**排序课堂纪要**

written by王保明

# 排序基本概念

|  |
| --- |
| 现实生活中排序很重要 |
| 算法已写好代码复用 & 和我们需要学习前辈们的经验 不矛盾，也不代表我们不需要不学习。 |
| 排序是计算机内经常进行的一种操作，其目的是将一组“无序”的数据元素调整为“有序”的数据元素。 |
| 排序数学定义：假设含n个数据元素的序列为{ R1, R2, …, Rn}  其相应的关键字序列为{ K1, K2, …, Kn}这些关键字相互之间可以进行比较，即在它们之间存在着这样一个关系 ：Kp1≤Kp2≤…≤Kpn  按此固有关系将上式记录序列重新排列为{ Rp1, Rp2, …，Rpn}的操作称作排序 |
| 排序的稳定性 |
| 如果在序列中有两个数据元素r[i]和r[j]，它们的关键字k[i] == k [j]，且在排序之前，对象  r[i]排在r[j]前面。如果在排序之后，对象r[i]仍在r[j]前面，则称这个排序方法是稳定的；  否则称这个排序方法是不稳定的。 |
| 多关键字排序 |
| 排序时需要比较的关键字多余一个  排序结果首先按关键字1进行排序  当关键字1相同时按关键字2进行排序  当关键字n-1相同时按关键字n进行排序  对于多关键字排序，只需要在比较操作时同时考虑多个关键字即可！ |
| 排序中的关键操作 |
| 比较  任意两个数据元素通过比较操作确定先后次序  交换  数据元素之间需要交换才能得到预期结果 |
| 内排序和外排序 |
| 内排序  整个排序过程不需要访问外存便能完成  外排序  待排序的数据元素数量很大，整个序列的排序过程不可能在内存中完成 |
| 排序的审判 |
| 时间性能  关键性能差异体现在比较和交换的数量  辅助存储空间  为完成排序操作需要的额外的存储空间  必要时可以“空间换时间”  算法的实现复杂性  过于复杂的排序法会影响代码的可读性和可维护性，也可能影响排序的性能 |
| 总结  排序是数据元素从无序到有序的过程  排序具有稳定性，是选择排序算法的因素之一  比较和交换是排序的基本操作  多关键字排序与单关键字排序无本质区别  排序的时间性能是区分排序算法好坏的主要因素 |
|  |
|  |

# 选择法

|  |
| --- |
| 基本思想 |
| 每一趟 (例如第 i 趟，i = 0, 1, …,n-2)在后面 n-i个待排的数据元素中选出关键字  最小的元素, 作为有序元素序列的第 i 个元素。 |
|  |
| 排序过程  首先通过n-1次关键字比较，从n个记录中找出关键字最小的记录，将它与第一个记录交换  再通过n-2次比较，从剩余的n-1个记录中找出关键字次小的记录，将它与第二个记录交换  重复上述操作，共进行n-1趟排序后，排序结束 |
|  |
|  |

# 插入排序

|  |
| --- |
| 基本思想：  元素1个元素， |
| 排序过程：整个排序过程为n-1趟插入，即先将序列中第1个记录看成是一个有序子序列，然后从第2个记录开始，逐个进行插入，直至整个序列有序  实质：对线性表执行n-1次插入操作，只是先要找到插入位置 |
| V[0], V[1], …, V[i-1]已经排好序。这时已经排好序。这时,用V[i]的关键字与V[i-1], V[i-2], …的关键字进行比较, 找到插入位置即将V[i]]插入, 原来位置上的对象向后顺移。  插入排序关键点：1、拿出一个元素，留出位置、2 符合条件的元素后移 |
|  |
|  |

# 冒泡排序

|  |
| --- |
|  |
|  |
|  |

# 希尔排序

|  |
| --- |
| 排序过程：先取一个正整数d1<n，把所有相隔d1的记录放一组，组内进行直接插入排序；然后取d2<d1，重复上述分组和排序操作；直至di=1，即所有记录放进一个组中排序为止  O(n-1.3)  Q(nlogn)  希尔排序是不稳定的。 |
|  |
|  |

# 快速排序

|  |
| --- |
| 思想：  快速排序是对冒泡排序的一种改进。它的基本思想是：  通过一躺排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一不部分的所有数据都要小，基准数据排在这两个子序列的中间；  然后再按次方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。 |
|  |
|  |
|  |
| // O(n\*logn)  不稳定，分组，后面的有可能跑到前面去了。  21 100 3 50 1 |

# 归并排序

|  |
| --- |
| 思想： |
| 注意：一个元素，可以看做有序的，是稳定的算法 |
|  |
|  |
|  |
| 对一个数组分成两路，mid中间 |
|  |
| 设两个有序的子文件(相当于输入堆)放在同一向量中相邻的位置上：R[low..m],R[m+1..high],先将它们合并到一个局部的暂存向量R1(相当于输出堆)中,待合并完成后将R1复制回R[low..high]中。 |
|  |
|  |
|  |

# 排序总结

|  |
| --- |
|  |