排序

- 排序
  - 基本概念
    - 稳定性
    - 衡量标准：时、空间复杂度
  - 内部排序
    - 插入排序
      - 直接插入排序
      - 折半插入排序
      - 希尔排序
    - 交换排序
      - 冒泡排序
      - 快速排序
    - 选择排序
      - 简单选择排序
      - 堆排序
    - 归并排序
    - 基数排序
  - 外部排序 —— 多路归并排序

由于大纲中的要求，我们只学习内部排序中的插入、交换、选择、归并、基数排序。对外部排序的内容省略，考生有兴趣可自行了解。

目 录

Contents

排序

1 排序(Sorting)

排序是将一批(组)任意次序的记录重新排列成按关键字有序的记录序列的过程，其定义为：

给定一组记录序列：{R_1，R_2，…，R_n}，其相应的关键字序列是{K_1，K_2，…，K_n}。确定1, 2, … n的一个排列p_1，p_2，…，p_n，使其相应的关键字满足如下非递减(或非递增)关系： $K_{p_1} \le K_{p_2} \le \ldots \le K_{p_n}$的序列{$K_{p_1}$，$K_{p_2}$，…，$K_{p_n}$}，这种操作称为排序。

排序

2 排序的稳定性

若记录序列中有两个或两个以上关键字相等的记录： $K_i = K_j (i \neq j,\ i, j = 1, 2, \ldots n)$，且在排序前$R_i$先于$R_j$($i<j$)，排序后的记录序列仍然是$R_i$先于$R_j$，称排序方法是稳定的，否则是不稳定的。

排序算法有许多，但就全面性能而言，还没有一种公认为最好的。每种算法都有其优点和缺点，分别适合不同的数据量和硬件配置。

评价排序算法的标准有：执行时间和所需的辅助空间，其次是算法的稳定性。

若排序算法所需的辅助空间不依赖问题的规模n，即空间复杂度是O(1)，则称排序方法是就地排序，否则是非就地排序。

排序

3. 排序的分类

待排序的记录数量不同，排序过程中涉及的存储器的不同，有不同的排序分类。

① 待排序的记录数不太多：所有的记录都能存放在内存中进行排序，称为内部排序；

② 待排序的记录数太多：所有的记录不可能存放在内存中， 排序过程中必须在内、外存之间进行数据交换，这样的排序称为外部排序。

插入排序

插入排序是一种简单直观的排序方法。其基本思想在于每次将一个待排序的记录，按其关键字大小插入到前面已经排好序的子序列中，直到全部记录插入完成。

即在考察记录R_i之前，设以前的所有记录$R_1, R_2, \ldots, R_{i-1}$已排好序，然后将$R_i$插入到已排好序的诸记录的适当位置。

最基本的插入排序是直接插入排序(Straight Insertion Sort)。

插入排序

1 直接插入排序

1. 排序思想

将待排序的记录R_i，插入到已排好序的记录表$R_1, R_2, \ldots, R_{i-1}$中，得到一个新的、记录数增加1的有序表。 直到所有的记录都插入完为止。

设待排序的记录顺序存放在数组R[1…n]中，在排序的某一时刻，将记录序列分成两部分：

- R[1…i-1]：已排好序的有序部分；
- R[i…n]：未排好序的无序部分。

显然，在刚开始排序时，R[1]是已经排好序的。

直接插入排序是一种稳定的排序方法。

插入排序

例：设有关键字序列为：7, 4, -2, 19, 13, 6，直接插入排序的过程如下图所示：

插入排序

2. 性能分析

空间效率：仅使用了常数个辅助单元，因此空间复杂度为O(1)。

时间效率：

(1) 最好情况：表中元素已经有序，此时每插入一个元素都只需要比较一次而不用移动元素，因此时间复杂度为O(n)。

(2) 最坏情况：表中元素逆序，总的比较次数与总的移动次数达到最大。分别为：

比较次数：$\sum_{i=2}^{n} i = \frac{(n-1)(n+1)}{2}$

移动次数：$\sum_{i=2}^{n} (i+1) = \frac{(n-1)(n+4)}{2}$

一般地，认为待排序的记录可能出现的各种排列的概率相同，则取以上两种情况的平均值，作为排序的关键字比较次数和记录移动次数，约为$n^2/4$，则复杂度为$O(n^2)$。

插入排序

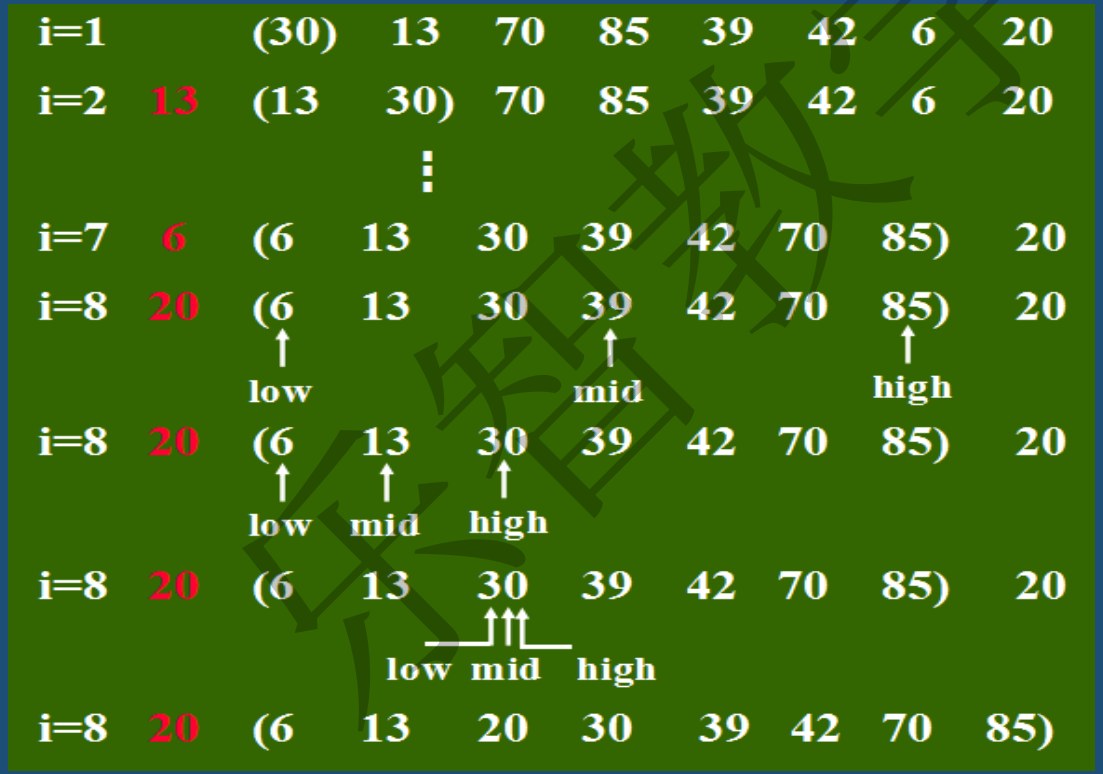
2 折半插入排序

1. 排序思想

当将待排序的记录R[i] 插入到已排好序的记录子表R[1...i-1]中时，由于$R_1, R_2, \ldots, R_{i-1}$已排好序，则查找插入位置可以用“折半查找”实现，则直接插入排序就变成为折半插入排序。

插入排序

例：设有一组关键字30, 13, 70, 85, 39, 42, 6, 20，采用折半插入排序方法排序的过程如图所示：



插入排序

2. 性能分析

从时间上比较，折半插入排序仅仅减少了关键字的比较次数，却没有减少记录的移动次数，故时间复杂度仍然为$O(n^2)$。虽然折半插入排序算法的时间复杂度也为$O(n^2)$，但对于数据量比较小的排序表，折半插入排序往往能表现出较好的性能。

折半插入排序是一种稳定的排序方法。

插入排序

3 希尔排序

1. 排序思想

① 先取一个正整数d_1(d_1<n)作为第一个增量，将全部n个记录分成d_1组，把所有相隔d_1的记录放在一组中，即对于每个k(k=1, 2, … d_1)，R[k], R[d_1+k], R[2d_1+k] , …分在同一组中，在各组内进行直接插入排序。这样一次分组和排序过程称为一趟希尔排序；

② 取新的增量d_2<d_1，重复①的分组和排序操作；直至所取的增量d_i=1为止，即所有记录放进一个组中排序为止。

由于希尔排序的时间复杂度依赖于增量序列的函数，这涉及数学上尚未解决的难题，所以其时间复杂度分析较为困难，当n在某个特定的范围时，希尔排序的时间复杂度约为0($n^{1.3}$)，在最坏情况下希尔排序的时间复杂度为$0(n^2)$

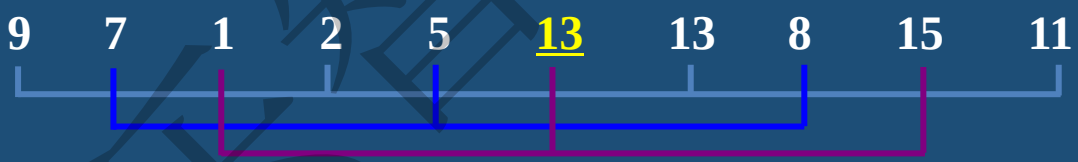
插入排序

例：设有10个待排序的记录，关键字分别为9, 13, 8, 2, 5, 13, 7, 1, 15, 11，增量序列是5, 3, 1，希尔排序的过程如图所示。

初始关键字序列：

第一趟排序过程：

第一趟排序后：



第二趟排序后：2 5 1 9 7 13 11 8 15 13

第三趟排序后：1 2 5 7 8 9 11 13 13 15

插入排序

2. 性能分析

希尔排序特点

子序列的构成不是简单的“逐段分割”，而是将相隔某个增量的记录组成一个子序列。

希尔排序可提高排序速度，原因是：

- 分组后n值减小，n^2更小，而$T(n)=O(n^2)$,所以T(n)从总体上看是减小了；
- 关键字较小的记录跳跃式前移，在进行最后一趟增量为1的插入排序时，序列已基本有序。

增量序列取法

- 无除1以外的公因子；
- **最后一个增量值必须为1。**

由例子可知，希尔排序是不稳定的。

交换排序

交换排序是一类基于交换的排序，系统地交换反序的记录的偶对，直到不再有这样一来的偶对为止。其中最基本的是冒泡排序(Bubble Sort)。

交换排序

1 冒泡排序

1. 排序思想

依次比较相邻的两个记录的关键字，若两个记录是反序的(即前一个记录的关键字大于后前一个记录的关键字)，则进行交换，直到没有反序的记录为止。

① 首先将L->R[1]与L->R[2]的关键字进行比较，若为反序(L->R[1]的关键字大于L->R[2]的关键字)，则交换两个记录；然后比较L->R[2]与L->R[3]的关键字，依此类推，直到L->R[n-1]与L->R[n]的关键字比较后为止，称为一趟冒泡排序，

② 然后进行第二趟冒泡排序，对前n-1个记录进行同样的操作。

一般地，第i趟冒泡排序是对L->R[1 … n-i+1]中的记录进行的，因此，若待排序的记录有n个，则要经过n-1趟冒泡排序才能使所有的记录有序。

交换排序

例：设有9个待排序的记录，关键字分别为23, 38, 22, 45, 23, 67, 31, 15, 41，冒泡排序的过程如图所示。

初始关键字序列:	23	38	22	45	23	67	31	15	41
第一趟排序后:	23	22	38	23	45	31	15	41	67
第二趟排序后:	22	23	23	38	31	15	41	45	67
第三趟排序后:	22	23	23	31	15	38	41	45	67
第四趟排序后:	22	23	23	15	31	38	41	45	67
第五趟排序后:	22	23	15	23	31	38	41	45	67
第六趟排序后:	22	15	23	23	31	38	41	45	67
第七趟排序后:	15	22	23	23	31	38	41	45	67

交换排序

2. 性能分析

时间复杂度

- 最好情况(正序)：比较次数：n-1；移动次数：0；
- 最坏情况(逆序)：

比较次数：$\sum_{i=1}^{n-1}(n-i)=\frac{n(n-1)}{2}$

移动次数：$3\sum_{i=1}^{n-1}(n-i)=\frac{3n(n-1)}{2}$

故时间复杂度：$T(n)=O(n^2)$

空间复杂度：$S(n)=O(1)$

交换排序

1 快速排序

1. 排序思想

通过一趟排序，将待排序记录分割成独立的两部分，其中一部分记录的关键字均比另一部分记录的关键字小，再分别对这两部分记录进行下一趟排序，以达到整个序列有序。

设待排序的记录序列是R[s...t]，在记录序列中任取一个记录(一般取R[s])作为参照(又称为基准或枢轴)，以R[s].key为基准重新排列其余的所有记录，方法是：

- 所有关键字比基准小的放R[s]之前；
- 所有关键字比基准大的放R[s]之后。

以R[s].key最后所在位置i作为分界，将序列R[s...t]分割成两个子序列，称为一趟快速排序。

交换排序

例：设有8个待排序的记录，关键字分别为49, 38, 65, 97, 76, 13, 27, 49, 快速排序的过程如图所示。

初始关键字序列:	49	38	65	97	76	13	27	49
一次交换:	27	38	65	97	76	13	49	49
二次交换:	27	38	49	97	76	13	65	49
三次交换:	27	38	13	97	76	49	65	49
四次交换:	27	38	13	49	76	97	65	49
完成一趟排序:	27	38	13	49	76	97	65	49

交换排序

例：设有8个待排序的记录，关键字分别为49, 38, 65, 97, 76, 13, 27, 49, 快速排序的过程如图所示。

初始关键字序列： 49 38 65 97 76 13 27 49

一次划分： { 27 38 13 } 49 {76 97 65 49}

分别进行： { 13 27 38 } 49 {49 65 76 97}

初始关键字序列： 13 27 38 49 49 65 76 97

交换排序

2. 性能分析

快速排序的时间复杂度平均时间复杂度是：$T(n)=O(n\log_2 n)$ 最坏情况下是：$O(n^2)$　　从所需要的附加空间来看，快速排序算法是递归调用，系统内用堆栈保存递归参数，当每次划分比较均匀时，栈的最大深度为$[\log 2n]+1$ 。

快速排序的空间复杂度最坏情况下是：$O(n)$

平均情况下为$O(\log_2 n)$

从排序的稳定性来看，快速排序是不稳定的。

选择排序

选择排序(Selection Sort)的基本思想是：每次从当前待排序的记录中选取关键字最小的记录表，然后与待排序的记录序列中的第一个记录进行交换，直到整个记录序列有序为止。

选择排序

1 简单选择排序

1. 排序思想

简单选择排序(Simple Selection Sort，又称为直接选择排序)的基本操作是：通过n-i次关键字间的比较，从n-i+1个记录中选取关键字最小的记录，然后和第i个记录进行交换，i=1, 2, … n-1 。

选择排序

例：设有关键字序列为：7, 4, -2, 19, 13, 6，直接选择排序的过程如下图所示。

初始记录的关键字： 7 4 -2 19 13 6

第一趟排序：-2 4 7 19 13 6

第二趟排序：-2 4 7 19 13 6

第三趟排序：-2 4 6 19 13 7

第四趟排序：-2 4 6 7 13 19

第五趟排序：-2 4 6 7 13 19

第六趟排序：-2 4 6 7 13 19

选择排序

2. 性能分析

时间复杂度是：$T(n)=O(n^2)$

空间复杂度是：$S(n)=O(1)$

从排序的稳定性来看，直接选择排序是不稳定的。

选择排序

2 堆

1. 堆的定义

是n个元素的序列$H=\{k_1, k_2, \dots k_n\}$，满足：

$$\begin{cases} k_i \le k_{2i} \\ k_i \le k_{2i+1} \end{cases}$$ 大根堆 或 $$\begin{cases} k_i \ge k_{2i} \\ k_i \ge k_{2i+1} \end{cases}$$ 小根堆

其中： $i=1,2,\dots,\lfloor n/2 \rfloor$

由堆的定义知，堆是一棵以k_1为根的完全二叉树。若对该二叉树的结点进行编号(从上到下，从左到右)，得到的序列就是将二叉树的结点以顺序结构存放，堆的结构正好和该序列结构完全一致。

选择排序

2. 堆的性质

① 堆是一棵采用顺序存储结构的完全二叉树， k_1是根结点；
② 堆的根结点是关键字序列中的最小(或最大)值，分别称为小(或大)根堆；
③ 从根结点到每一叶子结点路径上的元素组成的序列都是按元素值(或关键字值)非递减(或非递增)的；
④堆中的任一子树也是堆。

利用堆顶记录的关键字值最小(或最大)的性质，从当前待排序的记录中依次选取关键字最小(或最大)的记录，就可以实现对数据记录的排序，这种排序方法称为堆排序。

选择排序

3. 堆排序思想

① 对一组待排序的记录，按堆的定义建立堆；

② 将堆顶记录和最后一个记录交换位置，则前n-1个记录是无序的，而最后一个记录是有序的；

③ 堆顶记录被交换后，前n-1个记录不再是堆，需将前n-1个待排序记录重新组织成为一个堆，然后将堆顶记录和倒数第二个记录交换位置，即将整个序列中次小关键字值的记录调整(排除)出无序区；

④ 重复上述步骤，直到全部记录排好序为止。

结论：排序过程中，若采用小根堆，排序后得到的是非递减序列；若采用大根堆，排序后得到的是非递增序列。

选择排序

4. 堆的调整——筛选

(1) 堆的调整思想

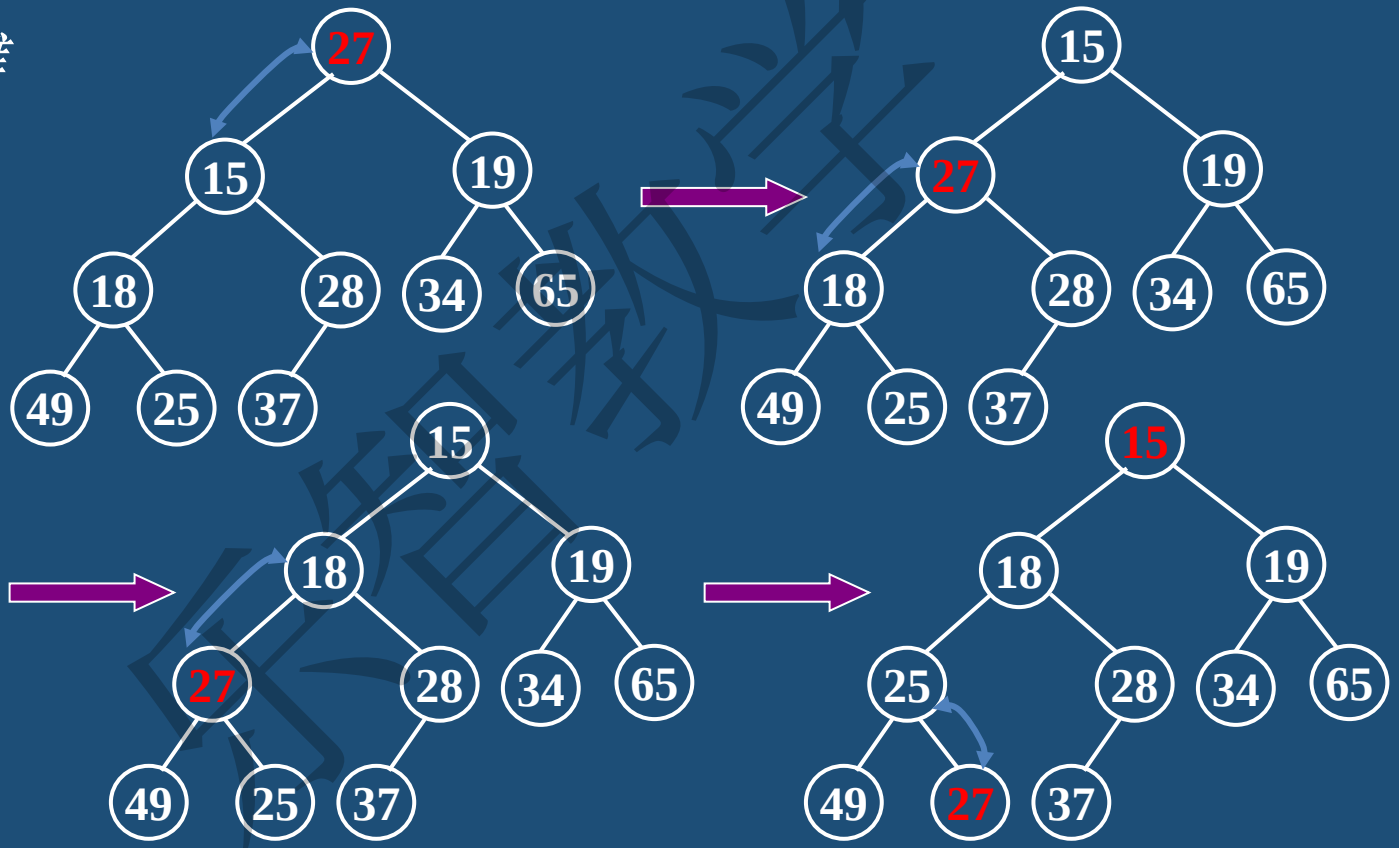
输出堆顶元素之后，以堆中最后一个元素替代之；然后将根结点值与左、右子树的根结点值进行比较，并与其中小者进行交换；重复上述操作，直到是叶子结点或其关键字值小于等于左、右子树的关键字的值，将得到新的堆。称这个从堆顶至叶子的调整过程为“筛选”，如图所示。

注意：筛选过程中，根结点的左、右子树都是堆，因此，筛选是从根结点到某个叶子结点的一次调整过程。

选择排序

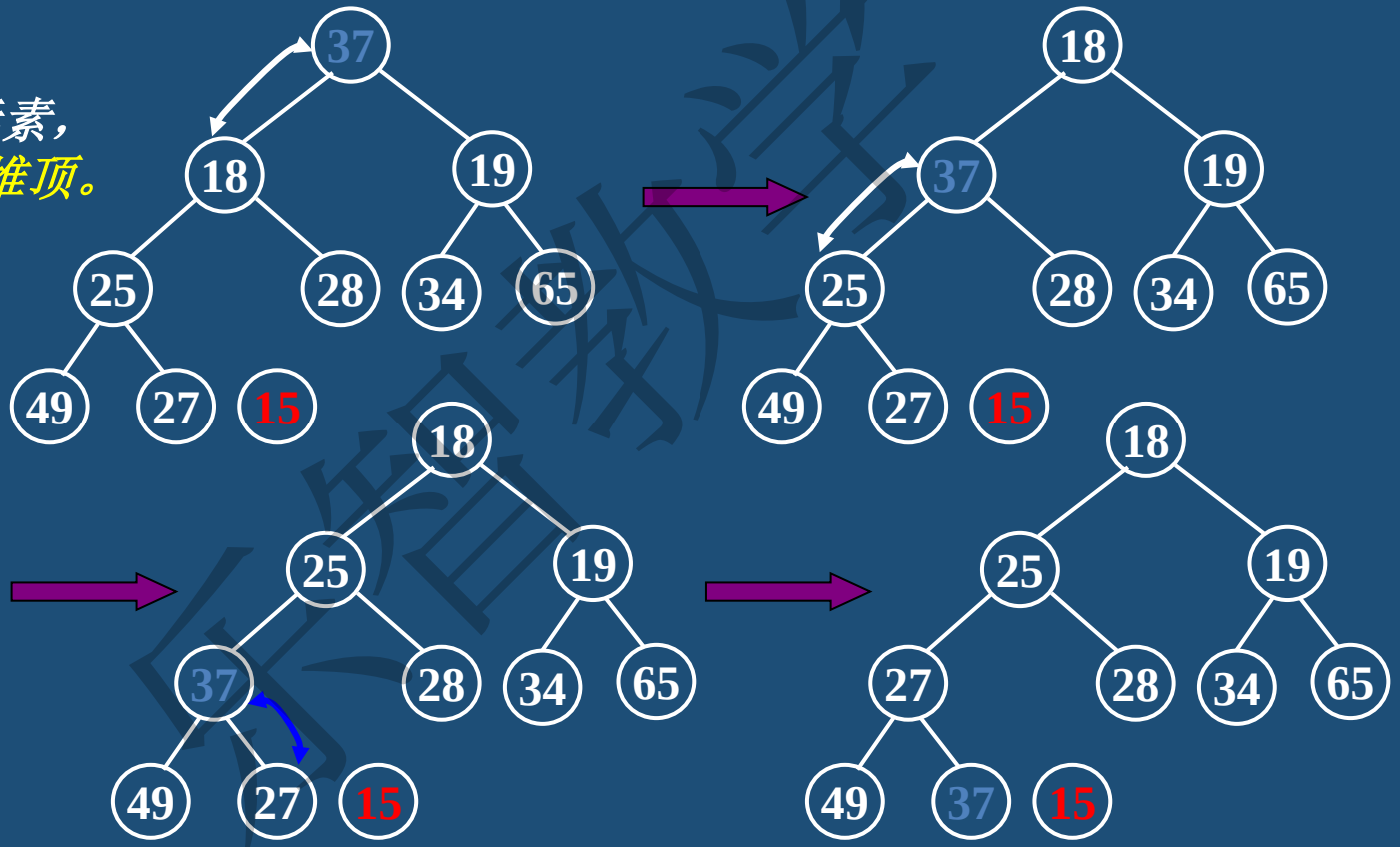
（1）建立初始堆

27 15 19 18 28 34
65 49 25 37



选择排序

（2）输出堆顶元素，
将堆底元素送至堆顶。



选择排序

（3）重新调整堆，使其成为小根堆，反复循环，直到堆中仅剩一个元素为止。

选择排序

5. 性能分析

时间复杂度是：$T(n)=O(n\log 2n)$

空间复杂度是：$S(n)=O(1)$

从排序的稳定性来看，堆排序是不稳定的。

归并排序

归并(Merging)：是指将两个或两个以上的有序序列合并成一个有序序列。若采用线性表(无论是那种存储结构)易于实现，其时间复杂度为O(m+n)。

1 排序思想

① 初始时，将每个记录看成一个单独的有序序列，则n个待排序记录就是n个长度为1的有序子序列；

② 对所有有序子序列进行两两归并，得到⌈n/2⌉个长度为2或1的有序子序列——一趟归并；

③ 重复②，直到得到长度为n的有序序列为止。

上述排序过程中，子序列总是两两归并，称为2-路归并排序。其核心是如何将相邻的两个子序列归并成一个子序列。

归并排序

例：设有9个待排序的记录，关键字分别为23, 38, 22, 45, 23, 67, 31, 15, 41，归并排序的过程如图所示。

归并排序

2. 性能分析

时间复杂度：整个归并排序的时间复杂度无论是最好还是最坏情况均为$O(n\log_2 n)$

空间复杂度：$O(n)$

归并排序是稳定的

基数排序

基数排序(Radix Sorting) 又称为桶排序或数字排序：按待排序记录的关键字的组成成分(或“位”)进行排序。

基数排序和前面的各种内部排序方法完全不同，不需要进行关键字的比较和记录的移动。借助于多关键字排序思想实现单逻辑关键字的排序。

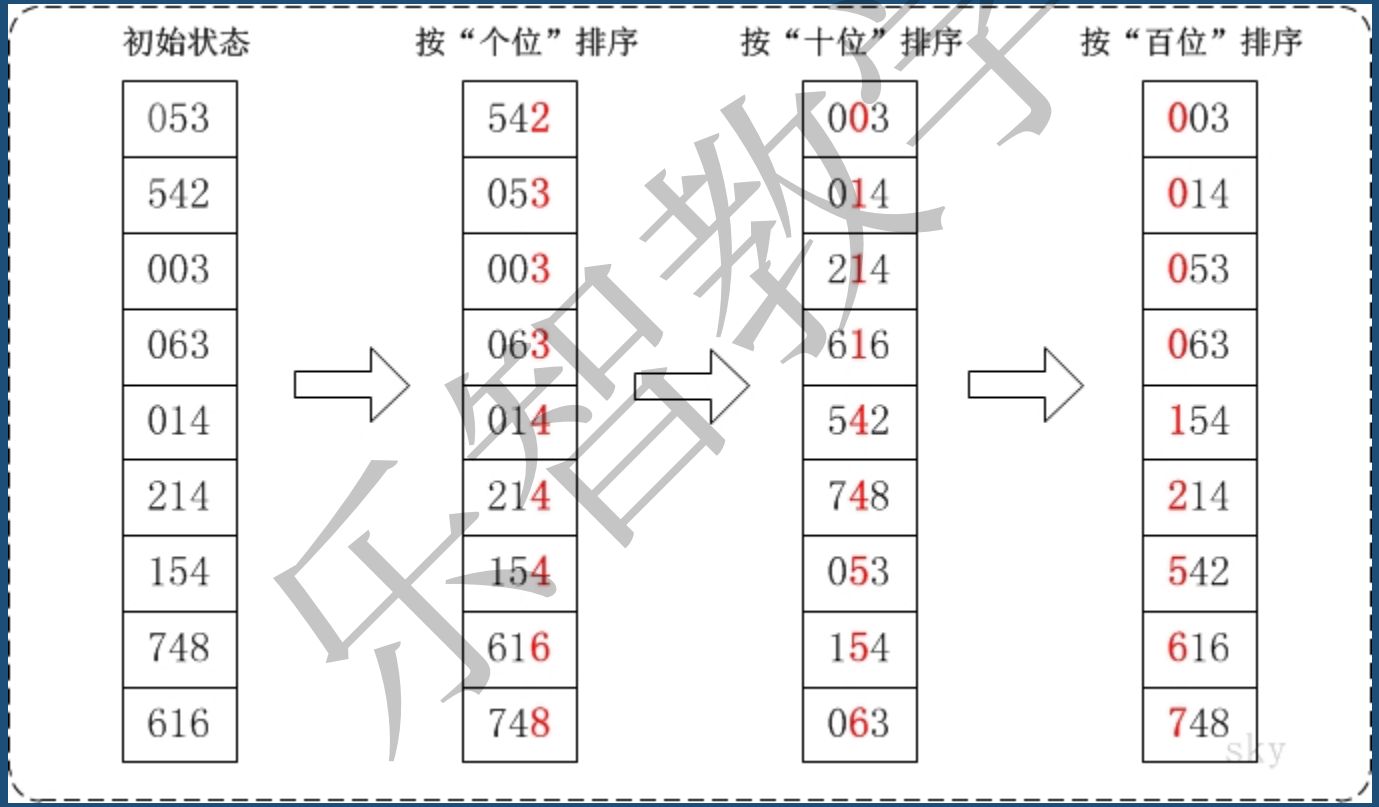
1 排序思想

先按第一个关键字K1进行排序，将记录序列分成若干个子序列，每个子序列有相同的K^1值；然后分别对每个子序列按第二个关键字K2进行排序，每个子序列又被分成若干个更小的子序列；如此重复，直到按最后一个关键字Kd进行排序。

最后，将所有的子序列依次联接成一个有序的记录序列，该方法称为最高位优先(Most Significant Digit first)。

另一种方法正好相反，排序的顺序是从最低位开始，称为最低位优先(Least Significant Digit first)。

基数排序



基数排序

2. 性能分析

设有n个待排序记录，关键字位数为d，每位有r种取值。则排序的趟数是d；在每一趟中：

基数排序的时间复杂度为： O(d(n+r))

在排序过程中使用的辅助空间是：2r个链表指针， n个指针域空间，则空间复杂度为：O(n+r)

基数排序是稳定的。

排序

各种内部排序的比较：

各种内部排序按所采用的基本思想(策略)可分为：插入排序、交换排序、选择排序，它们的基本策略分别是：

1 插入排序：依次将无序序列中的一个记录，按关键字值的大小插入到已排好序一个子序列的适当位置，直到所有的记录都插入为止。具体的方法有：直接插入、折半插入和希尔(shell)排序。

2 交换排序：对于待排序记录序列中的记录，两两比较记录的关键字，并对反序的两个记录进行交换，直到整个序列中没有反序的记录偶对为止。具体的方法有：冒泡排序、快速排序。

3 选择排序：不断地从待排序的记录序列中选取关键字最小的记录，放在已排好序的序列的最后，直到所有记录都被选取为止。具体的方法有：简单选择排序、堆排序。

排序

4 归并排序：利用“归并”技术不断地对待排序记录序列中的有序子序列进行合并，直到合并为一个有序序列为止。

5 基数排序：按待排序记录的关键字的组成成分(“位”)从低到高(或从高到低)进行。每次是按记录关键字某一“位”的值将所有记录分配到相应的桶中，再按桶的编号依次将记录进行收集，最后得到一个有序序列。

排序

各种内部排序的比较

方法	平均时间	最坏所需时间	附加空间	稳定性
直接插入	$O(n^2)$	$O(n^2)$	$O(1)$	稳定的
折半插入	$O(n^2)$	$O(n^2)$	$O(1)$	稳定的
希尔排序	$O(n^{1.3})$		$O(1)$	不稳定的
冒泡排序	$O(n^2)$	$O(n^2)$	$O(1)$	稳定的
快速排序	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$	不稳定的
直接选择	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定的
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	不稳定的
归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	稳定的
基数排序	$O(d(n+r))$	$O(d(n+r))$	$O(n+r)$	稳定的

排序

各种排序算法的性质					
算法种类	时间复杂度			空间复杂度	是否稳定
	最好情况	平均情况	最坏情况		
直接插入排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	是
冒泡排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	是
简单选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	否
希尔排序				$O(1)$	否
快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$	否
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	否
2-路归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	是
基数排序	$O(d(n+r))$	$O(d(n+r))$	$O(d(n+r))$	$O(r)$	是

排序

记忆方法：
时间复杂度：快些归队（快速 归并 堆排序） $O(n\log_2 n)$
空间复杂度：快速$O(\log_2 n)$ 归并$O(n)$ 基数$O(n+r)$ 其他都为$O(1)$
稳定性： 快些选一堆（快速 希尔 选择 堆排序） 是不稳定的
其他细节：
1）经过一次排序，能够保证一个关键字到达最终位置，这样的排序是交换的两类（冒泡、快速）和选择的两种（简单选择 堆）
2）排序算法的关键字比较次数和原始序列无关——简单选择和折半插入
3）排序算法的排序趟数和原始序列有关——交换类的排序

排序（真题检测）

1.已知集合{1039,3355,2121,4382,0066,0118,0427}，对元素进行从小到大的排序。

① 写出直接插入排序的第一次与第二次排序结果。

② 基数r=10，位数d=4，写出基数排序（最低位优先法）的第一趟和第二趟收集结果。

③ 写出快速排序第一趟排序结果。

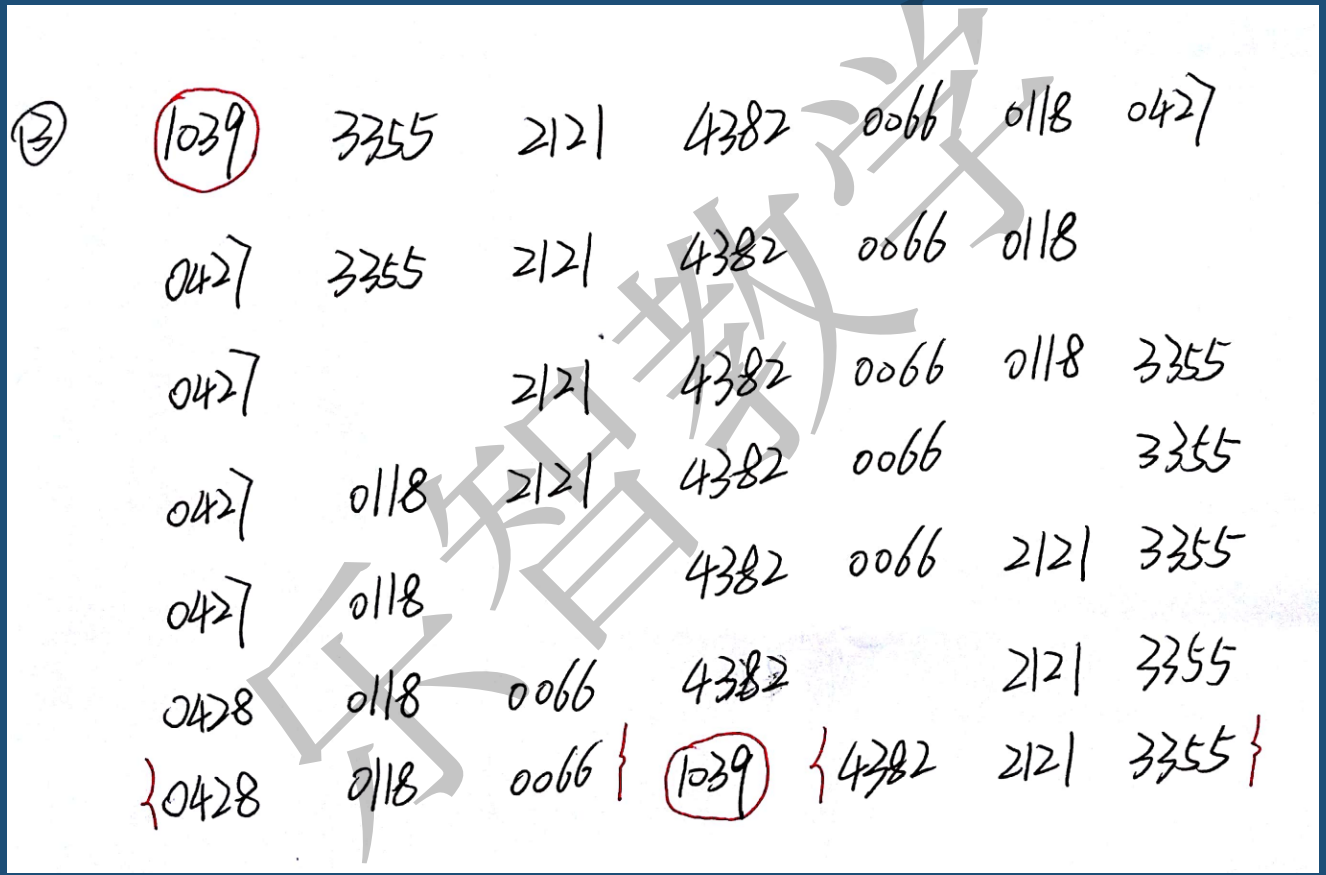
第一次	（1039）	3355	2121	4382	0066	0118	0427
第二次	（1039	3355）	2121	4382	0066	0118	0427

②解：

{2121，4382，3355，0066，0427，0118，1039}

{0118，2121，0427，1039，3355，0066，4382}

排序（真题检测）


③
1039 3355 2121 4382 0066 0118 0427
0427 3355 2121 4382 0066 0118
0427 2121 4382 0066 0118 3355
0427 0118 2121 4382 0066 3355
0427 0118 4382 0066 2121 3355
0428 0118 0066 4382 2121 3355
0428 0118 0066 1039 4382 2121 3355

排序（真题检测）

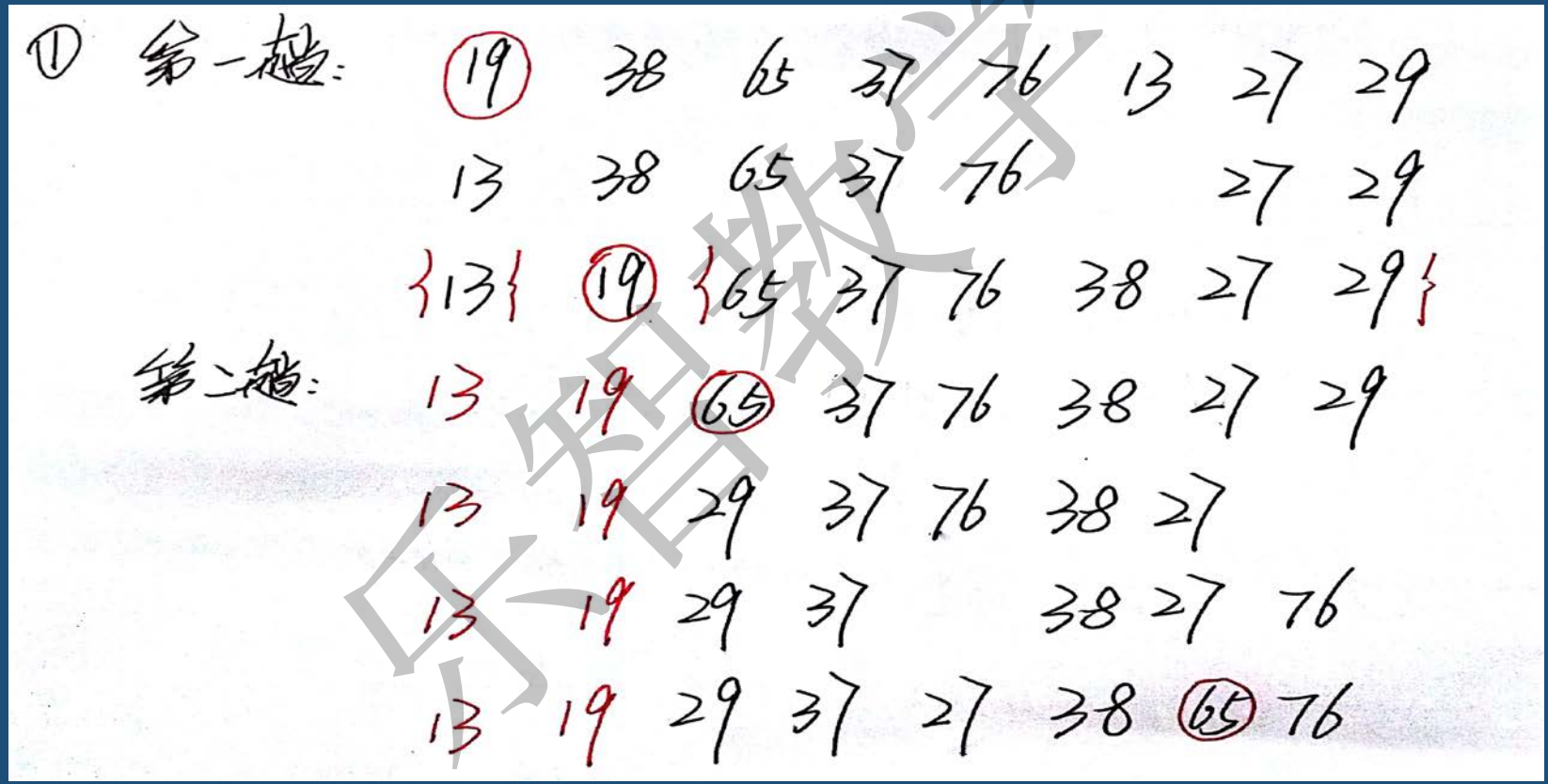
2. 已知集合{19,38,65,37,76,13,27,29}

① 写出快速排序第一趟和第二趟的结果（从待排序的n个记录中选取第一记录作为基准）。

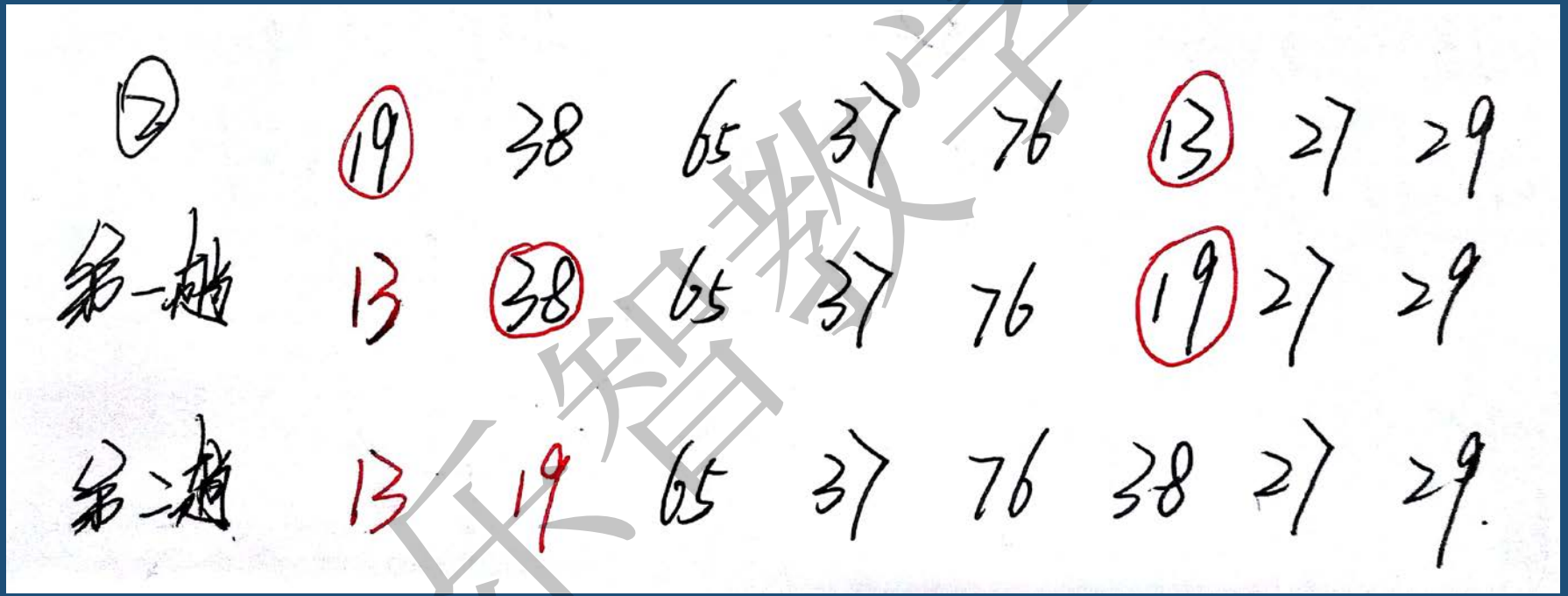
② 写出直接选择排序第一趟和第二趟的结果。

③ 写出二路归并排序全过程。

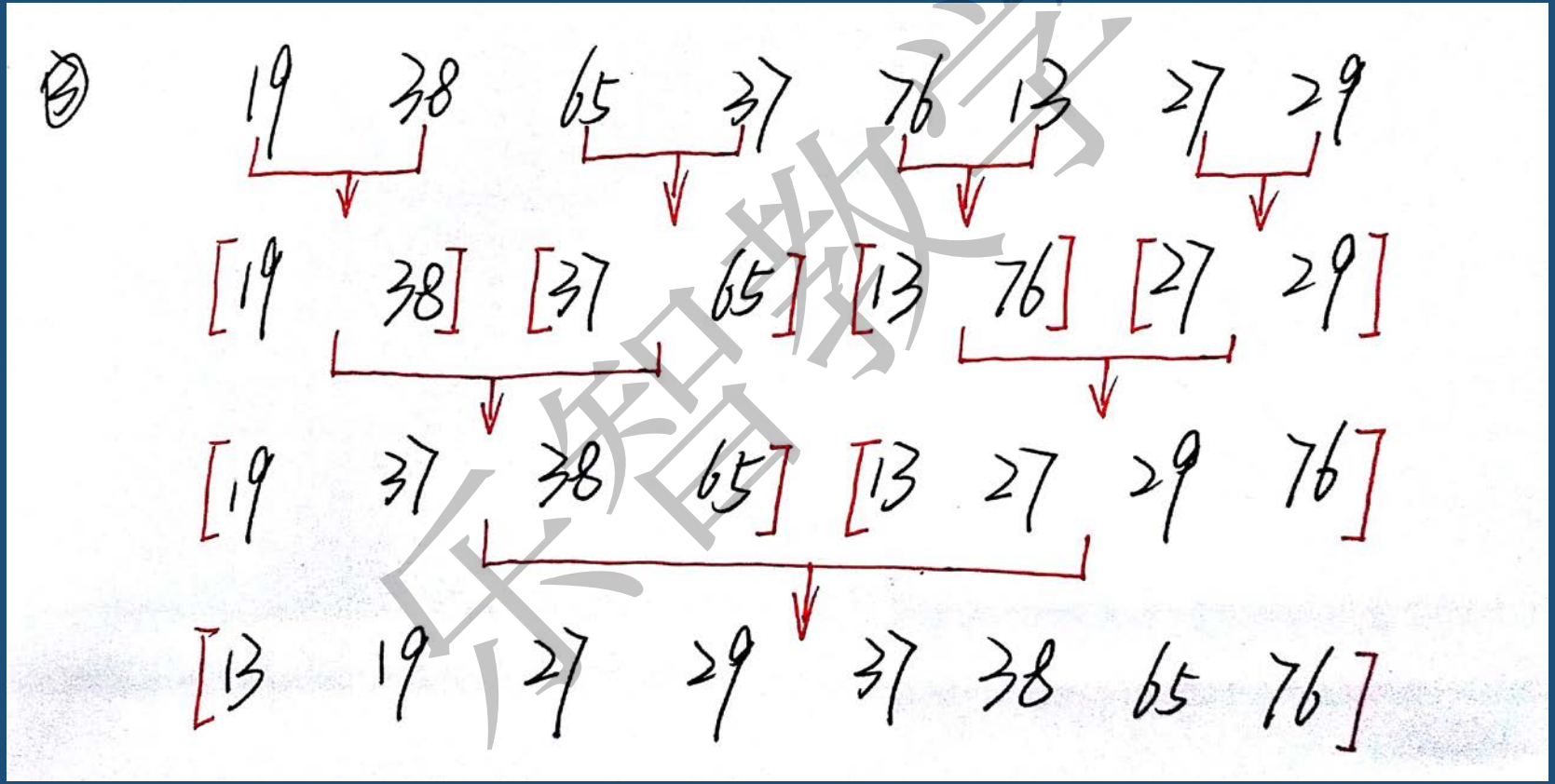
排序（真题检测）


① 第一趟：
19 38 65 37 76 13 27 29
13 38 65 37 76 27 29
{13} 19 {65 37 76 38 27 29}
第二趟：
13 19 65 37 76 38 27 29
13 19 29 37 76 38 27
13 19 29 37 38 27 76
13 19 29 37 27 38 65 76

排序（真题检测）



排序（真题检测）


⑤
19 38 65 37 76 13 27 29
[19 38] [37 65] [13 76] [27 29]
[19 37 38 65] [13 27 29 76]
[13 19 27 29 37 38 65 76]

谢 谢 观 看