第八章 排序（二）

马嫄

直接插入排序（cont.）

- 例:给定初始序列{34, 8, 64, 51, 32, 21}，冒泡排序和插入排序分别需要多少次元素交换才能完 成?
- 对于下标$i<j$，如果A[i]>A[j]，则称(i, j)是一对逆序对(inversion)
- 问题:序列{34, 8, 64, 51, 32, 21}中有多少逆序对?

  (34, 8) (34, 32) (34, 21) (64, 51) (64, 32) (64, 21) (51, 32) (51, 21) (32, 21)
- 交换 2个相邻元素正好消去 1个逆序对 !
- 插入排序： T(N, I) = O(N+I)

  — 如果序列基本有序，则插入排序简单且高效

直接插入排序（cont.）

- 定理: 任意N个不同元素组成的序列平均具有 N (N-1) / 4 个逆序对
- 定理: 任何仅以交换相邻两元素来排序的算法，其平均时间复杂度为 $\Omega(N^2)$
- 这意味着: 要提高算法效率，我们必须
  - 每次消去不止1个逆序对!
  - 每次交换相隔较远的2个元素!

希尔排序

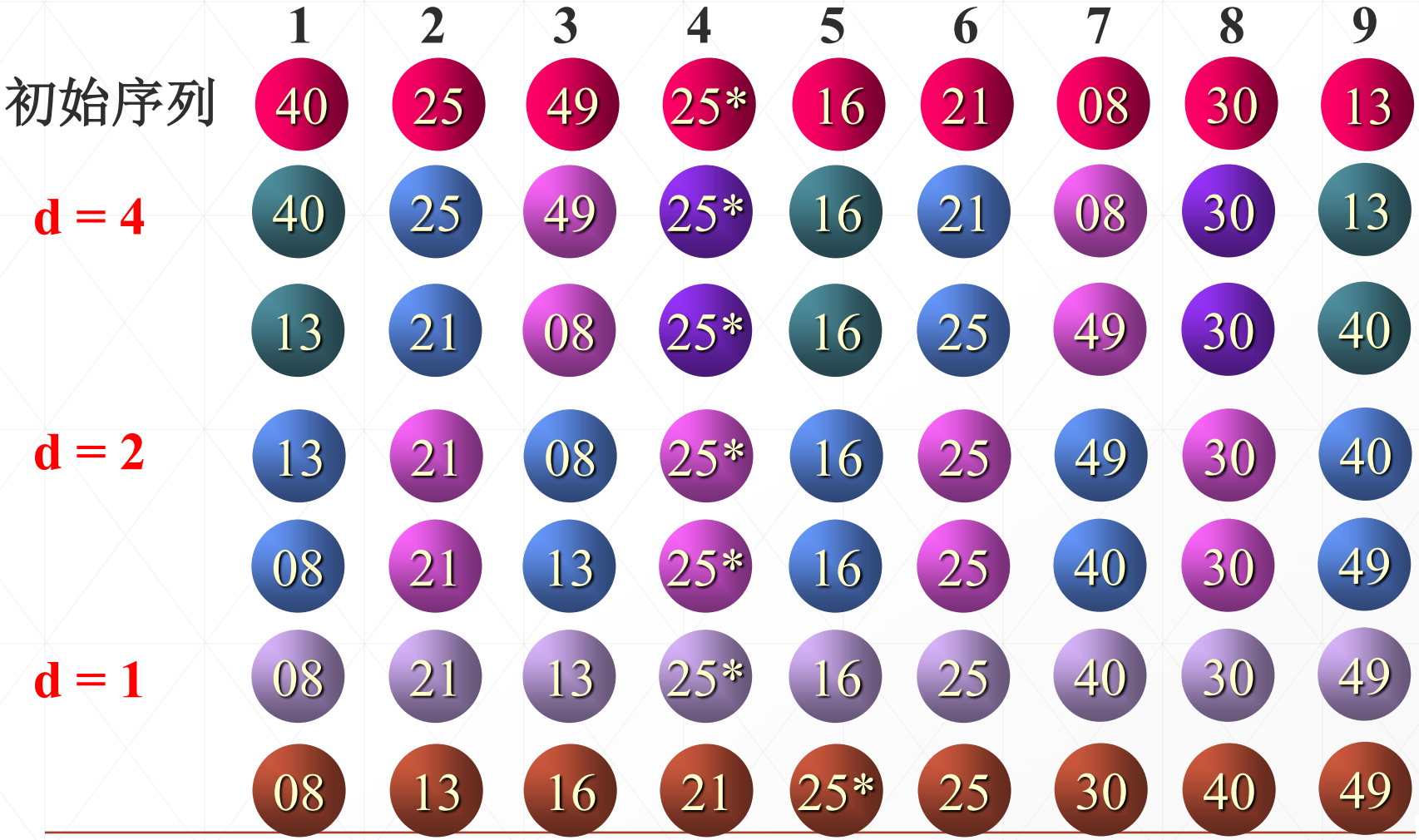
- 希尔排序是对直接插入排序的改进，改进的着眼点：
  - 若待排序记录按关键字值基本有序时，直接插入排序的效率可以大大提高；
  - 由于直接插入排序算法简单，则在待排序记录数量n较小时效率也很高。
- 希尔排序的基本思想：
  - 将整个待排序记录分割成若干个子序列，在子序列内分别进行直接插入排序，待整个序列中的记录基本有序时，对全体记录进行直接插入排序。

希尔排序

- 首先选一个整数gap< n(待排序记录数)作为间隔，将全部序列分为若干个子序列，所有距离为gap的记录放在同一个子序列中；
- 在每个子序列内分别进行直接插入排序；
- 然后缩小间隔gap，例如取gap=gap/2；
- 重复上述的子序列划分和排序工作，直到最后取gap=1，将所有记录放在同一个序列中排序为止。
- 希尔排序又称缩小增量排序。

希尔排序(cont.)

- 示例：缩小增量排序



希尔排序(cont.)

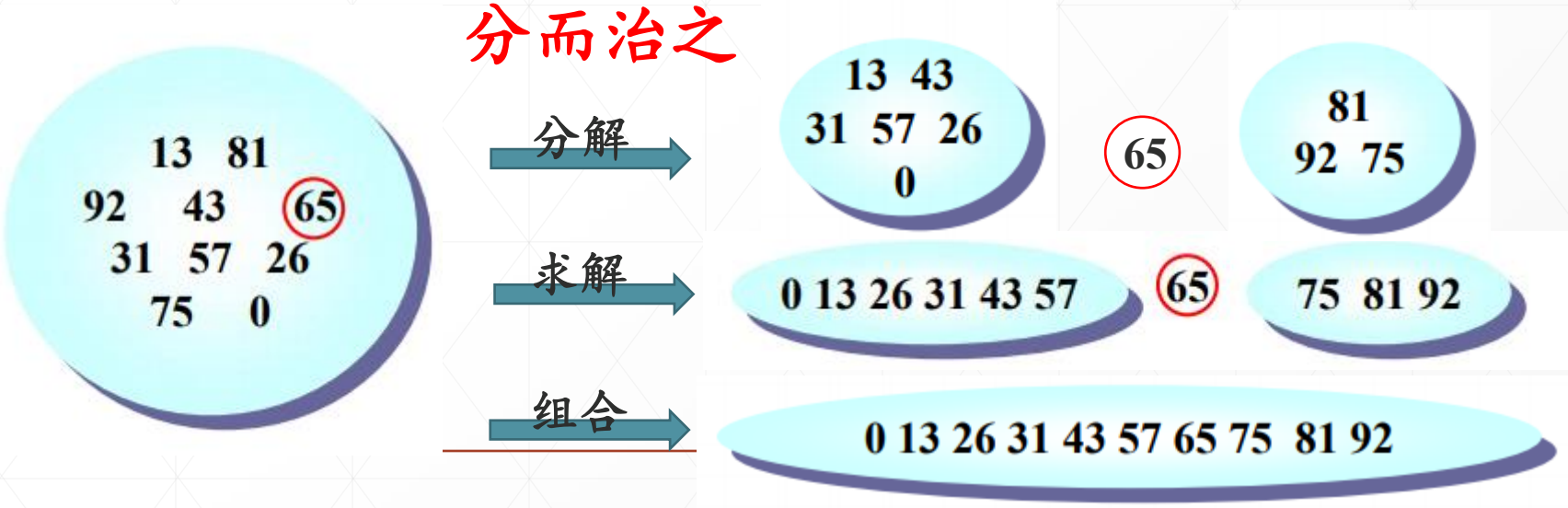
- 算法的性能分析：
  - 希尔排序开始时增量较大，每个子序列中的记录个数较少，从而排序速度较快；
  - 当增量逐渐变小时，虽然每个子序列中记录个数逐渐变多，但整个序列已基本有序，排序速度也较快。
  - 希尔排序的时间性能是所取增量的函数，而到目前为止尚未有人求得一种最好的增量序列。
  - 研究表明，当n在某个特定范围内，希尔排序所需的比较次数和记录的移动次数约为$n^{1.3}$；当n趋于无穷时可减少到$n(log_2 n)^2$
  - 希尔排序是一种不稳定的排序方法

快速排序

- 算法的基本思想：

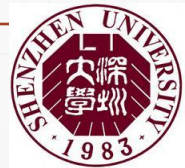
  是C.R.A.Hoare 1962年提出的一种划分交换排序。采用的是分治策略(一般与递归结合使用)，以减少排序过程中的比较次数

- 通过一趟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，然后再按此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



快速排序（cont.）

- 任取待排序记录序列中的某个记录（例如取第一个记录）作为基准（枢），按照该记录的关键字大小，将整个记录序列划分成左右两个子序列
- 左侧子序列中所有记录的关键字都小于或等于基准记录的关键字
- 右侧子序列中所有记录的关键字都大于基准记录的关键字

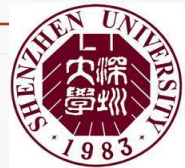


快速排序（cont.）

- 取序列第一个记录为枢轴记录，其关键字为Pivotkey；
  指针low指向序列第一个记录位置；
  指针high指向序列最后一个记录位置
- 一趟排序（某个子序列）过程

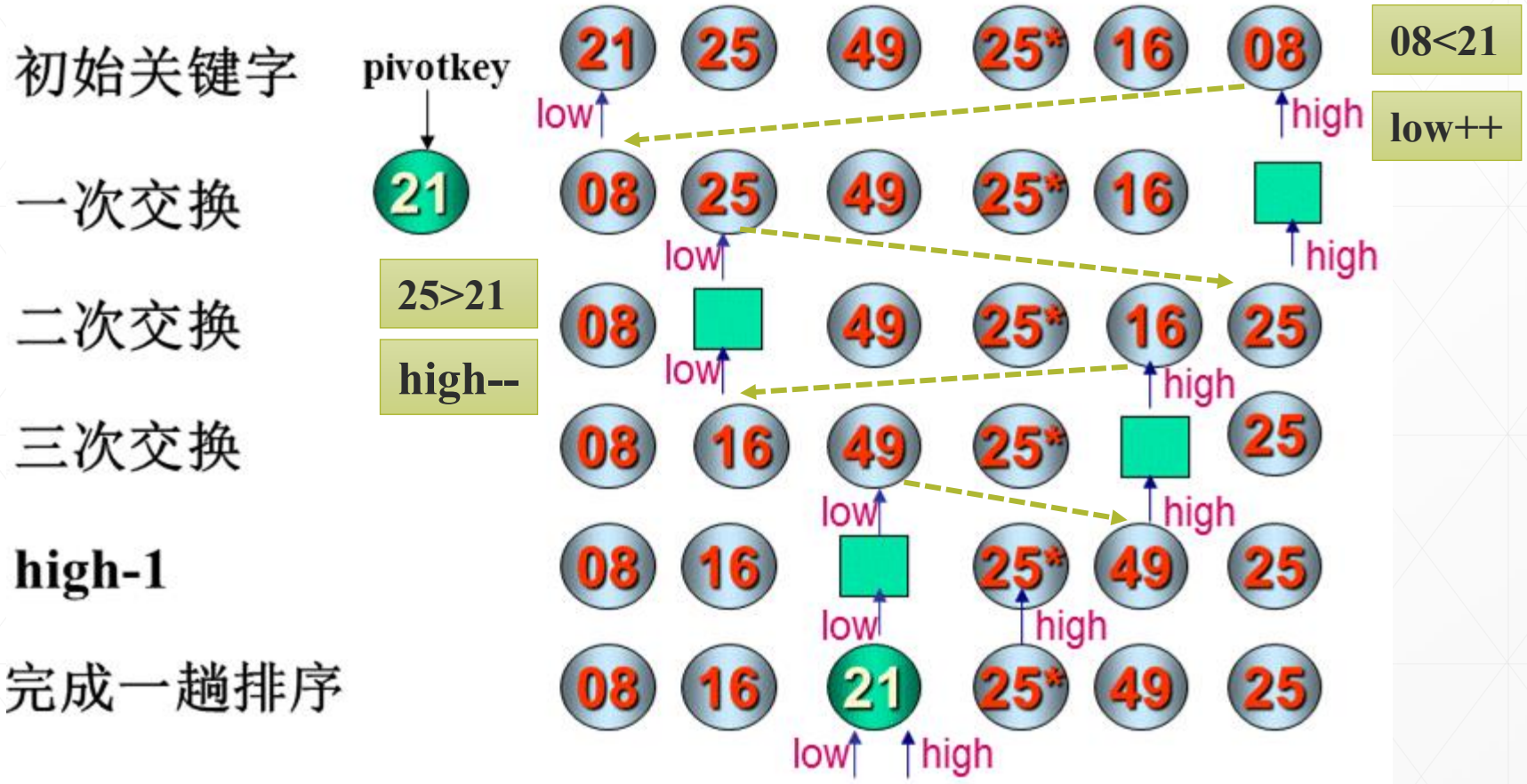
1. 从high指向的记录开始，向前找到第一个关键字的值小于Pivotkey的记录，将其放到low指向的位置，low++
2. 从low指向的记录开始，向后找到第一个关键字的值大于Pivotkey的记录，将其放到high指向的位置，high—
3. 重复1，2，直到low=high，将枢轴记录放在low （ high ）指向的位置

- 对枢轴记录前后两个子序列执行相同的操作，直到每个子序列都只有一个记录为止



快速排序（cont.）

- 快速排序举例



快速排序（cont.）

- 快速排序举例

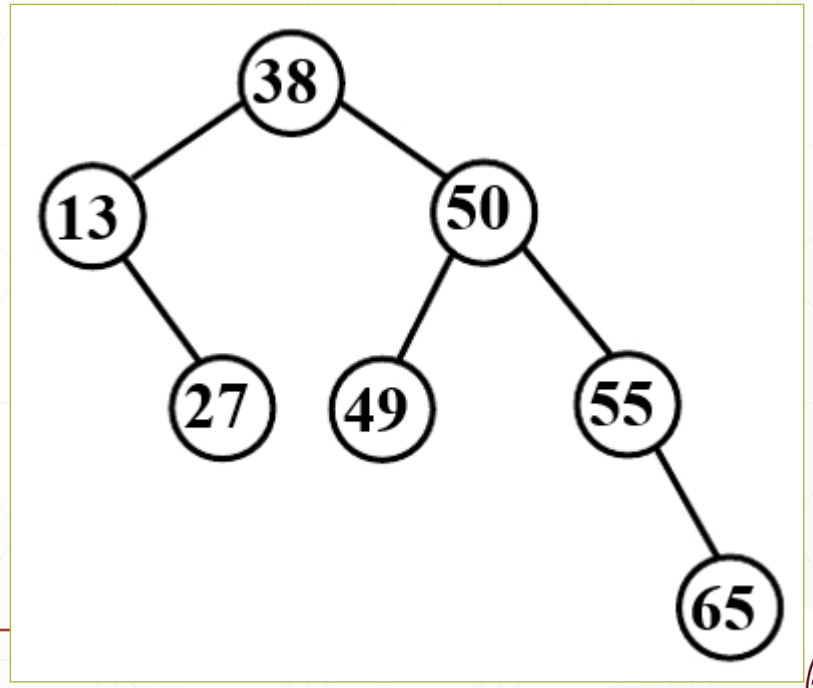
完成一趟排序

分别进行快速排序

有序序列

快速排序（cont.）

- 快速排序是一个递归过程，递归执行过程可以用递归树描述
- 利用序列第一个记录作为基准，将整个序列分为左右两个子序列。只要是关键字小于基准记录关键字的记录都移到序列左侧
- 例如，序列 {38, 27, 55, 50, 13, 49, 65}的快速排序递归如下



快速排序（cont.）

- 快速排序的趟数取决于递归树的高度

I. 最好情况：

- 每一次划分后，划分点的左侧子序列与右侧子序列的长度相同，则有

$$T(n) \le 2T(n/2)+cn \quad \text{// } c\text{是一个常数}$$

$$\le 2(2T(n/4)+cn/2)+cn = 4T(n/4)+2cn$$

$$\le 4(2T(n/8)+cn/4)+2cn = 8T(n/8)+3cn$$

$$\cdots\cdots$$

$$\le nT(1)+cn\log_2 n = O(n\log_2 n)$$

- 时间复杂度为$O(n\log_2 n)$

快速排序（cont.）

- 快速排序的趟数取决于递归树的高度

II. 平均情况：

- 可以证明，快速排序的平均计算时间也是$O(n\log_2 n)$

- 实验结果证明，就平均计算时间而言，快速排序是所有内排序方法中最好的一个，但是快速排序是一种不稳定的排序方法

快速排序（cont.）

- 快速排序的趟数取决于递归树的高度

III. 最坏情况：

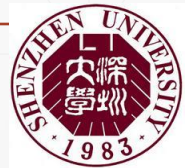
- 待排序记录序列已经按其关键字从小到大排好序，其递归树为单支树
- 每次划分只得到一个比上一次划分少一个记录的子序列，必须经过$n-1$趟才能把所有记录定位
- 而且第i趟需要经过$n-i$次关键字比较才能找到第i个记录的安放位置，总的关键字比较次数将达到

$$\sum_{i=1}^{n-1} n\text{-}i = \frac{1}{2}n(n\text{-}1) \approx \frac{n^2}{2}$$

- 时间复杂度为$O(n^2)$

快速排序（cont.）

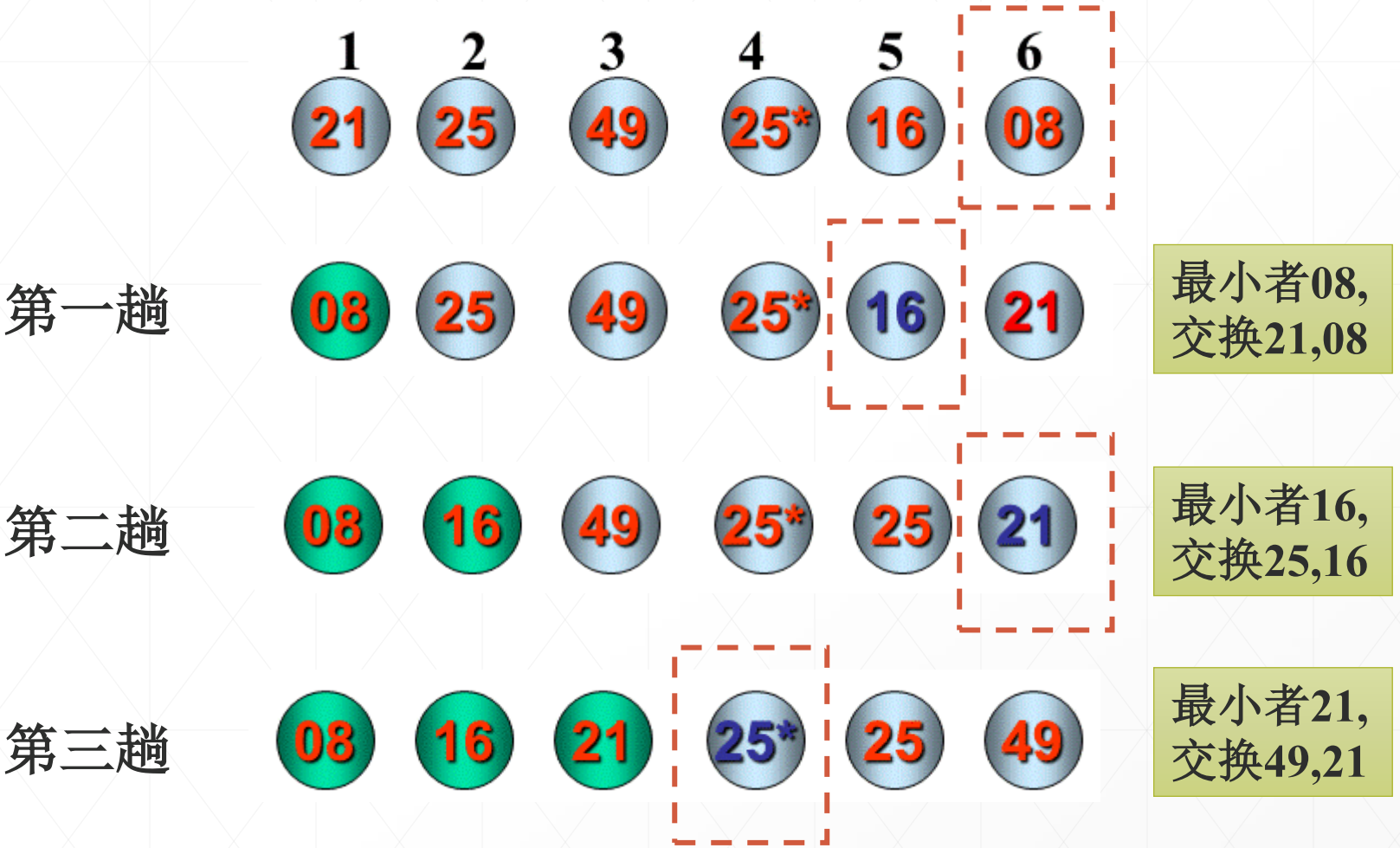
- 快速排序的改进：枢轴记录取low、high、(low+high)/2三者指向记录关键字居中的记录



简单选择排序

- 选择排序的主要操作是选择，其主要思想是：每趟排序在当前待排序序列中选出关键字值最小（最大）的记录，添加到有序序列中。
- 简单选择排序，对待排序的记录序列进行$n-1$遍的处理，第1遍处理是将A[1 ... n]中最小者与A[1]交换位置，第2遍处理是将A[2 ... n]中最小者与A[2]交换位置，.....，第i遍处理就是通过$n-i$次比较，将A[i ... n]中最小者与A[i]交换位置。这样，经过i遍处理之后，前i个记录的位置就已经按从小到大的顺序排列好了
- 简单选择排序与气泡排序的区别在：气泡排序每次比较后，如果发现顺序不对立即进行交换，而选择排序不立即进行交换，而是找出最小关键字记录后再进行交换。

简单选择排序（cont.）


1 2 3 4 5 6
21 25 49 25* 16 08
第一趟
08 25 49 25* 16 21
最小者08,
交换21,08
第二趟
08 16 49 25* 25 21
最小者16,
交换25,16
第三趟
08 16 21 25* 25 49
最小者21,
交换49,21

简单选择排序（cont.）

第四趟

最小者25*，不需要交换

第五趟

最小者25，不需要交换

简单选择排序（cont.）

- 简单选择排序的关键字比较次数与记录的初始排列无关
- 设整个待排序记录序列有n个记录，则第i趟选择最小关键字记录所需的比较次数总是$n-i$次。总的关键字比较次数为

$$\sum_{i=1}^{n} n-i = n(n-1)/2$$

- 移动次数

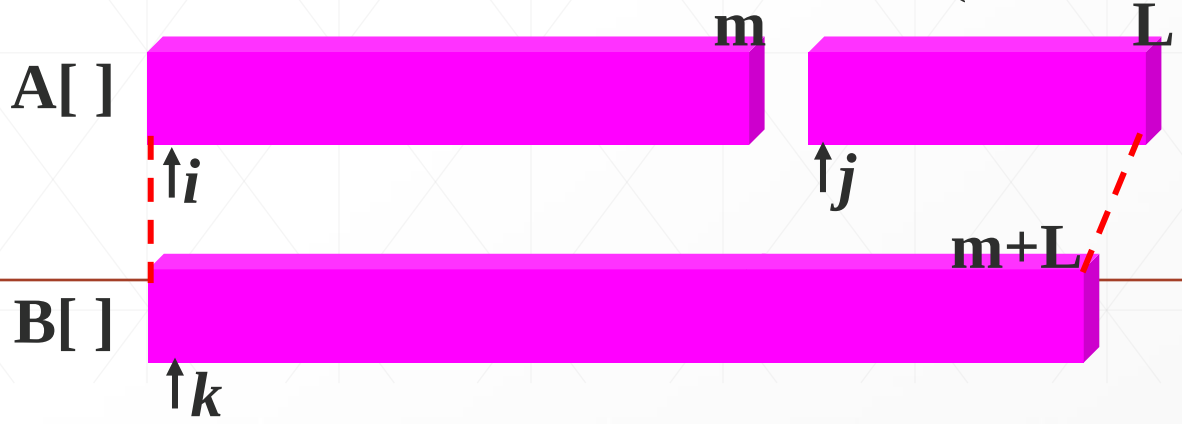
I. 最好情况（正序）：0次

II. 最坏情况（每一趟都要进行交换）： $3(n-1)$

- 简单选择排序是一种不稳定的排序方法

归并排序

- 归并：将两个或两个以上的有序序列合并成一个有序序列的过程。
- 归并排序的主要操作是归并，其主要思想是：将若干有序序列逐步归并，最终得到一个有序序列。
- 两路归并
  - 假设待归并的两个有序表长度分别为m和L，则两路归并后，新的有序表长度为m+L，
  - 两路归并操作至多只需要m+L次移位和m+L次比较
  - 因此两路归并的时间复杂度为O(m+L）



归并排序（cont.）

- 二路归并排序的基本思想（自底向上的非递归算法）
  - 将具有n个待排序记录的序列看成是n个长度为1的有序序列；
  - 然后进行两两归并，得到⌈n/2⌉个长度为2的有序序列；
  - 再进行两两归并，得到⌈n/4⌉个长度为4的有序序列；
  - ……
  - 直至得到1个长度为n的有序序列为止。

{26} {5} {77} {1} {61} {11} {59} {15} {48} {19}

{ 5 26 } { 1 77 } { 11 61 } { 15 59 } { 19 48 }

{ 1 5 26 77 } { 11 15 59 61 } { 19 48 }

{ 1 5 11 15 26 59 61 77 } { 19 48 }

{ 1 5 11 15 19 26 48 59 61 77 }

归并排序（cont.）

- （二路）归并排序算法性能分析
  - 时间性能：

    一趟归并操作是将A[1]~A[n]中相邻的长度为h的有序序列进行两两归并，并把结果存放到B[1]~B[n]中，这需要O(n)时间。整个归并排序需要进行$\lceil log_2 n \rceil$趟，因此，总的时间代价是O($nlog_2 n$)。
  - 空间性能：

    算法在执行时，需要占用与原始记录序列同样数量的存储空间，因此空间复杂度为O(n)。
  - 归并排序是一种稳定的排序方法

各种排序方法的比较

I. 时间复杂度比较：

排序方法	平均情况	最好情况	最坏情况
直接插入排序	$O(n^2)$	$O(n)$	$O(n^2)$
起泡排序	$O(n^2)$	$O(n)$	$O(n^2)$
简单选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$
希尔排序	$O(n\log_2 n)$	$O(n^{1.3})$	$O(n^2)$
快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$
(二路)归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$

各种排序方法的比较（cont.）

II. 空间复杂度比较和稳定性比较：

排序方法	辅助空间	稳定性	适合情况
直接插入排序	$O(1)$	稳定	记录数较少
起泡排序	$O(1)$	稳定	不太多
简单选择排序	$O(1)$	不稳定	不太多
希尔排序	$O(1)$	不稳定	不太多
快速排序	$O(\log_2 n)$	不稳定	较多
(二路)归并排序	$O(n)$	稳定	都可以