第10章 排序

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

排序的分类

[根据排序时文件记录的存放位置]

内部排序：排序过程中将全部记录放在内存中处理。

外部排序：排序过程中需在内外存之间交换信息。

[根据排序前后相同关键字记录的相对次序]

稳定排序：设文件中任意两个记录的关键字值相同，即$K_i=K_j(i\neq j)$，若排序之前记录R_i领先于记录R_j，排序后这种关系不变(对所有输入实例而言)。

不稳定排序：只要有一个实例使排序算法不满足稳定性要求。

几种常见的插入排序

- 根据往**已经排好序的有序数据表中插入**的不同方法，可以将插入排序分为：
  - 直接插入排序
  - 二分插入排序
  - 希尔排序

直接插入排序

- 假定待排序的是数组a，它共有n个数组元素。那么对于每个1<=j<n的j，将a[j]插入到已经排好序的序列a[0]，a[1], a[2],……, a[j-1]中去。
- 首先将a[j]放入到一个某个变量tmp中。然后从右到左与a[j-1]……, a[2], a[1], a[0]，进行比较。
  - 若tmp < a[j-1]，将a[j-1]的内容移到a[j]中去
  - 若tmp < a[j-2]，将a[j-2]的内容移到a[j-1]中去
  - 重复这个过程，直至找到不大于tmp的数组元素a[k]，将tmp存入a[k+1]。
  - 如果一直找到a[0]都没有找到一个不大于tmp的值，将tmp存入a[0]。

直接插入排序（增量法）

[示例]　{ R(0)　R(-4)　R(8)　R(1)　R(-4)　R(-6) } n=6

i=1	[0]	-4	8	1	-4	-6
i=2	[-4	0]	8	1	-4	-6
i=3	[-4	0	8]	1	-4	-6
i=4	[-4	0	1	8]	-4	-6
i=5	[-4	-4	0	1	8]	-6
i=6	[-6	-4	-4	0	1	8]

稳定排序

[算法思想]　每次使有序区增加一个记录

直接插入排序算法

```
template <class KEY, class OTHER>
void simpleInsertSort(SET<KEY, OTHER> a[], int size)
{    int k;
     SET<KEY, OTHER> tmp;
     for (int j=1; j<size; ++j) {
          tmp = a[j];
          for ( k = j-1; tmp.key < a[k].key && k >= 0; --k)
              a[k+1] = a[k];
          a[k+1] = tmp;
     }
}
```

时间复杂度分析

- 因为简单插入排序是两层循环嵌套，因此时间复杂度是$O(N^2)$的。
- **该上限是可达的。当输入是逆序的时候，达到上限。**
- **当输入已排序时，运行时间是O(N)。因为内层循环只执行一个周期**
- 适用情况：排序元数较少，且几乎是已排序的

二分插入排序

- 利用二分查找法，快速地找到a[j]的插入位置。从而达到减少比较次数的目的
- 最坏情况下总的比较次数还是O（n^2）

希尔排序

对待排记录序列先作“宏观”调整，再作“微观”调整。

[算法思想的出发点]

- 直接插入排序在待排序列的关键字基本有序时，效率较高
- 在待排序的记录个数较少时，效率较高

[算法思想]

先选定一个记录下标的增量d，将整个记录序列按增量d从第一个记录开始划分为若干组，对每组使用直接插入排序的方法；然后减小增量d，不断重复上述过程，如此下去，直到d=1(此时整个序列是一组)。

希尔排序

- 希尔排序是插入排序算法的改进
- 直接插入排序的高代价主要由大量的移动产生
- 希尔的想法是避免大量的数据移动，先是比较那些离得稍远些的元素，这样，一次交换就相当于直接插入排序中的多次交换。然后比较那些离得近一点的元素，以此类推，逐步逼近直接的插入排序

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

希尔排序算法思想

- 设待排序的对象序列有n个对象，
- 首先取一个整数gap < n作为增量，
- 将全部对象分为gap个子序列，
- 所有距离为gap的对象放在同一个序列中，
- 在每一个子序列中分别施行直接插入排序，
- 然后缩小增量gap，
- 重复上述的子序列划分和排序工作，直到最后取gap为1为止。

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

[示例]

<初态> **46 82 52 40 67 31 40 73 53 76**

d1=5

31 46
40 82
52 73
40 53
67 76

<第一趟结果> **31 40 52 40 67 46 82 73 53 76**

d2=3

31 40 76 82
40 67 73
46 52 53

<第二趟结果> **31 40 46 40 67 52 76 73 53 82**

d3=1

<第三趟结果> 31 40 40 46 52 53 67 73 76 82

不稳定排序

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

选择排序

首先，从n个元素中选出关键字最小的元素。再从剩下的(n-1)个元素中选出关键字最小的元素，依次类推，每次从剩下的元素序列中挑出关键字最小的元素，直至序列中最后只剩下一个元素为止。这样，把每次得到的元素排成一个序列，就得到了按非递减序排列的排序序列。

- 直接选择排序
- 堆排序

直接选择排序

- 首先在所有元素中用逐个比较的方法选出最小元素，把它与第一个元素交换；然后在剩下的元素中再次用逐个比较的方法选出最小元素，把它与第二个元素交换；以此类推，直到所有元素都放入了正确的位置。
- 对k个元素而言，每次选出最小元素需要k-1次比较。因此排序一个n个元素组成的序列所需的比较次数为：

$$(n-1) + (n-2) + \ldots\ldots + 2 + 1$$

$$= n(n-1)/2 = O(n^2)$$

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

直接选择排序

[算法思想]

每次从待排序列中选出关键字最小记录作为有序序列中最后一个记录，直至最后一个记录为止。

[示例]（n=8）

<初态>	49	38	65	97	76	49	13	27
<第1趟>	13	38	65	97	76	49	49	27
<第2趟>	13	27	65	97	76	49	49	38
<第3趟>	13	27	38	97	76	49	49	65
<第4趟>	13	27	38	49	76	97	49	65
<第5趟>	13	27	38	49	49	97	76	65
<第6趟>	13	27	38	49	49	65	76	97
<第7趟>	13	27	38	49	49	65	76	97

（每趟排序使有序区增加一个记录）

不稳定排序

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

初始时	7	9	3	8	5	0	6
1	0	9	3	8	5	7	6
2	0	3	9	8	5	7	6
3	0	3	5	8	9	7	6
4	0	3	5	6	9	7	8
5	0	3	5	6	7	9	8
6	0	3	5	6	7	8	9

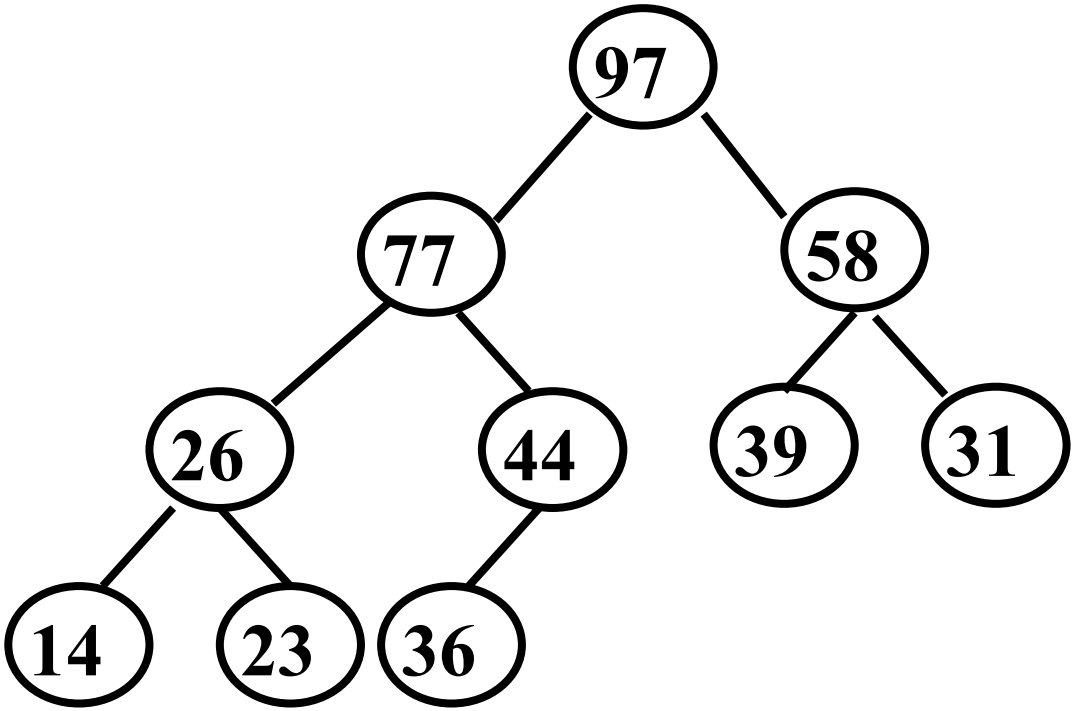
堆排序

- 直接选择排序在n个元素中选出最小元素需要n-1次比较。而利用基于堆的优先级队列选出最小元素只需要O(logN)的时间
- 排序N个元素，步骤如下：
  - 应用buildHeap对N个元素创建一个优先级队列
  - 通过调用N次deQueue取出每个项，结果就排好序了。
- 时间效益：建堆用了O(N)的时间，deQueue是对数时间，一共deQueue了N次。因此总的时间是O(NlogN)。

实现中的主要问题

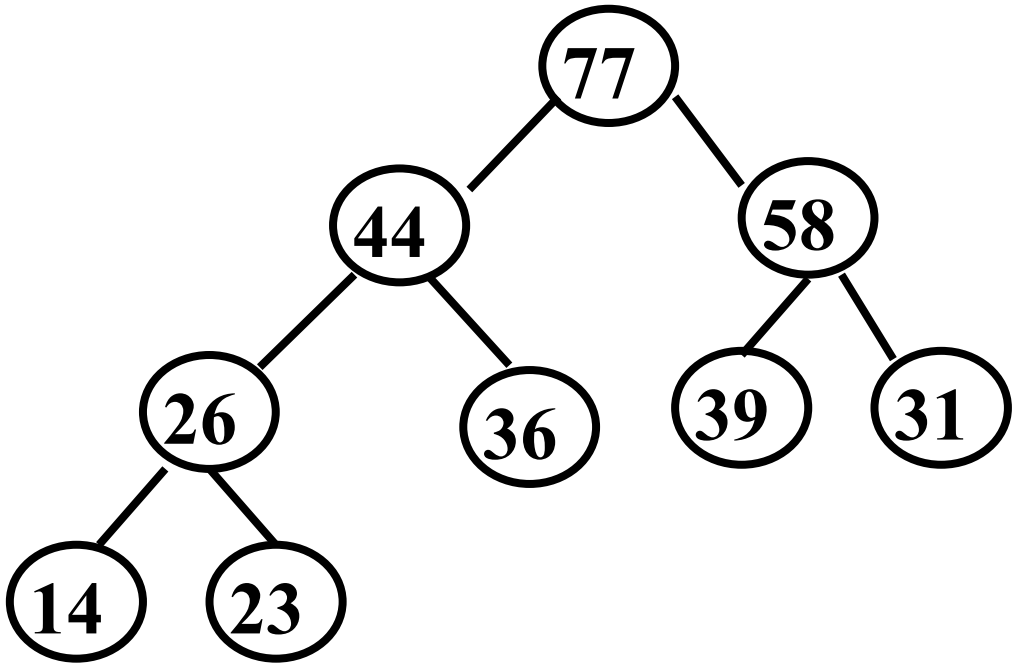
- 空间问题：
  - N元素的优先级队列需要N个空间，每次deQueue的元素也要放在一个空间中。因此需要两倍的空间。
- 解决方案：在每一次deQueue后，堆缩小1。因此，在堆中的最后那个位置能被用来存储刚被删去的元素。
- 假设有一个六个元素的堆。第一次deQueue产生了A1。现在堆中仅有五个元素，我们可以把A1放在位置6。下一个deQueue产生A2。因为堆中现在仅有四个元素，我们可以把A2放在位置5。
- 为了产生递增排序，可以采用最大堆。

例如，要排序39, 36, 58, 23, 44, 97, 31, 14, 26和77，先将这些元素创建一个堆



97	77	58	26	44	39	31	14	23	36
0	1	2	3	4	5	6	7	8	9

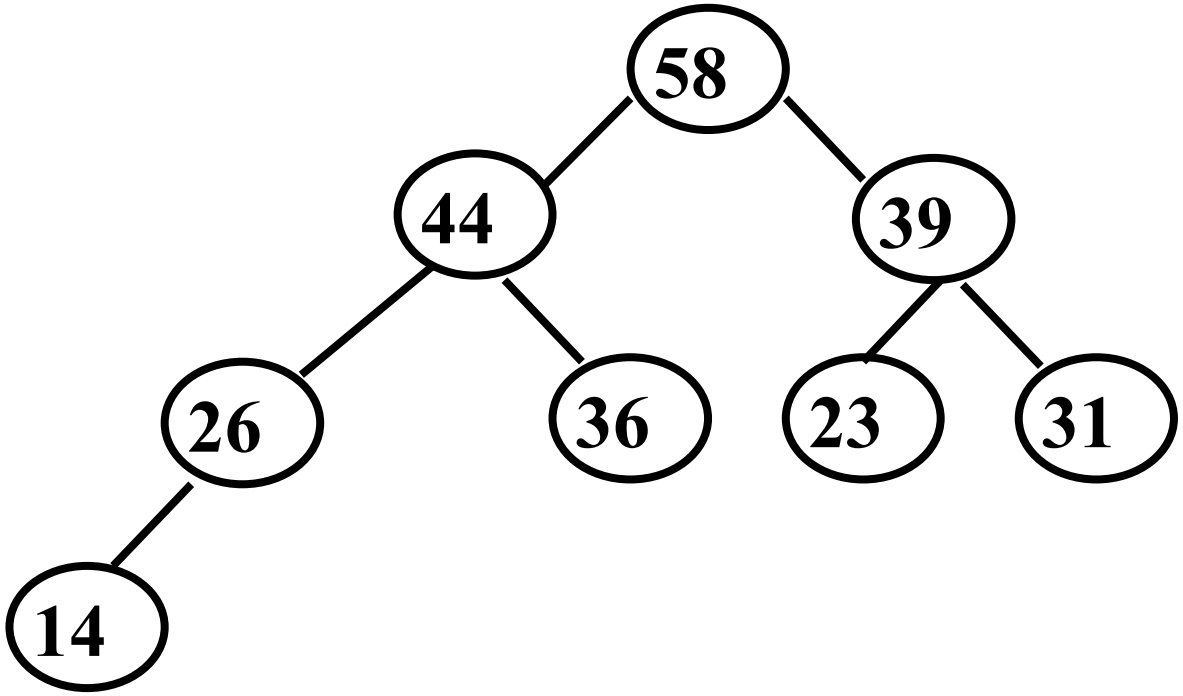
执行一次deQueue



77	44	58	26	36	39	31	14	23	97
0	1	2	3	4	5	6	7	8	9

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

再执行一次deQueue



58	44	39	26	36	23	31	14	77	97
0	1	2	3	4	5	6	7	8	9

1. 设一组初始记录关键字序列为（50, 40, 95, 20, 15, 70, 60, 45)，则以增量 d=4 的一趟希尔排序结束后前 4 条记录关键字为_____

A. 40, 50, 20, 95　　B. 15, 40, 60, 20

C. 15, 20, 40, 45　　D. 45, 40, 15, 20

2.一组记录的关键字序列为（33, 66, 43, 25, 27, 71），为排成非递减序列利用堆排序的方法建立的初始堆为_____

A. (66, 33, 43, 25, 27, 71)　　B. (71, 66, 43, 33, 27, 25)

C. (71, 66, 43, 25, 27, 33)　　D. (71, 43, 66, 27, 33, 25)

交换排序

- 交换排序就是根据序列中两个数据元素的比较结果来确定是否要交换这两个数据元素在序列中的位置。
- 交换排序的特点是：通过交换，将关键字值较大的数据元素向序列的尾部移动，关键字值较小的数据元素向序列的头部移动。

冒泡排序

快速排序

冒泡排序

- 从头到尾比较相邻的两个元素，将小的换到前面，大的换到后面。经过了从头到尾的一趟比较，就把最大的元素交换到了最后一个位置。这个过程称为一趟起泡。
- 然后再从头开始到倒数第二个元素进行第二趟起泡。经过了第二趟比较，又将第二大的元素放到了倒数第二个位置。
- 依次类推，经过第n-1趟起泡，将倒数第n-1个大的元素放入第2个单元。

[算法思想]

将两个相邻记录的关键字进行比较，若为逆序则交换两者位置，小者往上浮，大者往下沉。

[算法步骤]

记录1和2、2和3、……、(n-1)和n的关键字比较(交换)；

记录1和2、2和3、……、(n-2)和(n-1)的关键字比较(交换)；

……

直到某一趟不出现交换操作为止。

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

初始时	7	9	3	8	5	0	6
1	7	3	8	5	0	6	9
2	3	7	5	0	6	8	9
3	3	5	0	6	7	8	9
4	3	0	5	6	7	8	9
5	0	3	5	6	7	8	9
6	0	3	5	6	7	8	9

冒泡排序的时间性能

- 当原始序列有序时，冒泡排序出现最好的情况。此时，只需要一次起泡，执行n-1次的比较。因此，最好情况下的时间复杂度为O（N）。
- 当原始序列正好是逆序时，冒泡排序出现最坏的情况。此时，需要n-1趟起泡，第i趟起泡需要做（n-i）次比较以及（n-i）次交换。所以时间复杂度是O（N^2）的。
- 冒泡排序法适合那些原始数据本来就比较有序的情况。

冒泡排序是稳定排序

快速排序

[分治算法原理]

1) 分解：将原问题分解为若干子问题

2) 求解：递归地解各子问题，若子问题的规模足够小，则直接求解

3) 组合：将各子问题的解组合成原问题的解

[快速排序算法思想]

指定**枢轴/支点/基准记录**r[p](通常为第一个记录)，通过一趟排序将其放在正确的位置上，它把待排记录分割为独立的两部分，使得

左边记录的关键字 ≤ r[p].key ≤ 右边记录的关键字

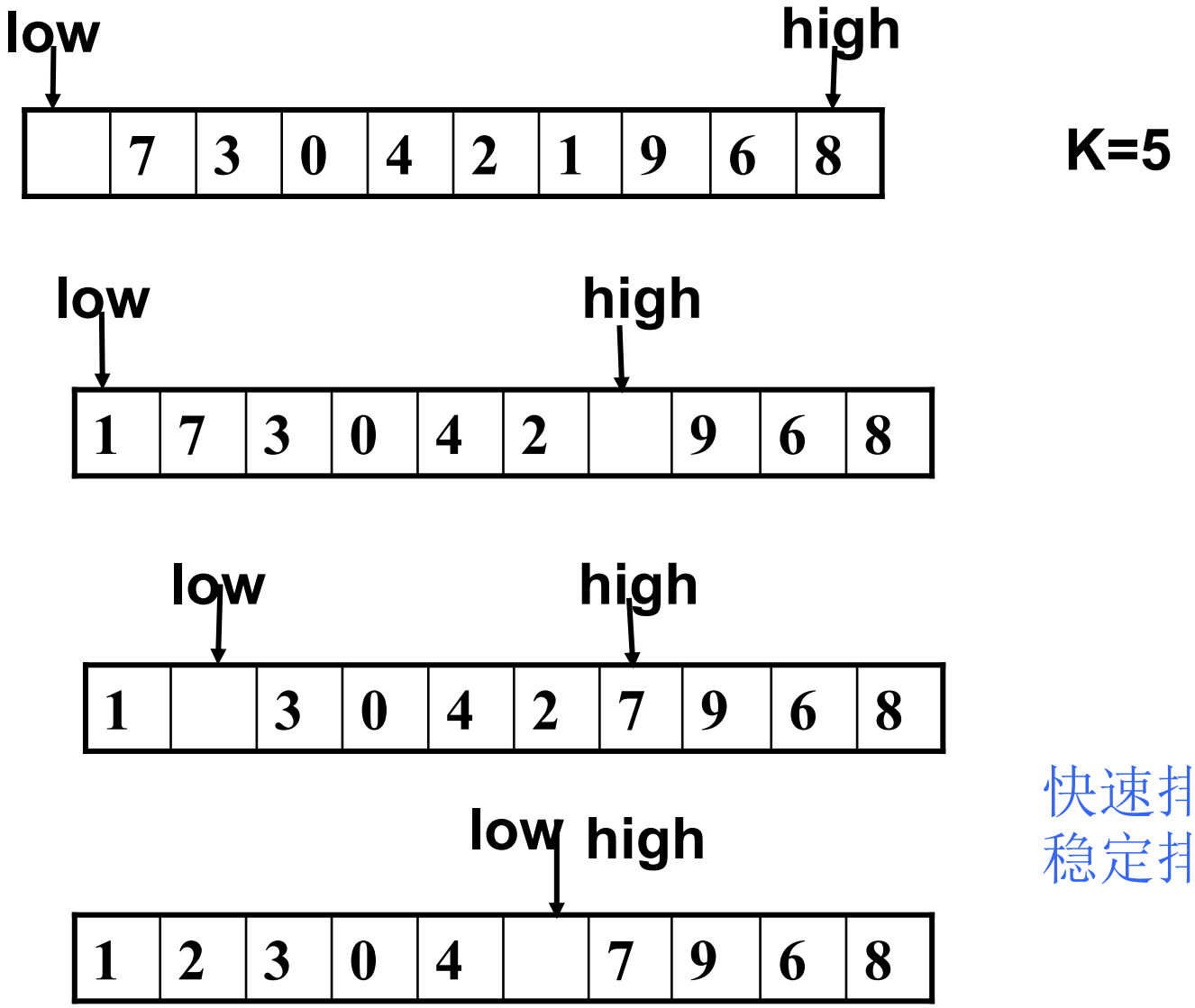
对左右两部分记录序列重复上述过程，依次类推，直到子序列中只剩下一个记录或不含记录为止。(可以用递归方法实现)

如何划分

- 从右向左开始检查。如果high的值大于k，该位置中的值位置正确，high减1，继续往前检查，直到遇到一个小于k的值。
- 将小于k的这个值放入low的位置。此时high的位置又空出来了。然后从low位置开始从左向右检查，直到遇到一个大于k的值。
- 将low位置的值放入high位置，重复第一步，直到low和high重叠。将k放入此位置。

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

5，7，3，0，4，2，1，9，6，8的划分步骤



快速排序是不稳定排序

划分函数的实现

```
template <class KEY, class OTHER>
int divide( SET<KEY, OTHER> a[], int low, int high)
{
    SET<KEY, OTHER>  k = a[low];
    do  {
        while (low < high && a[high].key >= k.key) --high;
        if (low < high) { a[low] = a[high]; ++low;}
        while (low < high && a[low].key <= k.key) ++low;
        if (low < high)  {a[high] = a[low]; --high;}
    } while (low != high);
    a[low] = k;
    return low;
}
```

快速排序的实现

- 执行一次划分，在执行两个递归
- 快速排序是用递归的方式实现的，它的递归参数就是排序的范围

```
template <class KEY, class OTHER>
void quickSort(SET<KEY, OTHER> a[], int low, int high)
{
    int mid;
    if (low >= high) return;
    mid = divide(a, low, high);
    quickSort( a, low, mid-1);//排序左一半
    quickSort( a, mid+1, high);//排序右一半
}
```

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

归并排序

归并的思想来源于合并两个已排序的有序表

实现方法：顺序比较两者的相应元素，小者移入另一表中，反复如此，直至其中一表为空为止，将另一表中剩余结点自左至右复制到表C的剩余位置。

[归并的概念]

指将两个或两个以上的同序序列归并成一个序列的操作。

两路归并排序

[算法思想]

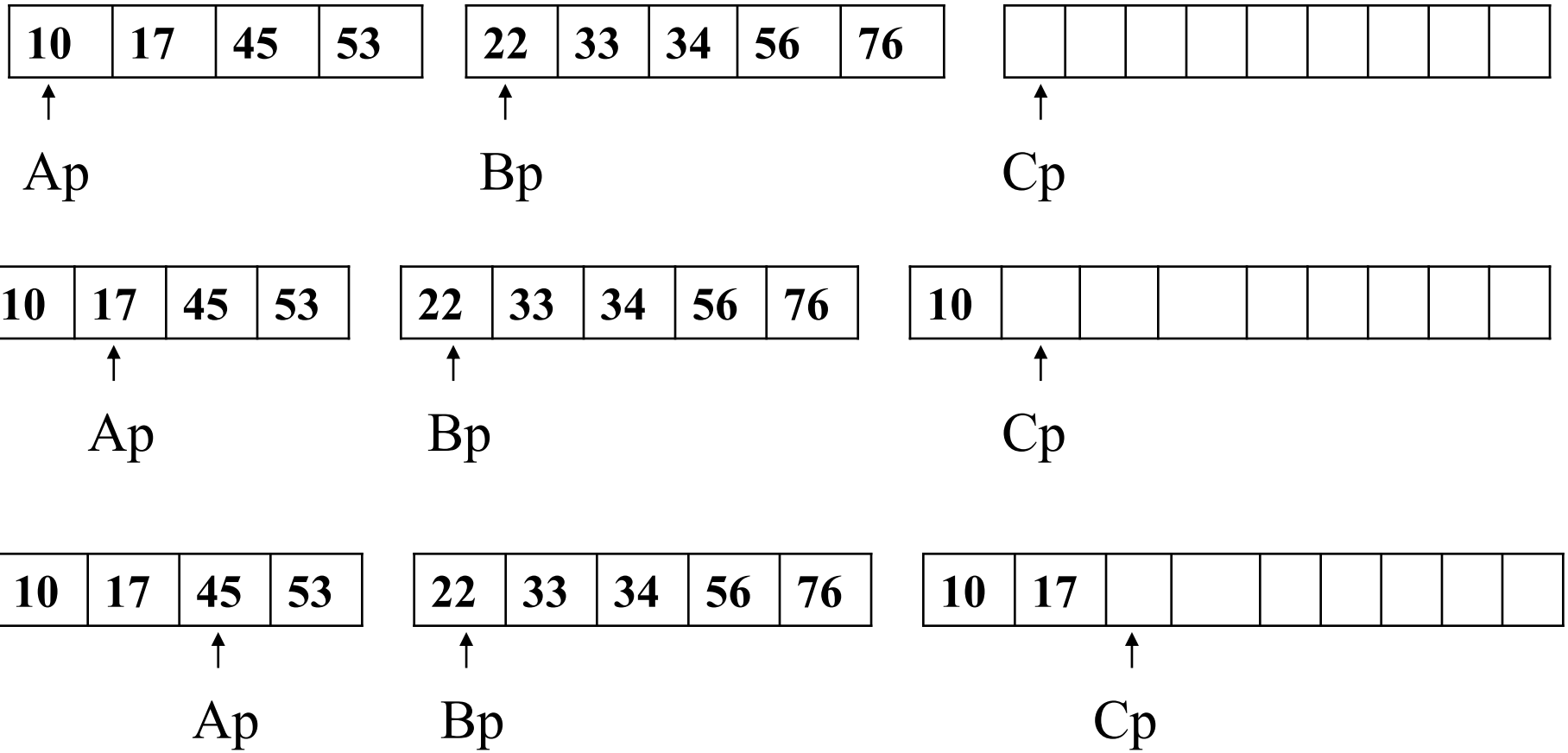
第1趟：将待排序列R[1..n]看作n个长度为1的有序子序列，两两归并，得到$\lceil n/2 \rceil$个长度为2的有序子序列(或最后一个子序列长度为1)；

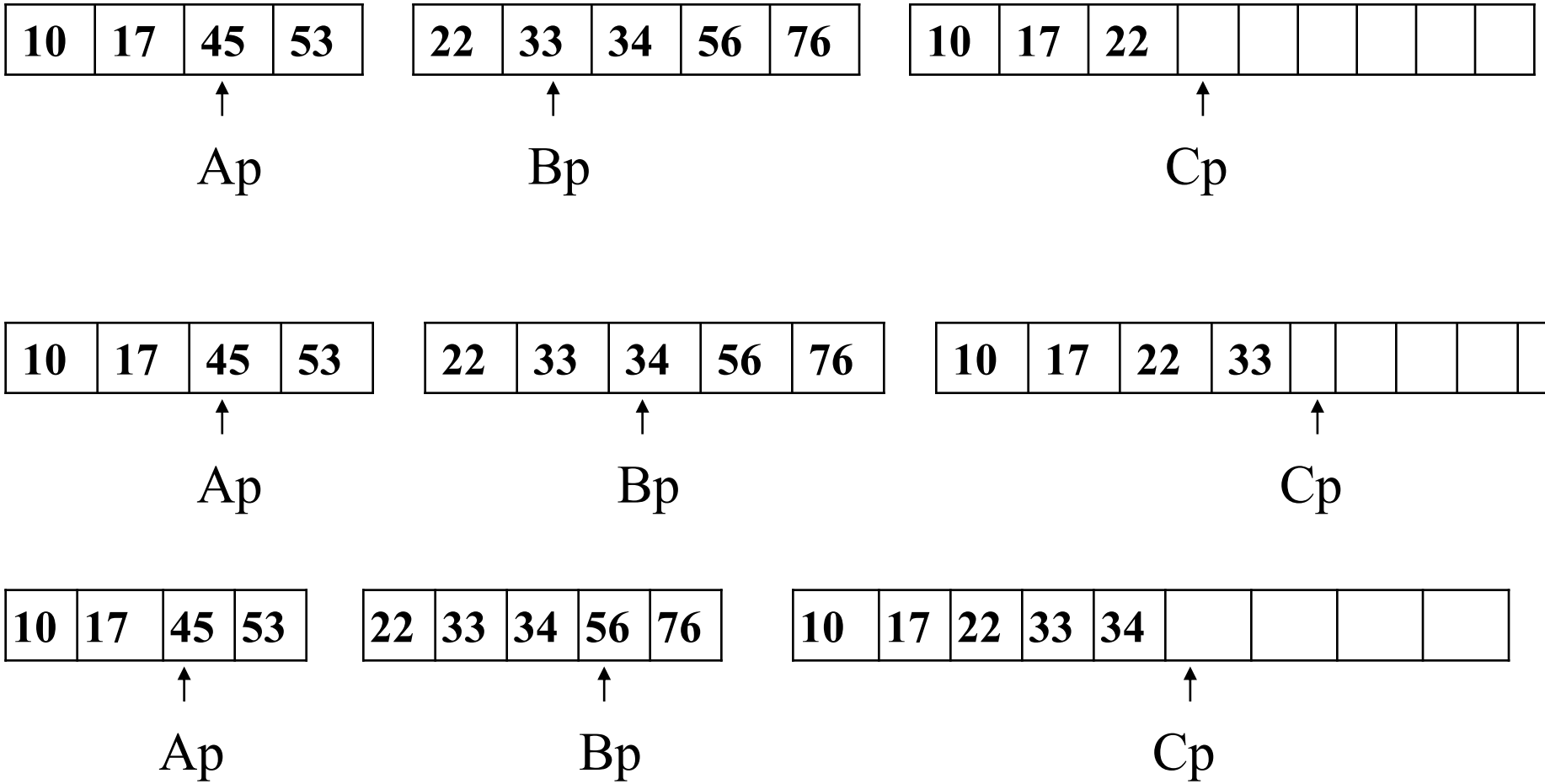
第2趟：将上述$\lceil n/2 \rceil$个有序子序列两两归并；

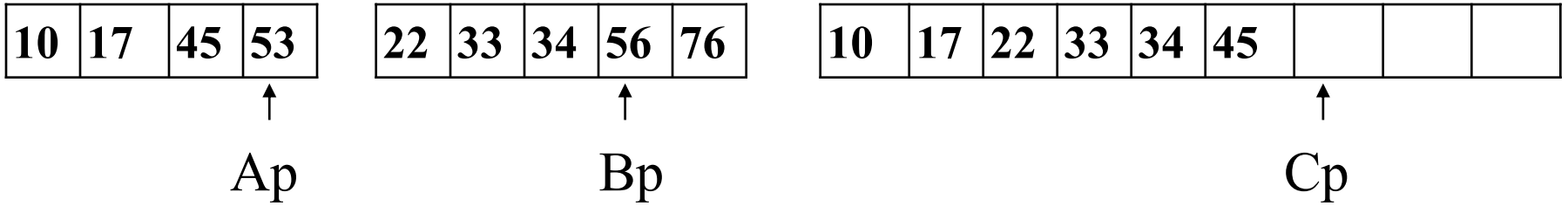
……

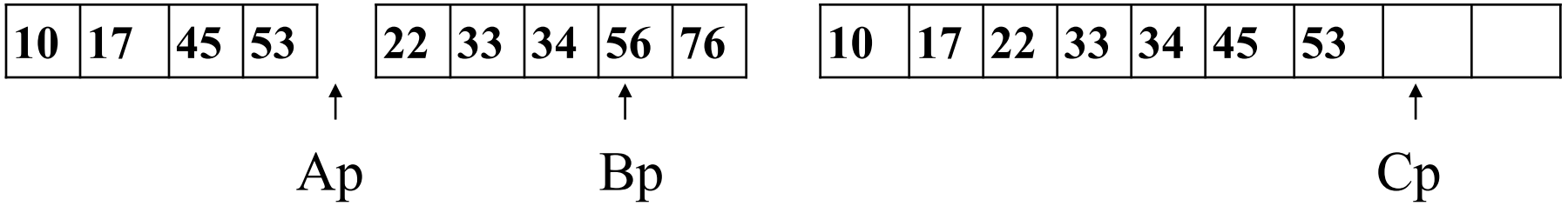
直到合并成一个序列为止。

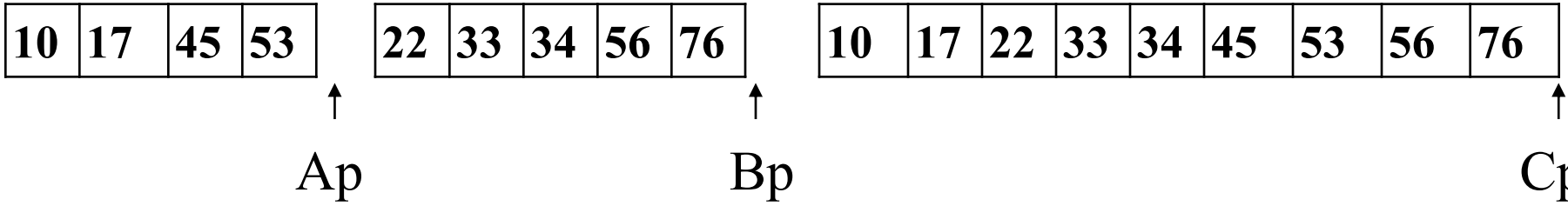
合并实例


10 17 45 53
22 33 34 56 76
Ap
Bp
Cp
10 17 45 53
22 33 34 56 76
10
Ap
Bp
Cp
10 17 45 53
22 33 34 56 76
10 17
Ap
Bp
Cp


10 17 45 53
Ap
22 33 34 56 76
Bp
10 17 22
Cp
10 17 45 53
Ap
22 33 34 56 76
Bp
10 17 22 33
Cp
10 17 45 53
Ap
22 33 34 56 76
Bp
10 17 22 33 34
Cp


10 17 45 53
22 33 34 56 76
10 17 22 33 34 45
Ap
Bp
Cp


10 17 45 53
22 33 34 56 76
10 17 22 33 34 45 53
Ap
Bp
Cp


10 17 45 53
22 33 34 56 76
10 17 22 33 34 45 53 56 76
Ap
Bp
Cp

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

[示例] (n=7)

<初态>	(49)	(38)	(65)	(97)	(76)	(49)	(13)
<第1趟>	(38	49)	(65	97)	(49	76)	(13)
<第2趟>	(38	49	65	97)	(13	49	76)
<第3趟>	(13	38	49	49	65	76	97)

稳定排序

[性能分析]

- 任何情况时间复杂度$O(n\log_2 n)$
- 空间复杂度O(n)
- 很少用于内部排序

基数排序

[特点] 通过“分配”和“收集”过程来实现排序，时间复杂度可以突破基于关键字比较一类方法的下界O(nlgn)，达到O(n)。

[方法] 借助多关键字排序的思想对单关键字排序

[多关键字排序示例] 扑克牌排序(点数+花色)

双关键字 花色：♣<♦<♥<♠

点数：2<3<…<Q<K<A

“花色”地位高于“面值”

最高位优先法（MSD法, Most Significant Digit first）

按花色分成4堆；

对每一堆：按面值分堆；

从小到大排序；

按花色从小到大排序

最低位优先法（ LSD法, Least Significant Digit first）

按点数大小分成13堆；
按点数从小到大收集起来；
再按花色分成四堆；
按花色从小到大收集起来

[MSD与LSD不同特点]

- 按MSD排序，必须将序列逐层分割成若干子序列，然后对各子序列分别排序
- 按LSD排序，不必分成子序列，对每个关键字都是整个序列参加排序；并且可不通过关键字比较，而通过若干次分配与收集实现排序

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

[基数排序算法思想] 借鉴LSD法

设参加排序的序列为$K=\{K_1, K_2, \ldots, K_n\}$，

其中K_i是d位rd进制的数，**rd**称为**基数**；

d由所有元素中最长的一个元素的位数计量，

$K_i = K_i^0 K_i^1 \ldots K_i^{d-1}$

高 ← 低

从低位到高位依次对K^j(j=d-1, d-2, ..., 0)根据基数分配，再按基数递增序收集，则可得有序序列。

[例]

(1) K={3621 0724 8385 0075 0514 7368 0008 }

rd=10, d=4

基数排序

- 又称为口袋排序法
- 通过分配的方法对整数进行排序
- 以排序十进制非负整数为例，可设置10个口袋
  - 首先将元素按个位数分别放入十个口袋，然后将每个口袋中的元素倒出来
  - 按元素的十位数分别放入十个口袋。然后把它们倒出来
  - 再按百位数分配
  - 到最后一次倒出来时，元素就已经排好了序

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

[链式基数排序示例] { 477 241 467 5 363 81 5 } **rd=10 d=3**

第1趟：分配 [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

241 363 5 477
81 5 467

收集 {241 81 363 5 5 477 467}

第2趟：分配 [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

5 241 363 477 81
5 467

收集 {5 5 241 363 467 477 81}

第3趟：分配 [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

5 241 363 467
5 477
81

收集 {5 5 81 241 363 467 477 } 稳定排序

总结

- 介绍了常用的内排序，包括插入排序、选择排序、交换排序、归并排序。具体使用哪种排序方法取决于待排序数据的规模和原来的次序。
- 直接插入、直接选择排序适合于非常少量的数据。希尔排序对中等的数据量是一个好选择。冒泡排序适合原来就较有次序的序列。归并排序、快速排序、堆排序都有O(NlogN)的性能。

各种内部排序方法的比较

排序方法	最好时间	平均时间	最坏时间	辅助空间	稳定性
直接插入	O(n)	$O(n^2)$	$O(n^2)$	O(1)	√
希尔		$O(n^{1.3})$		O(1)	×
冒泡	O(n)	$O(n^2)$	$O(n^2)$	O(1)	√
快速	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$	×
简单选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	×
堆	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	O(1)	×
归并	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	O(n)	√
基数	O(d(rd+n))	O(d(rd+n))	O(d(rd+n))	O(1)	√

- 按平均时间排序方法分为四类

  $O(n^2)$、$O(n\lg n)$、$O(n^{1+\varepsilon})$、$O(n)$
- **快速排序是目前基于比较的内部排序中最好的方法**
- 关键字随机分布时，快速排序的平均时间最短，堆排序次之，但后者所需的辅助空间少
- **当n较小时**如($n<50$)，可采用直接插入或简单选择排序，前者是稳定排序，但后者通常记录移动次数少于前者
- 当n较大时，应采用时间复杂度为$O(n\lg n)$的排序方法(主要为快速排序和堆排序)或者基数排序的方法
- **当n较大时**，为避免顺序存储时大量移动记录的时间开销，可考虑用链表作为存储结构（如插入排序、归并排序、基数排序）

- **文件初态基本按正序排列时，应选用直接插入、冒泡**
- 选择排序方法应综合考虑各种因素

1. 时间复杂度不受数据初始状态影响而恒为$O(Nlog_2N)$的是（**A**）

A. 堆排序　　B.冒泡排序　　C. 希尔排序　　D.快速排序

2. 设一组初始关键字记录关键字为（20,15,14,18,21,36,40,10）,则以20为基准记录的一趟快速排序结束后的结果为（**A**）。

A. 10,15,14,18,20,36,40,21　　B. 10,15,14,18,20,40,36,21
C. 10,15,14,20,18,40,36,21　　D. 15,10,14,18,20,36,40,21

3. 下列排序算法中，（**B**）是稳定的。

A. 插入排序、希尔排序和冒泡排序　　B. 归并排序、冒泡排序与插入排序
C. 选择排序、归并排序与堆排序　　D. 插入排序、快速排序与堆排序

4. 简单插入排序、冒泡排序、快速排序、选择排序、归并排序、堆排序在平均情况下的时间复杂度分别为 $O(n^2), O(n^2), O(nlog_2n), O(n^2), O(nlog_2n), O(nlog_2n)$ _____。

5. 在快速排序、插入排序、堆排序中，如果初始数据基本接近于正序，则选用 插入排序 为好，如果初始数据接近于反序，则选用 堆排序 方法为好，如果初始无序，则选用 快速排序 方法。

6. 对序列 {98，36，-9，0，47，23，1，8，10，7} 采用希尔排序，下列序列（**A**）是增量为4的一趟排序结果。

A {10，7，-9，0，47，23，1，8，98，36}
B {-9，0，36，98，1，8，23，47，7，10}
C {36，98，-9，0，23，47，1，8，7，10}
D 以上都不对

7.用直接插入排序方法对下面四个序列进行排序（由小到大），元素比较次数最少的是（**C**）。

A. 94,32,40,90,80,46,21,69　　B. 32,40,21,46,69,94,90,80

C. 21,32,46,40,80,69,90,94　　D. 90,69,80,46,21,32,94,40

8. 数据序列F= {2，1，4，9，8，10，6，20}只能是下列排序算法中的（ A 两趟排序后的结果（从小到大）。

A 快速排序　　B 冒泡排序　　C 选择排序　　D 插入排序

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

排序

用下列排序方法对线性表{4,5,6,3,2,1}进行从小到大排序时，元素序列的变化情况为：

1）冒泡排序

第一趟：4,5,3,2,1,6

第二趟：4,3,2,1,5,6

第三趟: 3,2,1,4,5,6

第四趟：2,1,3,4,5,6

第五趟：1,2,3,4,5,6

补充：对n个不同的数进行冒泡排序，实现升序排序，什么情况下比较次数最多，什么情况下比较次数最少，最多要排序多少趟？

从大到小排序时比较次数最多（和目标顺序相反，**基本逆序**），**从小到大**排序好时比较次数最少（和目标顺序相同，**基本有序**）。最多要排序n-1趟

2）直接插入排序

初始：{4,5,6,3,2,1}

第一趟：4,5,6,3,2,1（将5插到{4}）

第二趟：4,5,6,3,2,1（将6插到{4,5}）

第三趟：3,4,5,6,2,1（将3插到{4,5,6}）

第四趟：2,3,4,5,6,1（将2插到{3,4,5,6}）

第五趟：1,2,3,4,5,6（将1插到{2,3,4,5,6}）

3）快速排序（选第一个数为基准）

初始：{4,5,6,3,2,1},

第一步（选4为基准，**从后向前**找第一个比4小的数，交换）**1**,5,6,3,2,_

第二步（**从前向后**找第一个比4大的数，交换）

1,_,6,3,2,**5**

第三步（**从后向前**找第一个比4小的数，交换）

1,**2**,6,3,_,5

第四步（**从前向后**找第一个比4大的数，交换）

1,2,_,3,**6**,5

第五步（**从后向前**找第一个比4小的数，两指针相遇，第一趟排序结束）1,2,**3**,**4**,6,5

第一趟结束：1,2,3,4,6,5

快速排序的运行时间与**划分是否对称**有关，当划分两个区域分别包含n-1个元素和0个元素时，达到最大程度的不对称。即对于基本有序或基本逆序，得到最坏情况下的时间复杂度为$O(n^2)$

4）希尔排序（增量为3）

初始：{4,5,6,3,2,1}

4,5,6,3,2,1

4,5,6,3,2,1

4,5,6,3,2,1

第一趟：3,2,1,4,5,6

第二趟，步长为1：1,2,3,4,5,6