四、习题及解答

**应掌握：**

**算法思路----〉能举出例子**

**复杂度 ----〉不要求严格分析，但能知道结果**

**特征：是否稳定排序、与分布**

**其他：归并顺序对复杂度的影响。**

分组做，3-5人一组，组内提出存在问题。