# 第九章 排序

本章学习另外一种计算机中常用的操作——排序，它又称为分类（Sorting）。

# §9.1 分类概述

## 一、什么是分类?

排序（Sorting），它计算机内经常进行的一种操作，其目的是将一组“无序”的记录序列调整为“有序”的记录序列。*具体定义为:*

假设含n个记录的序列为 $\{ R_1, R_2, \ldots, R_n \}$，
其相应的关键字序列为 $\{ K_1, K_2, \ldots, K_n \}$，
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：$K_{p1} \le K_{p2} \le \ldots \le K_{pn}$
按此固有关系将记录序列 $\{ R_1, R_2, \ldots, R_n \}$ 重新排列为 $\{ R_{p1}, R_{p2}, \ldots, R_{pn} \}$ 的过程称作分类。

# §9.1 分类概述

$\{ R_1, R_2, \cdots, R_n \}$

分类方法

$\{ R_{p1}, R_{p2}, \cdots, R_{pn} \}$

原始序列为：

52, 49, 80, 36, 14, 58, 61, 23, 97, 75

按某方法分类后：

14, 23, 36, 49, 52, 58, 61 ,75, 80, 97

# §9.1 分类概述

## 二、内部分类和外部分类

内部分类：若整个分类过程不需要访问外存便能完成，则称为内部分类；

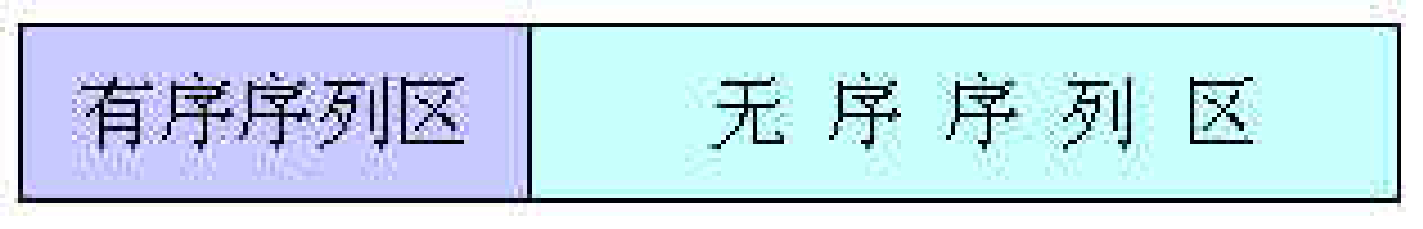
外部分类：若参加分类的记录数量很大，整个序列的分类过程不可能在内存中完成，则称为外部分类。

我们重点讨论内部分类的方法

# §9.1 分类概述

## 三、内部分类的方法

内部排序的过程是一个逐步扩大记录的有序序列长度的过程。在排序的过程中，参与排序的记录序列中存在两个区域：有序区和无序区，如下图：




使有序区中记录的数目增加一个或几个的操作称为*一趟排序*!

一种分类方法执行一趟分类的方法是不同的，且一般都由多趟组成（方法不同趟数不同）。

# §9.1 分类概述

根据分类方法进行一趟的基本操作不同，内部分类方法分为下面几大类：

1、基于“插入”思想的分类方法

执行一趟是将一个元素“插入”到有序序列中仍然有序，使有序部分扩大。这类方法有：

**直接插入分类**
**折半插入分类**
**表插入分类**
**2路插入分类**
**SHELL分类**

# §9.1 分类概述

## 2、基于“交换”思想的分类方法

执行一趟是通过交换“逆序”元素使之到有序序列中，使有序部分扩大。这类方法有：

- 冒泡（标准交换）分类
- 奇偶（成对）交换分类
- 穿梭分类
- 快速分类

## 3、基于“选择”思想的分类方法

执行一趟是通过出当前无序部分的最小元素放到有序序列的后面，使有序部分扩大。这类方法有：

- 简单选择分类
- 锦标赛（打擂台）分类
- 堆分类

# §9.1 分类概述

## 4、基于“归并”思想的分类方法

执行一趟是通过归并两个短的有序序列为一个有序序列，使有序部分扩大。这类方法有：

**2路归并分类**
**多路归并分类**

## 5、其他思想的分类方法

**计数分类**
**基数分类**

# §9.1 分类概述

## 四、内部分类方法的效率问题

时间效率：比较次数、交换或移动次数；
（1次交换=3次移动）；

空间效率：除了存储元素本身外，分类过程中需要的空间大小；

- 普通分类方法：$O(n^2)$
- 高效分类方法：$O(n\log_2 n)$

# §9.1 分类概述

## 五、分类方法的稳定性

若有两个记$R_i$,$R_j$，$K_i=K_j$，$(1\le i\le n, 1\le j\le n)$，且分类前$R_i$在$R_j$之前(即$i<j$)，若分类后$R_i$仍在$R_j$之前(即$i<j$)，则称所用的分类方法是稳定的。

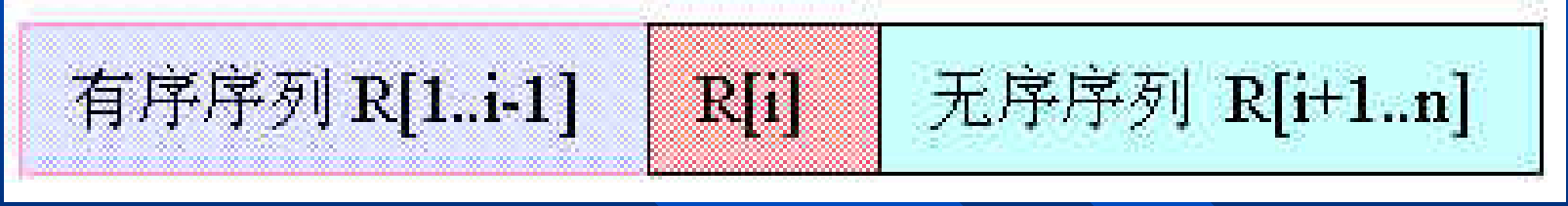
反之，若分类后可能会出现$R_i$在$R_j$之后(即$i>j$)，则称所用的分类方法是不稳定的。

** 一般普通分类方法都是稳定的，
高效分类方法都是不稳定的！！

# §9.2 插入分类方法

## 一、插入分类方法的基本思想

假设待分类记录集合为 $R_1,R_2,...R_n$，简记为R[1..n]。插入分类方法由n趟组成，假设要进行第i趟，此时第1~i-1个记录已经插入排好序，第i趟是将第i个记录插入到有序序列中，使之仍然有序。




“将记录$R_i$插入到有序子序列R[1..i-1]中，使记录的有序序列从R[1..i-1]变为R[1..i]”。即找到$R_i$的位置并放入该位置！

# §9.2 插入分类方法

显然，完成这个“插入”需分三步进行：

1、查找$R_i$的插入位置j；

2、将R[j..i-1]中的记录后移一个位置；

3、将$R_i$复制到$R_j$的位置上。

**根据查找位置的方法不同、移动记录的方法不同，插入分类有多种方法：**

直接插入分类——查找采用顺序查找方法

折半插入分类——查找采用折半查找方法

2-路插入分类——移动有了变化

SHELL分类——提高效率的改进（移动步长变化）

# §9.2 插入分类方法

## 一、直接插入分类

### 1、基本思想

利用*顺序查找*实现“在R[1..i-1]中查找R[i]的插入位置”的插入排序。

第i趟描述为：

（1）从Ri-1起向前进行顺序查找,查找位置j，满足：

$R_j.key \le R_i.key < R_{j+1}.key$ ， 监视哨设置在$R_0$。

（2）对于在查找过程中找到的那些关键字不小于$R_i.key$的记录，并在查找的同时实现记录向后移动；

（3）插入到正确位置，$R_j := R_0$

# §9.2 插入分类方法——直接插入分类

## 2、算法

略

## 3、举例：

待排序序列： **49 38 65 97 76 13 27 49**

略

# §9.2 插入分类方法——直接插入分类

## 4、效率分析：

最好的情况（关键字在记录序列中顺序有序）：

“比较”的次数 $\sum_{i=2}^{n} 1 = n-1$

“移动”的次数 $0$

最坏的情况（关键字在记录序列中逆序有序）：

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

“移动”的次数 $\sum_{i=2}^{n} (i+1) = \frac{(n+4)(n-1)}{2}$

# §9.2 插入分类方法——直接插入分类

若待排序的序列是随机的，即待排序序列中的记录可能出现的各种排列的概率相同，则：

平均情况下：

“比较”的次数： 最好和最坏的平均值，约为$n^2/4$
“移动”的次数： 最好和最坏的平均值，约为$n^2/4$

时间： $O(n^2)$

空间：辅助空间1个， $O(1)$

# §9.2 插入分类方法

## 二、折半插入分类

### 1、基本思想

因为R[1..i-1]是一个按关键字有序的有序序列，则可以利用*折半查找*实现“在R[1..i-1]中查找R[i]的插入位置”如此实现的插入排序为折半插入排序。

第i趟描述为：

（1）在[R1..Ri-1]中采用折半查找,查找位置j，满足：
$R_j.key \le R_i.key < R_{j+1}.key$ ，监视哨设置在$R_0$。

（2）找到位置后，移动元素；

（3）插入到正确位置，$R_j := R_0$

# §9.2 插入分类方法——折半插入分类

## 2、算法

略

## 3、举例：

待排序序列： 49 38 65 97 76 13 27 49

略

## 4、效率分析：

与直接插入比较，该方法的“比较“次数减少了许多, $O(n\log_2 n)$。“移动”次数没有减少，$O(n^2)$。

时间： $O(n^2)$

空间：辅助空间1个， $O(1)$

# §9.2 插入分类方法

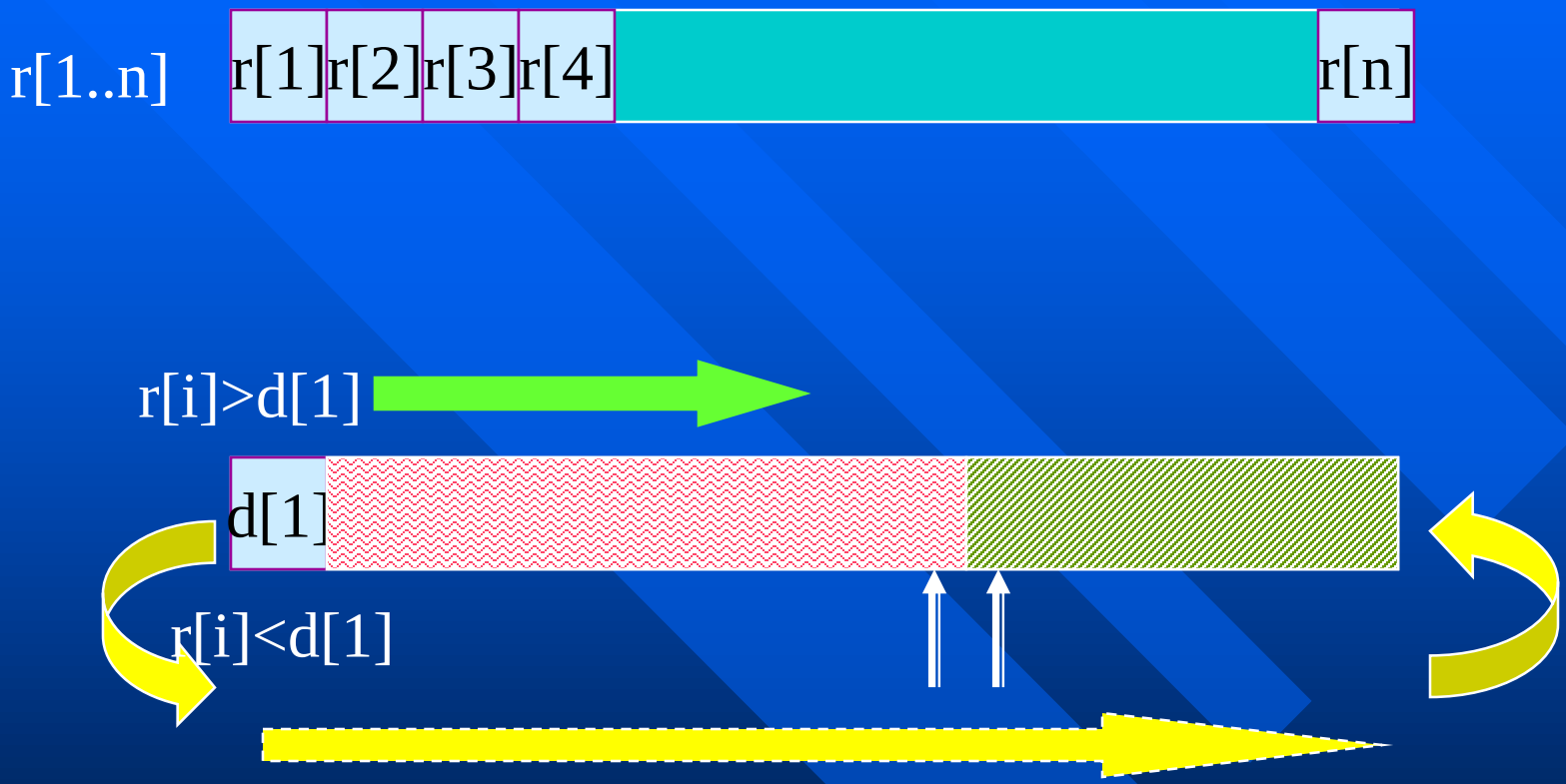
## 三、2-路插入分类

### 1、基本思想

为了减少移动次数，我们可以采用如下的策略：另设一个和r同类型的数组d, 首先将r[1]赋值给d[1], 并将d[1]看成是排好序的序列中处于中间位置的记录，然后从r中的第2个记录起依次视比d[1]大还是小插入到d[1]之前或后的序列中中：

待插入关键字与d[1]比较，若r[i].key<d[1].key，则将r[i]插入到d[1]之前的有序表中；反之，将r[i]插入到d[1]之后的有序表中。（分别采用折半方法）

# §9.2 插入分类方法—2-路插入分类



# §9.2 插入分类方法—2-路插入分类

## 2、举例

| 49 | 38 | 65 | 97 | 76 | 13 | 27 | 49 |
|---|---|---|---|---|---|---|---|

| 49 | 49 | 65 | 76 | 97 | 13 | 27 | 38 |
|---|---|---|---|---|---|---|---|

## 3、效率

查找位置采用折半，效率提高；
分别插入到两部分，移动次数减少（不能避免）；
需要的辅助空间大o(n)；

# §9.2 插入分类方法

## 四、SHELL分类（缩小增量法）

### 1、基本思想

根据插入分类的特点，当元素比较少时效率比较高；或者，当元素已经基本有序时，效率比较高，为此提出了基于下面思想的插入分类方法：

开始时，把元素分为几组，组内元素是比较少的，因此组内插入分类时效率比较高，每进行一躺后，增加组内元素的个数，与此同时，元素也越来越有序，效率也会越来越高。

组内元素个数增加
效率在降低！

随趟数的增加

组内元素越来越有序
效率在提高！

# §9.2 插入分类方法——SHELL分类

具体地： 有一个增量序列：$d_1,d_2,d_3,...,d_k$， 满足：

$d_1>d_2>d_3...>d_k=1$

分类共有 k 趟：

第 i 趟：取增量值 $d_i$，对第 i-1 趟的结果序列，把相距 $d_i$的元素“视”为一组，组内进行插入分类（直接插入分类或折半插入分类）。

i=1,2,...k

特点：关键字之间的比较不是逐个比较的，而是跳跃式的。开始时，关键字是大“步幅”地调整，随着步幅的变小，序列也越来越有序，最后dk=1时，进行逐个比较，但是，元素已经基本有序了。

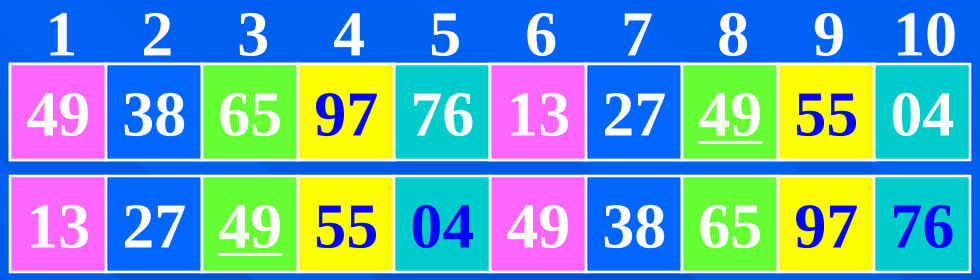
增量序列的取法：有很多方法，有人在研究。一般地：

$d_1=n/2$ $d_i=d_{i-1}/2$ n是元素个数

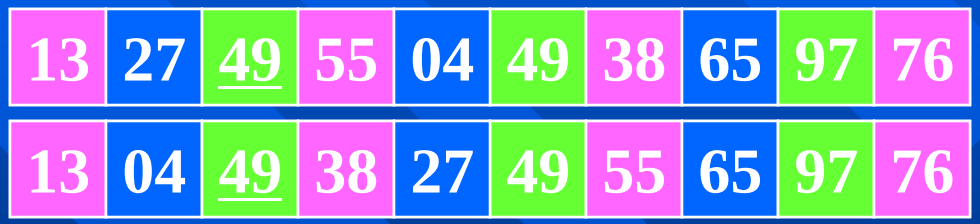
# §9.2 插入分类方法——SHELL分类

2、举例：


第1趟，$d_1=5$

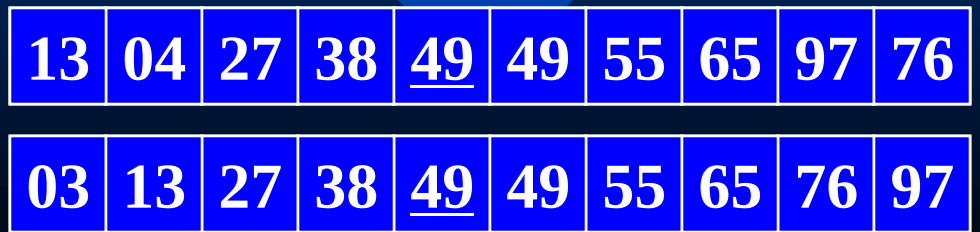



第2趟，$d_2=3$




第3趟，$d_3=2$


第4趟，$d_4=1$



# §9.2 插入分类方法——SHELL分类

## 3、算法

略

## 4、效率分析:

我们知道：

一方面，开始几趟，组内元素少（步长大，调整幅度大），插入分类的效率是高的；　另一方面，越往后，组内元素越多，但是经过前几趟，元素越来越有序，效率也比较高。

但是，SHELL分类的效率分析是很复杂的，因为它是“增量”序列的函数，目前还是数学上的难题。

# §9.2 插入分类方法——SHELL分类

*部分结论：*

（1）$d[k]=2^{t-k+1}-1$时，时间为$o(n^{3/2})$，其中t是趟数 $1\le k\le t\le \lfloor \log_2(n+1) \rfloor$

（2）n在某个特定范围内，SHELL分类所需的比较和移动次数约为$n^{1.3}$，$n\to\infty$时，可减少到$n(\log_2 n)^2$

（3）增量序列有多种取法，但是应注意：应使增量序列中的值没有除1之外的公因子，并且最后一个增量值必须等于1。

# §9.3 交换分类方法

## 一、交换分类方法的基本思想

假设待分类记录集合为 **R1,R2,...Rn**，简记为**R[1..n]**。交换分类方法由多趟组成，假设要进行某一趟，它是借助对无序序列中的记录进行“交换”的操作，将无序序列中某关键字（最大、最小或其它）的记录“交换”到该记录应该在的位置上。

根据比较交换的方法不同，交换分类也分为很多方法：

- 标准交换分类（冒泡分类）
- 成对交换分类（奇偶交换分类）
- 穿梭分类
- 快速分类

# §9.3 交换分类方法

## 二、冒泡分类方法

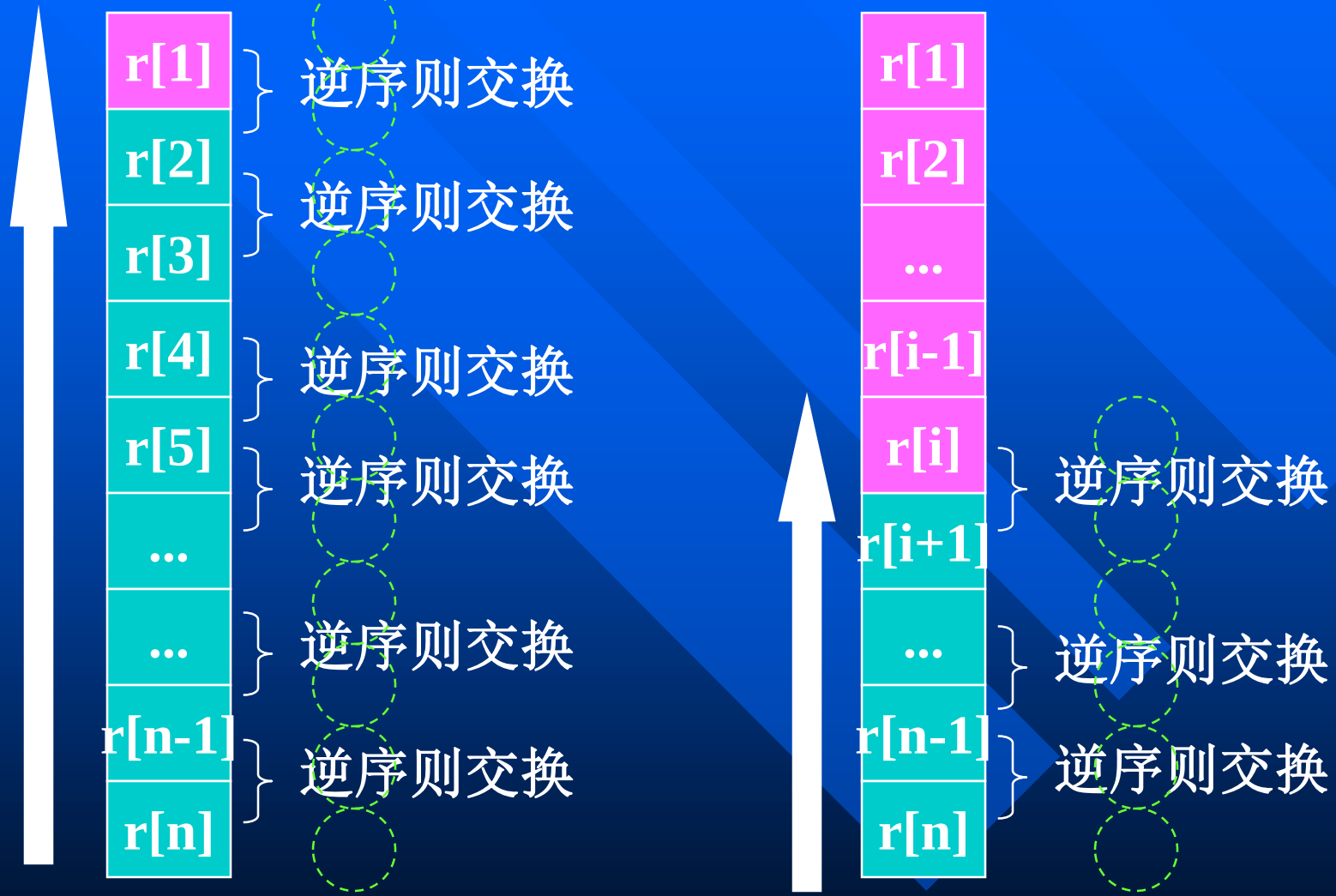
1、基本思想：　整个分类过程由多趟组成：

第1趟：　r[n]与r[n-1]比较，逆序则交换；
r[n-1]与r[n-2]比较，逆序则交换；
…… ……
r[2]与r[1]比较，逆序则交换；
于是：*r[1]是无序序列中最小的！*

第i趟：　r[n]与r[n-1]比较，逆序则交换；
r[n-1]与r[n-2]比较，逆序则交换；
…… ……
r[i+1]与r[i]比较，逆序则交换；
于是：*r[i]是无序序列中最小的！*

重复，直到某趟中没有逆序发生为止

# §9.3 交换分类方法——冒泡分类方法



# §9.3 交换分类方法——冒泡分类方法

2、举例：略

3、算法：略

4、效率分析：

最好：若序列已经正序排列时，仅需进行一趟，n-1次比较没有交换；

最坏：若序列逆序排列时，需进行n-1趟：

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

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

平均：$O(n^2)$

# §9.3 交换分类方法

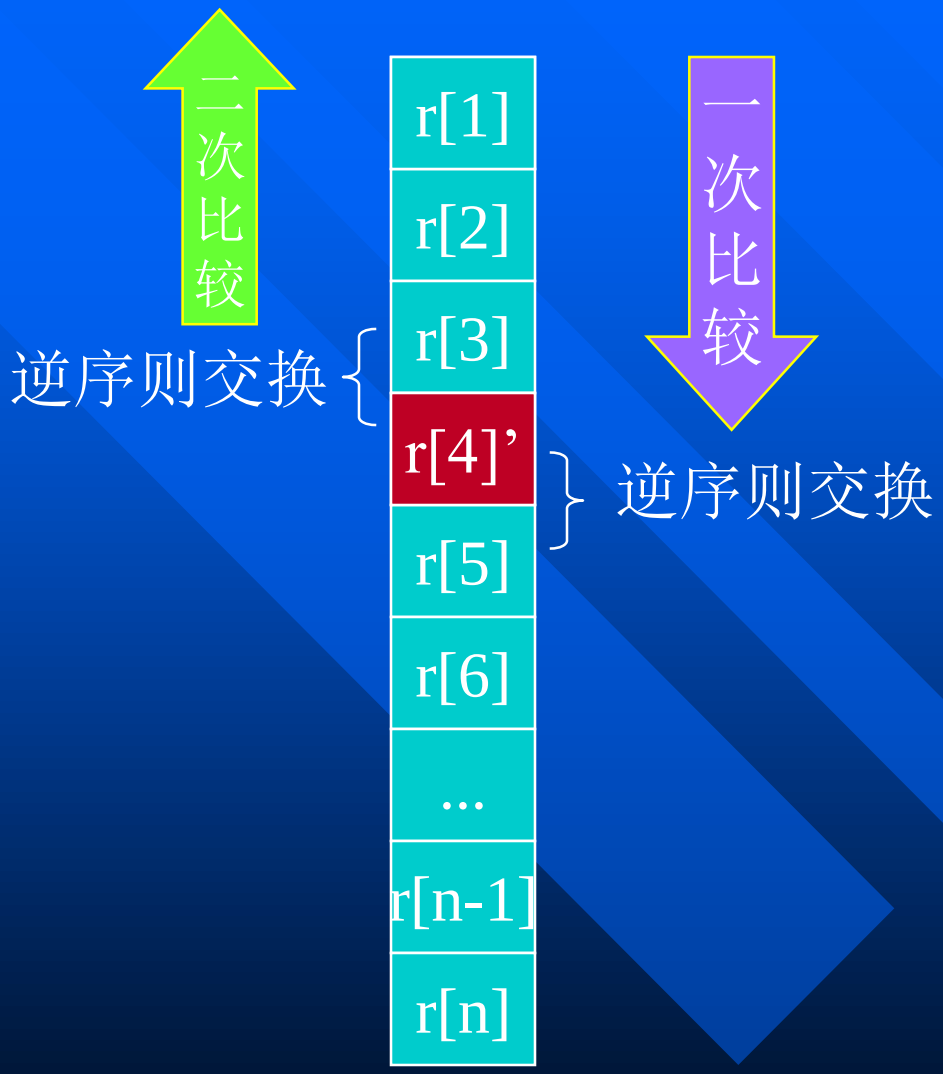
## 二、穿梭（交换）分类方法

1、基本思想：　整个分类过程只有一趟，但是这一趟时进时停，具体地：

一次比较：r[1]与r[2]比较，正序则一次比较继续，r[2]与r[3]比较，……，
若r[i]与r[i+1]比较出现逆序，则交换，一次比较停止，转而进行二次比较；

二次比较：一次比较逆序交换后，r[i]与r[i-1]比较，逆序则交换，然后r[i-1]与r[i-1]比较，直到正序，然后继续一次比较（从一次比较停止的位置），即发现小元素，尽可能向上走；

# §9.3 交换分类方法——穿梭（交换）分类方法



# §9.3 交换分类方法

## 五、快速分类方法

1、基本思想：　在待分类序列中指定一个元素，它称为“轴”元素，然后经过一些操作把轴元素安置好，即把它安置在排好序后应该在的位置，亦即，它不小于前面的元素，不大于后面的元素。安置好的轴元素将分类序列分为左右两部分，对这两部分利用同样的策略进行分类（递归）。

# §9.3 交换分类方法—快速分类方法

具体地:

假设待分类序列为 {r[s],r[s+1],...,r[t]}，整个分类有多趟组成：一趟过程如下：

（1）首先任意选取一个记录作为轴元素，一般选取序列的第1个元素。

（2）重新排列其余元素，凡其关键字小于枢轴的记录均移动至该记录之前，反之，凡关键字大于枢轴的记录均移动至该记录之后。从而得到轴元素的所在位置i。

（3）安置轴元素，轴元素将原序列分为两个子序列：
{r[s],r[s+1],...,r[i-1]} {r[i+1],r[i+2],...,r[t]}

（4）分别对分割所得两个子序列进行快速排序，依次类推，直至每个子序列中只含一个记录为止。

# §9.3 交换分类方法——快速分类方法

## 2、确定轴元素的位置: 设轴元素为序列的第1个元素


附设两个指针i,j，初始值分别为s,t，轴元素 $r_p=r[s]$

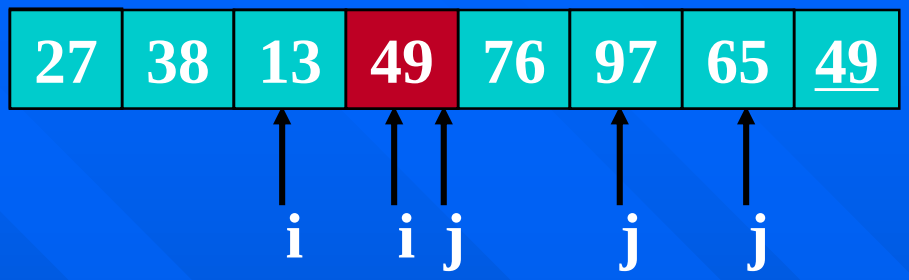
j从当前位置向前搜索找到第1个比轴元素小的记录，把该元素交换到前面；(与$r_p$交换)

i从当前位置向后搜索找到第1个比轴元素大的记录，把该元素交换到后面；(与$r_p$交换)

可以发现，轴元素左、右跳跃，最后落在最终位置上！而实际上前面的交换都是多余的，只要找到最终的位置把$r_p$放置到最后的位置即可。

# §9.3 交换分类方法——快速分类方法

## 2、举例：

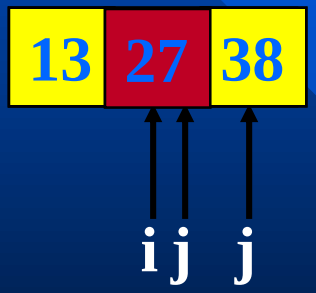



第一趟： 轴元素 x=49

安置轴元素：

结果：


第二趟：




轴元素x=27

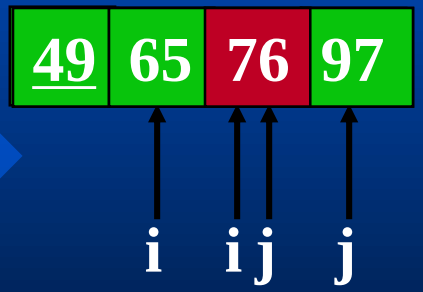
安置轴元素：

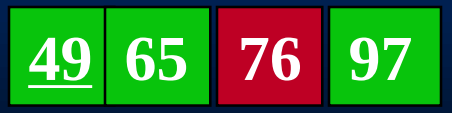
结果：


轴元素x=76

安置轴元素：

结果：



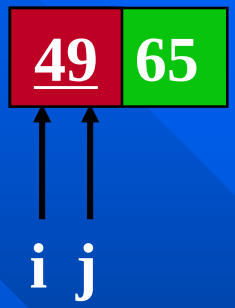
# §9.3 交换分类方法——快速分类方法

上一趟后结果：


第三趟：

轴元素x=49

安置轴元素：




最终结果：

# §9.3 交换分类方法——快速分类方法

## 3、算法 略

## 4、效率分析：

该算法是一个递归算法，按照递归算法的时间复杂性分析方法，假设一次划分所得枢轴位置i=k，则对n个记录进行快排所需时间可由递归方程表示：

$$T(n) = T_{pass}(n)+T(k-1)+T(n-k)$$

其中 $T_{pass}(n)$为对n个记录进行一次划分所需时间；

若待排序列中记录的关键字是随机分布的，则k取1至n中任意一值的可能性相同，由此可得快速排序所需时间的平均值为：

$$T_{avg}(n) = Cn + \frac{1}{n}\sum_{k=1}^{n}\left[T_{avg}(k-1) + T_{avg}(n-k)\right]$$

$$= Cn + \frac{2}{n}\sum_{i=0}^{n-1} T_{avg}(i)$$

# §9.3 交换分类方法——快速分类方法

设 Tavg(0)≤b 设 Tavg(1)≤b，则当k=2(b+c)和n=2时有：

Tavg(n) ≤kn ln n

***最坏情况：*** 当待排序序列基本有序时，快速分类蜕化为冒泡分类，时间复杂性为$O(n^2)$。例如：

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|

***最好情况：*** 每次都将轴元素安置在序列的中间，序列在最快的时间内蜕化为长度是 1 的序列。O(nlogn)，例如：

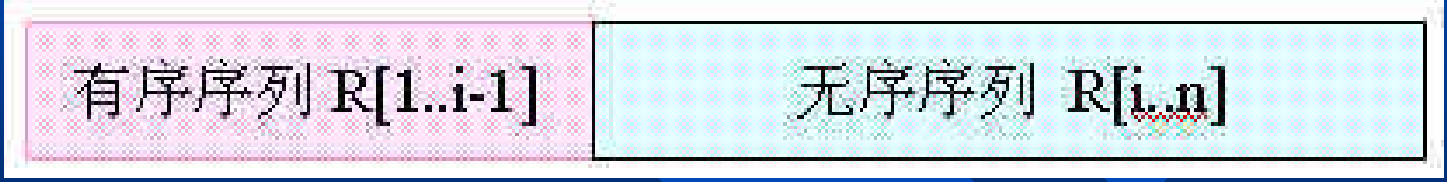
| 4 | 1 | 3 | 2 | 6 | 5 | 7 |
|---|---|---|---|---|---|---|

空间：递归需要辅助栈空间，栈的最大深度为n，最小深度为logn。

# §9.4 选择分类方法

## 一、选择分类方法的基本思想

假设待分类记录集合为 R1,R2,...Rn，简记为R[1..n]。选择分类方法由多趟组成，假设要进行某一趟，它是在当前无序序列中选择出“最小”或“最大”的记录，然后将它加入到有序序列中。




每一趟在n-i+1个记录中选取关键字最小的记录作为有序序列中的第i个记录。

# §9.4 选择分类方法

根据选择最小或最大元素的方法不同，选择分类也分为很多方法：

简单选择分类

树选择分类（锦标赛分类）

堆分类

# §9.4 选择分类方法

## 一、简单选择分类方法

1、基本思想：每次从无序序列中采用简单选择方法选择最小的元素。

具体地，整个分类共有n-1趟：

第 1 趟，r[1]与r[2], r[3], ..., r[n]比较，得到最小元素放置在r[1]中；

第 2 趟，r[2]与r[3], r[4], ..., r[n]比较，得到最小元素放置到r[2]中；

…… ……

第n-1趟，r[n-1]与r[n]比较，得到最小元素，放置到r[n-1]中；

# §9.4 选择分类方法—简单选择分类方法

2、举例：略

3、算法：略

4、效率分析

对n个记录进行简单选择排序，所需进行的关键字间的比较次数总计为：

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

移动记录的次数，最小值为0, 最大值为3(n-1)

# §9.4 选择分类方法

## 二、锦标赛（树型）选择分类方法

1、基本思想：

用更快的方法选出最小元素，方法是：

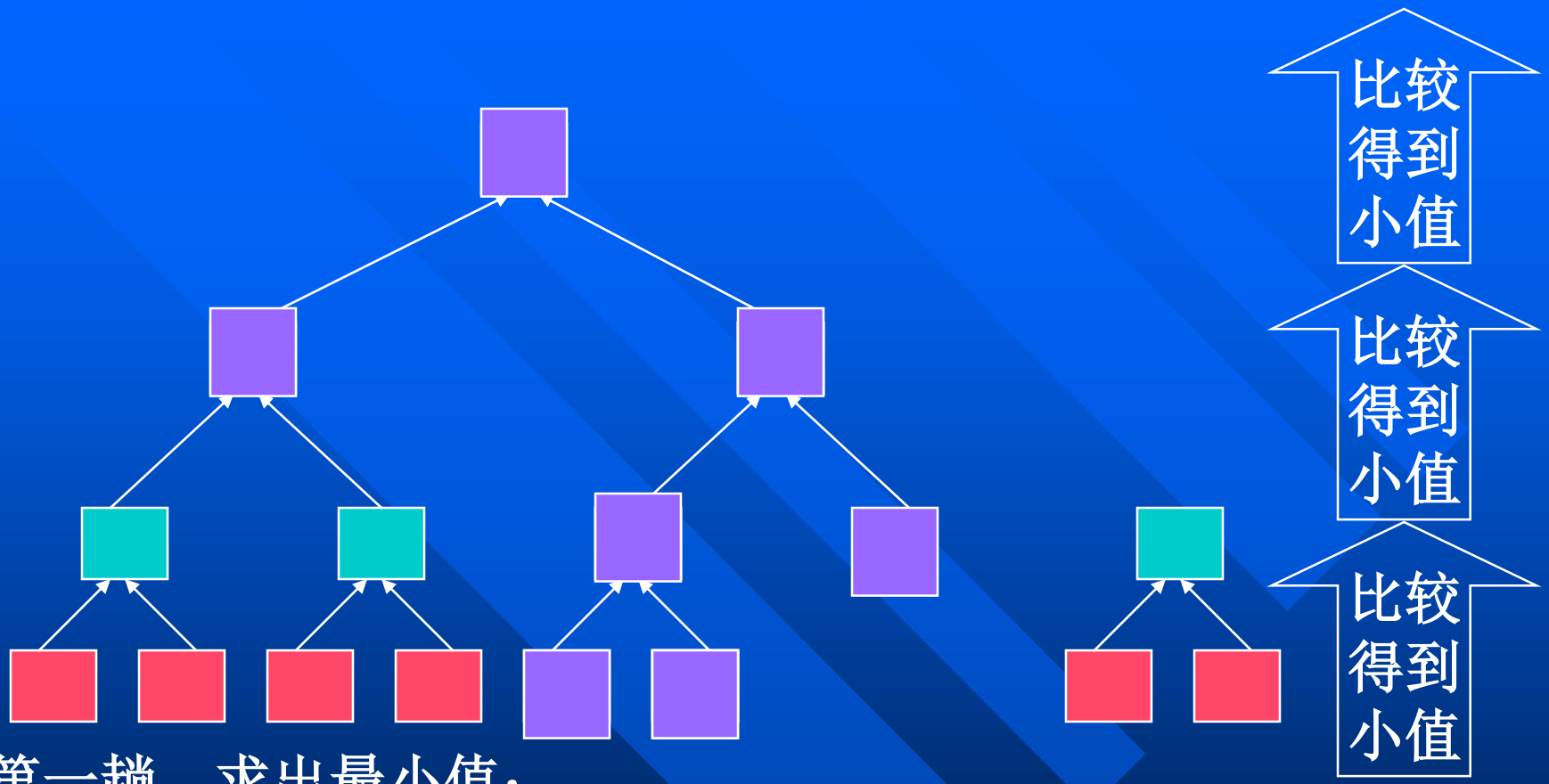
首先，对n个记录的关键字进行两两比较，然后在其中[n/2]个较小者之间再进行两两比较，如此重复，直至选出最小关键的记录为止。

然后，根据关系的可传递性，将叶子结点中的最小关键字改为“最大值”，然后从该叶子结点开始，和其左（或右）兄弟的关键字比较，修改从叶子结点到根的路径上各结点的关键字，则根结点关键字即为次小的关键字。

同理，可以依此从小到大找出所有关键字。

参见下图：

# §9.4 选择分类方法—树型选择分类方法



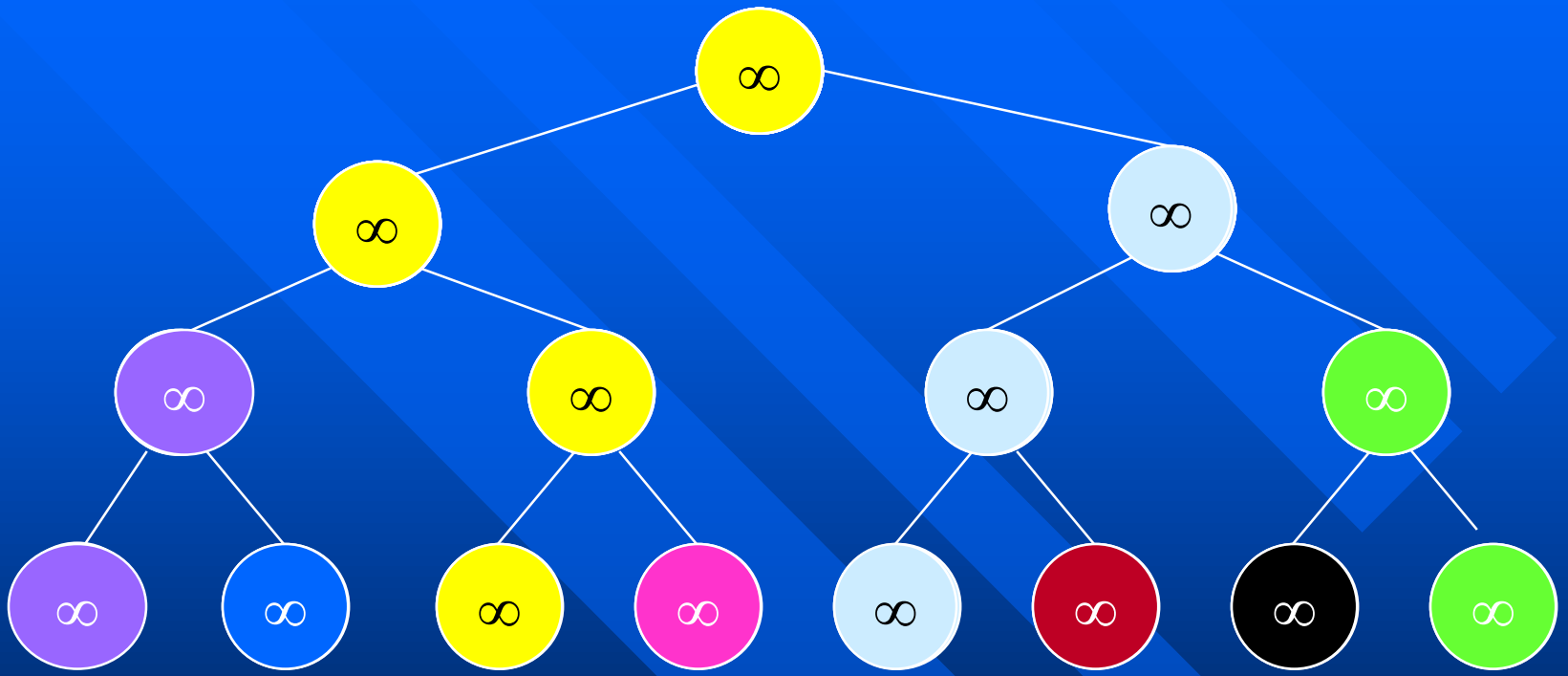

第一趟，求出最小值；

第二趟，沿最小值产生的分支，把对应的叶子值变为∞，然后从该叶子开始，和其左或右兄弟关键字比较，修改从叶子结点到根的各结点的关键字，得到次小关键字记录

第3----n-1趟，与第二趟类似，

# §9.4 选择分类方法——树型选择分类方法

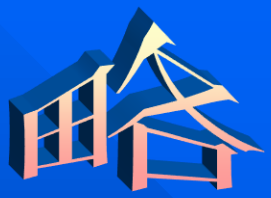
2、举例：　49 38 65 97 76 13 27 49




| 13 | 27 | 38 | 49 | 49 | 65 | 76 | 97 |
|---|---|---|---|---|---|---|---|

# §9.4 选择分类方法—树型选择分类方法

## 3、算法



## 4、效率分析

时间：
第1趟，求最小值的比较次数为：n/2+n/4+n/8+...+2+1；
第2—n-1趟，比较次数为logn；
因此，时间复杂性为：O(nlogn)；
空间：有较多的辅助空间，n-1个（2n-1-n）

# §9.4 选择分类方法

## 三、堆分类方法

1、堆的定义和特点：

堆的特点：

若序列是最小堆（小顶堆），则$K_1$必是序列中的最小值；
若序列是最大堆（大顶堆），则$K_1$必是序列中的最大值；

# §9.4 选择分类方法—堆分类方法

## 2、堆分类的基本思想：

设待分类序列为r[1],r[2],r[3],...,r[n]，根据堆的性质，把该序列调整为堆（最大堆或最小堆），则堆顶元素为最大（最小）值，把该元素加入到有序序列中；对剩余的无序序列，再调整为堆，得到次大元素，加入到有序序列中，.....，依次下去，直到无序序列只有一个元素为止。具体地，整个分类有n-1趟：

第 1 趟，将原始待分类序列*调整*为堆，求出最小值，加入到有序序列中；

第 i 趟，把剩余的元素序列*再调整*为堆，取堆顶元素，加入到有序序列中。

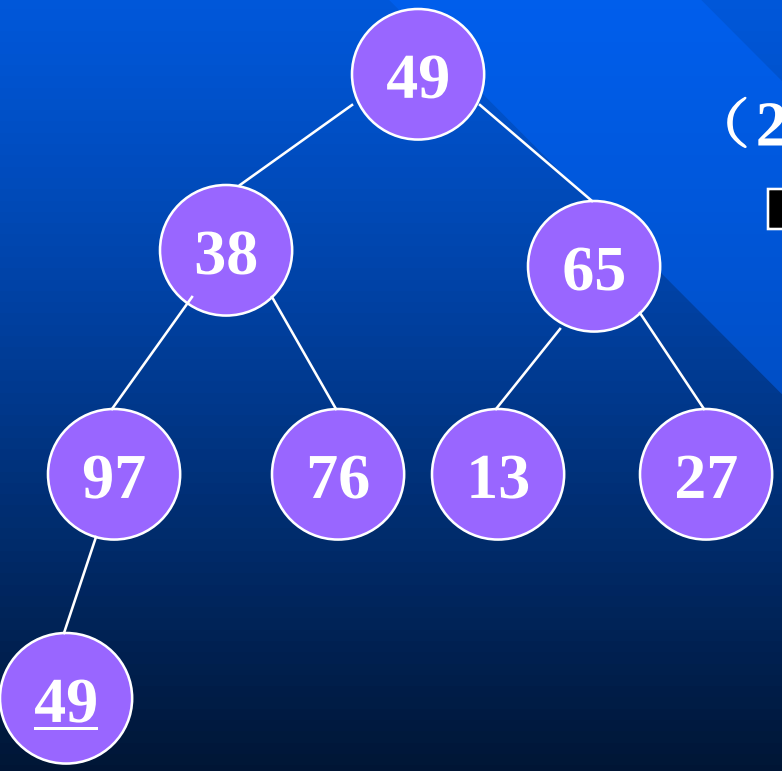
堆分类的关键是序列如何调整成为堆！

# §9.4 选择分类方法——堆分类方法

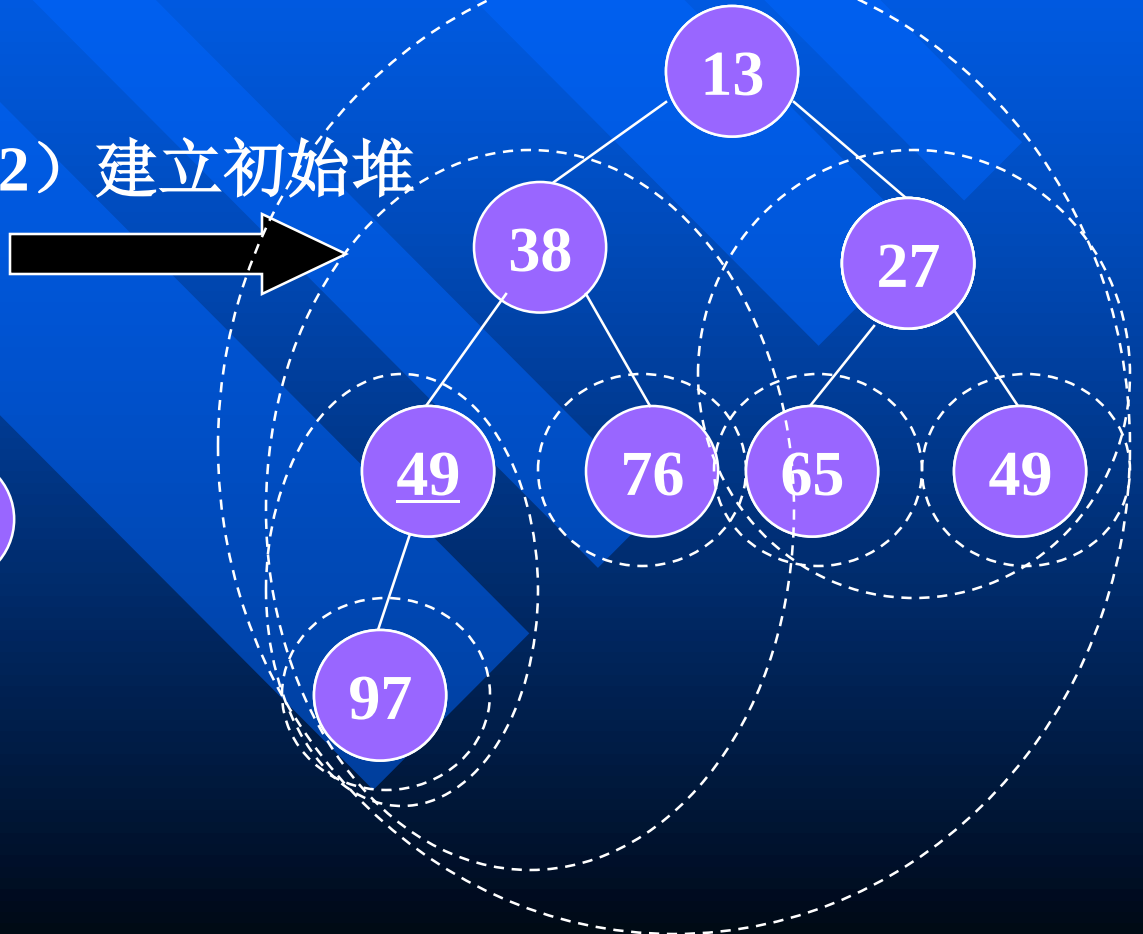
3、堆分类的算法：堆的调整前面已经讲过，略！

4、举例：{49，38，65，97，76，13，27，49}

（1）建立完全二叉树

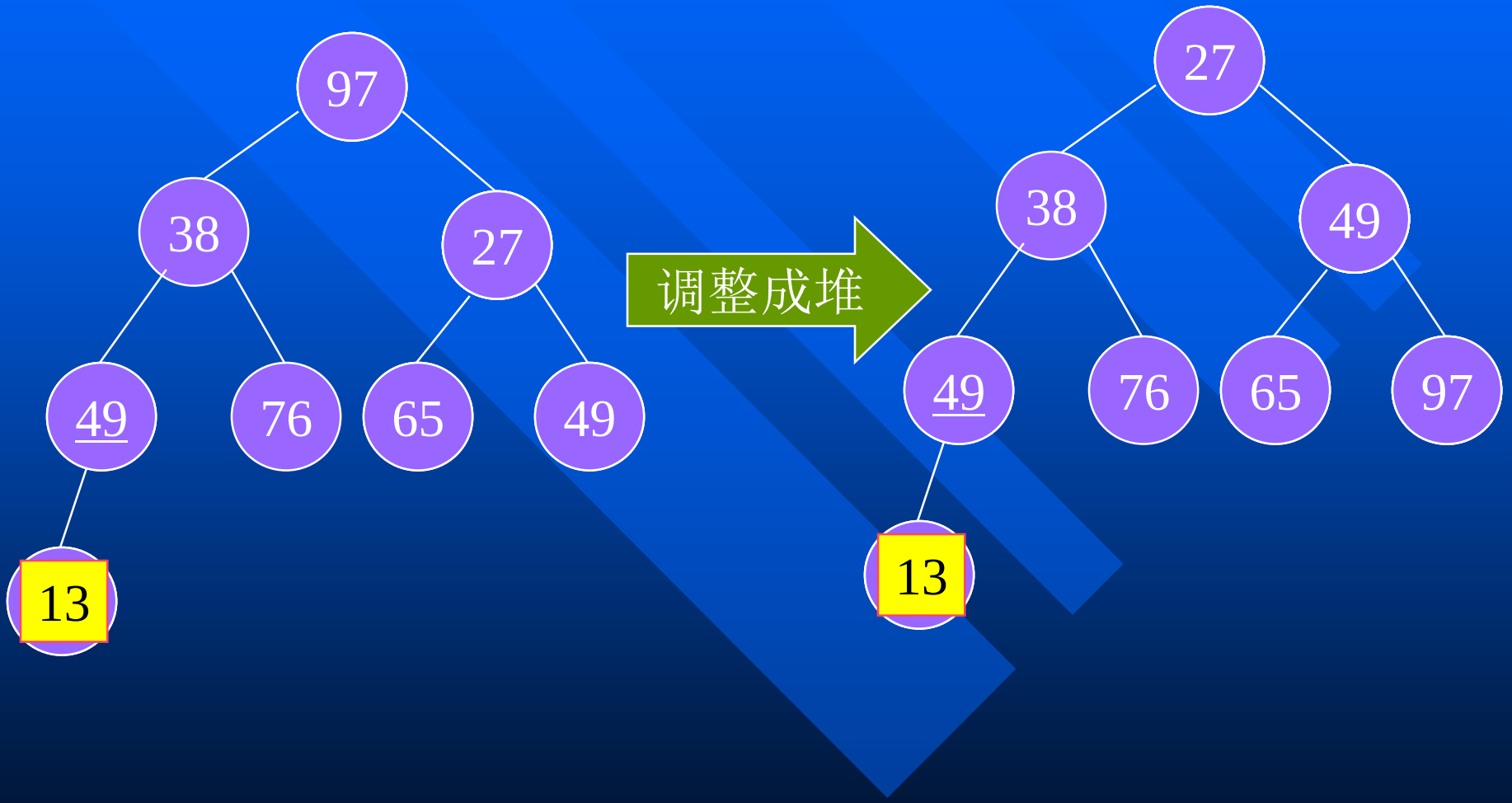



（2）建立初始堆

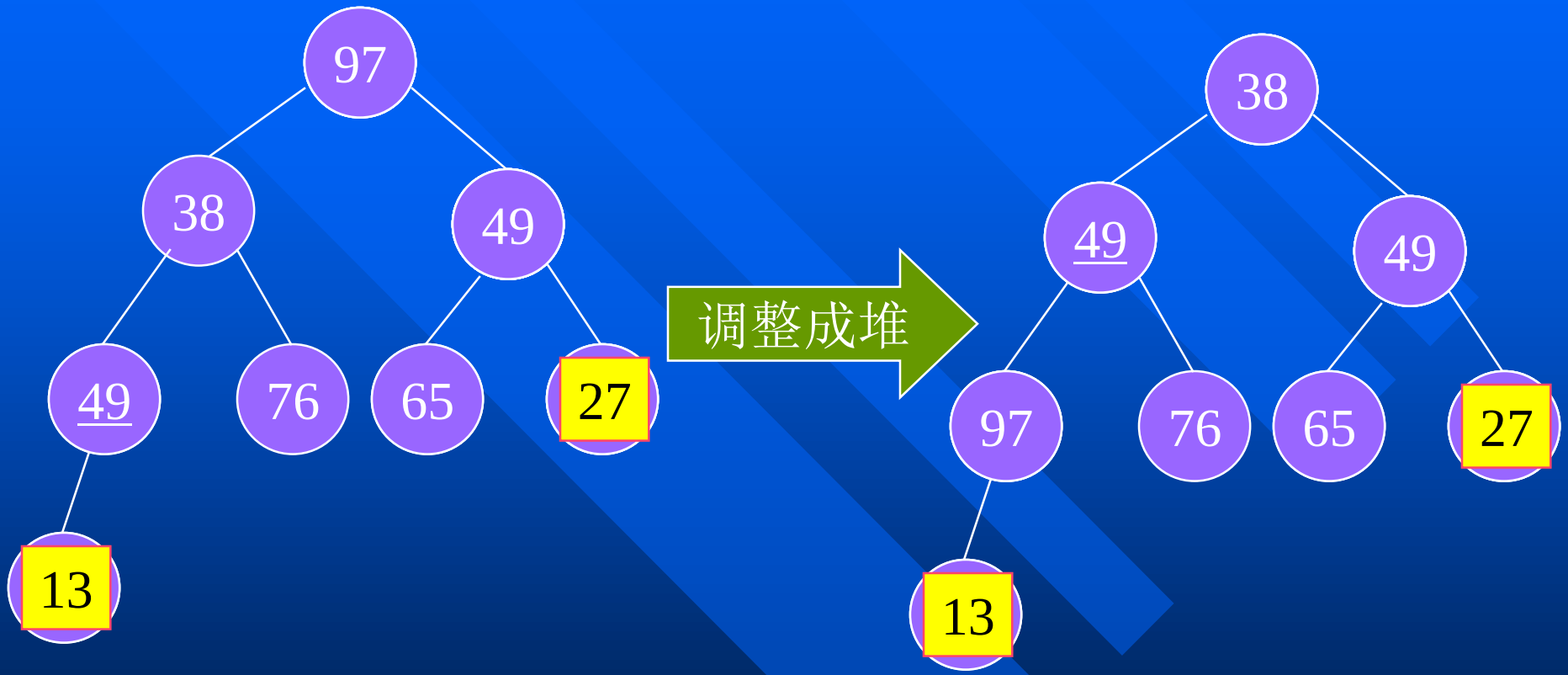


# §9.4 选择分类方法——堆分类方法

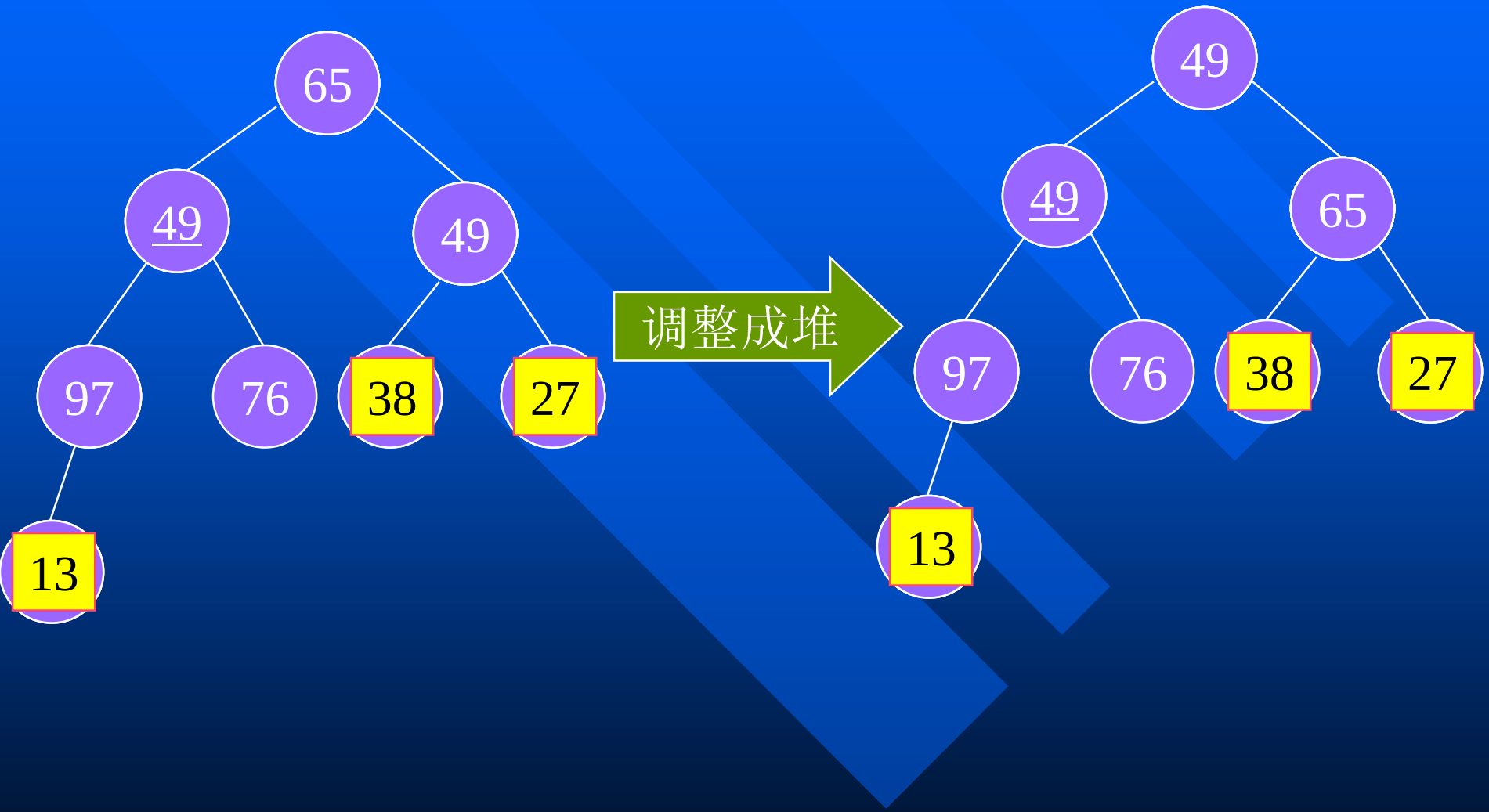
（3）排序过程：不断地选出最小元素的过程



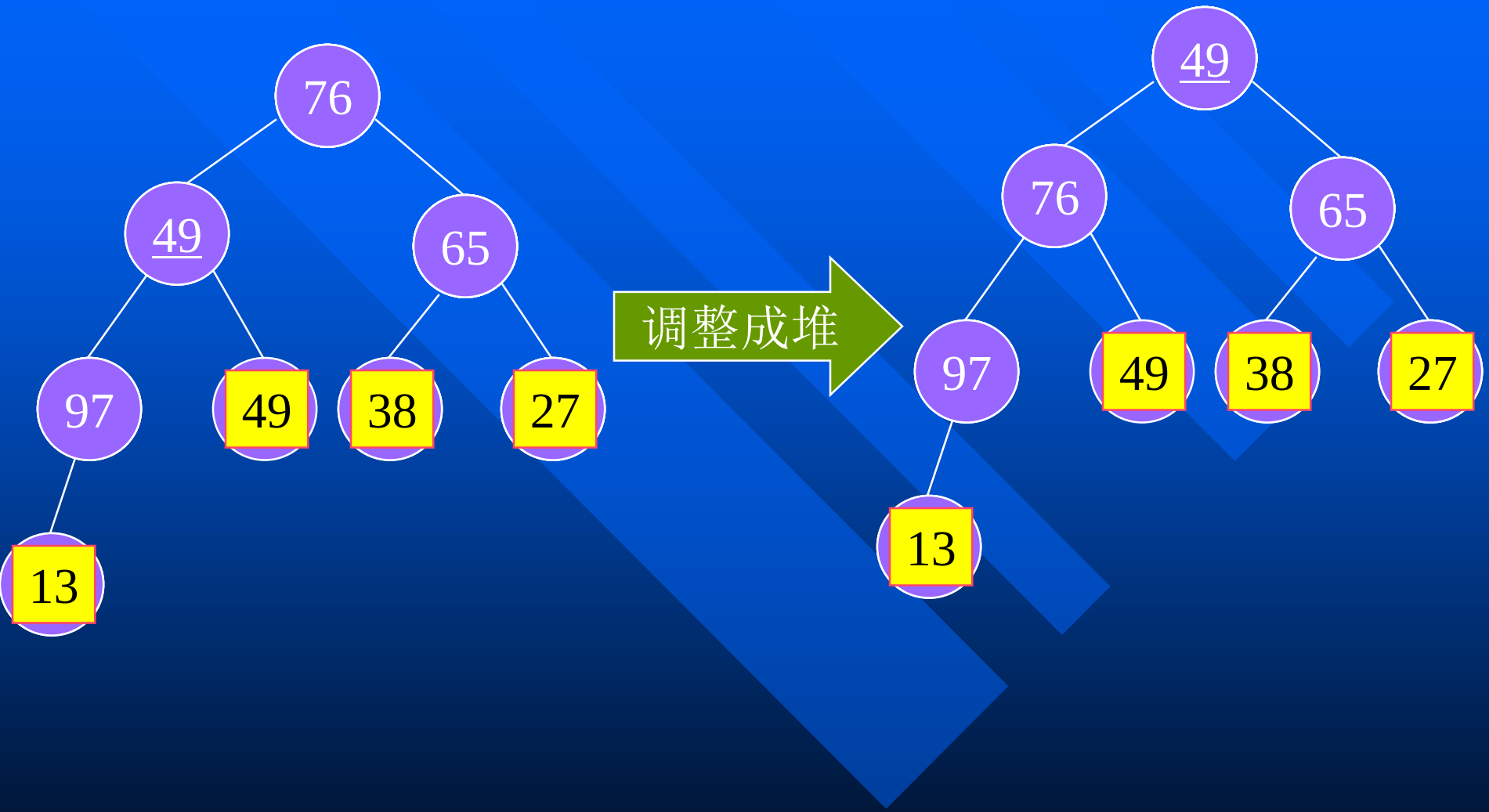
# §9.4 选择分类方法——堆分类方法



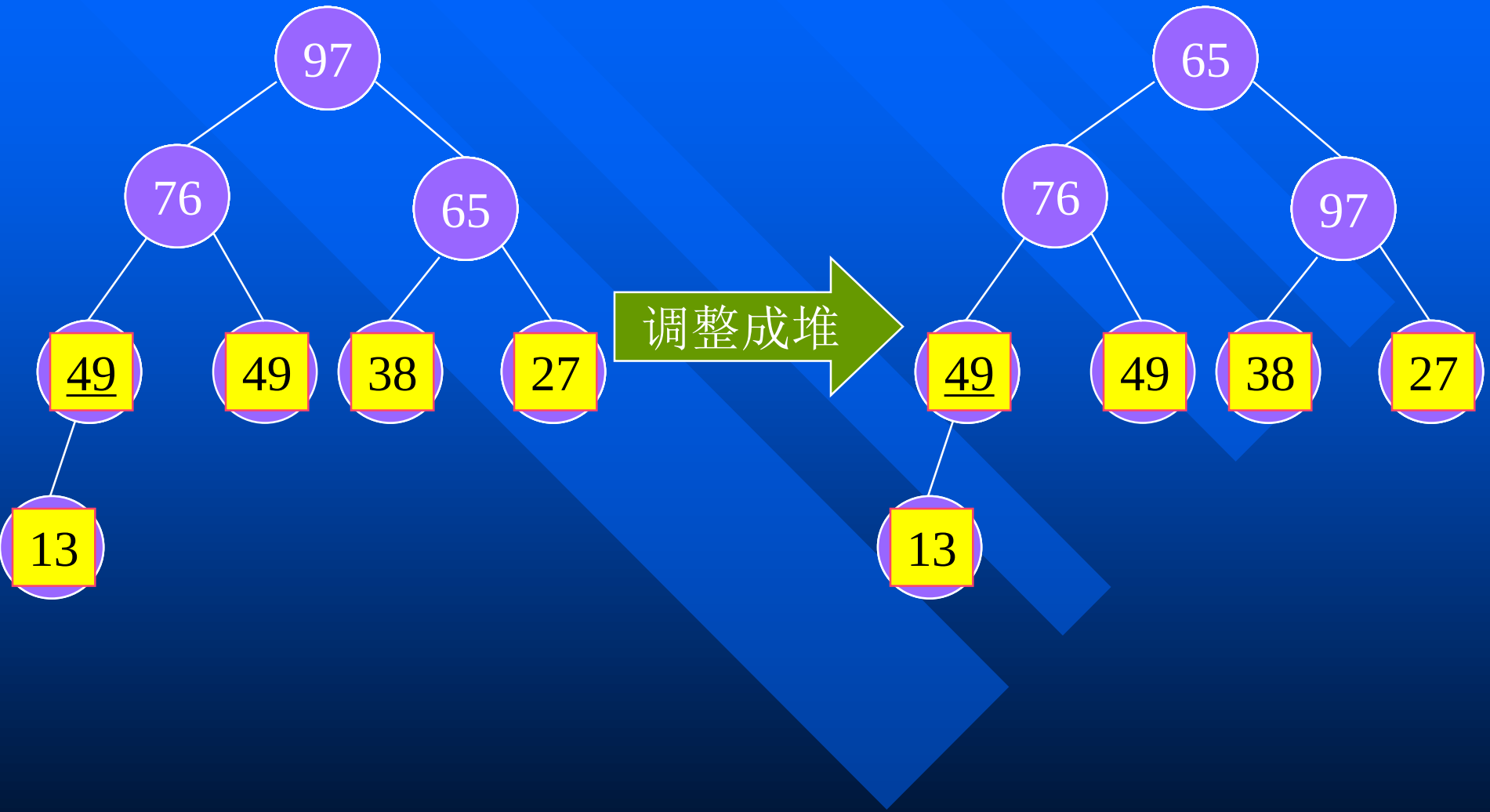
# §9.4 选择分类方法——堆分类方法



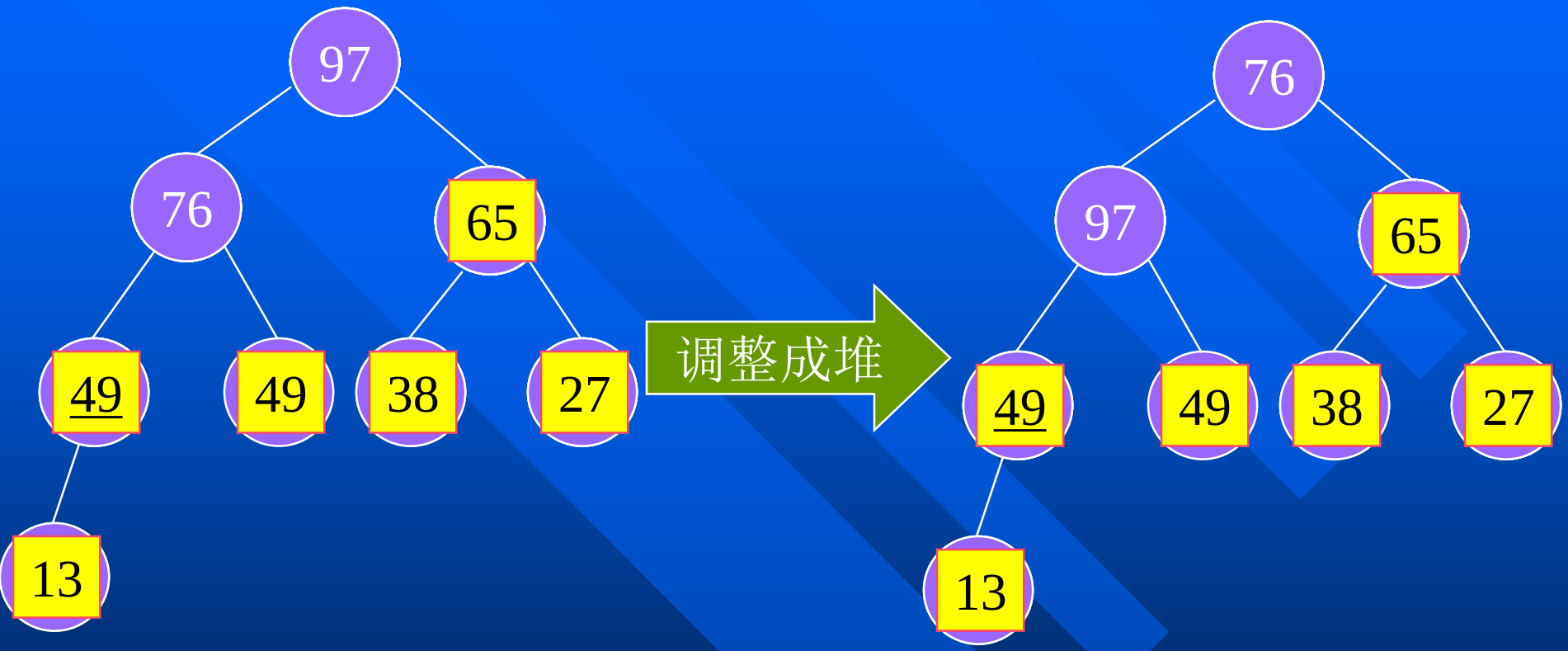
# §9.4 选择分类方法——堆分类方法



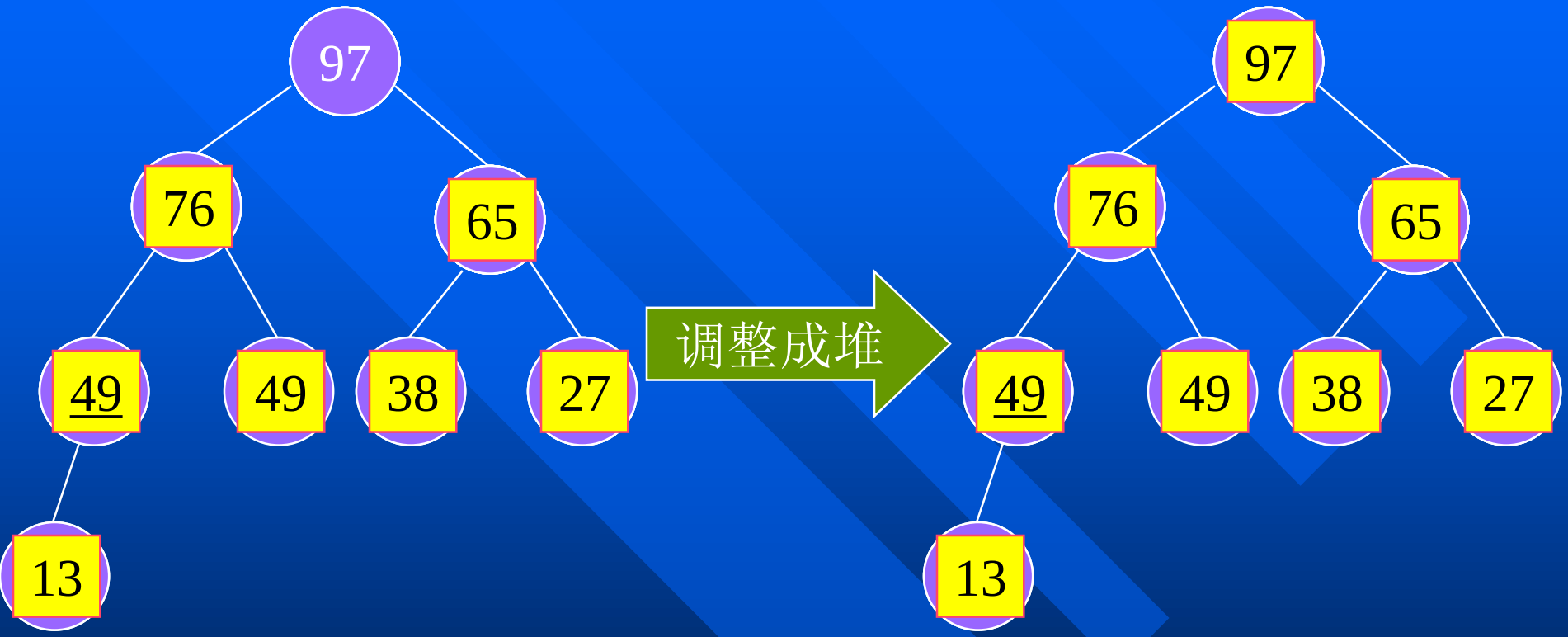
# §9.4 选择分类方法——堆分类方法



# §9.4 选择分类方法——堆分类方法



# §9.4 选择分类方法—堆分类方法




分类结束：13, 27, 38, 49, 49, 65, 76, 97

# §9.4 选择分类方法—堆分类方法

## 4、效率分析:

1、对深度为k的堆，“筛选”所需进行的关键字比较的次数至多为2(k-1);

2、对n个关键字，建成深度 $h(=\lfloor \log_2 n \rfloor+1)$ 为的堆，所需进行的关键字比较的次数至多为4n;

3、调整“堆顶”n-1次，总共进行的关键字比较的次数不超过：

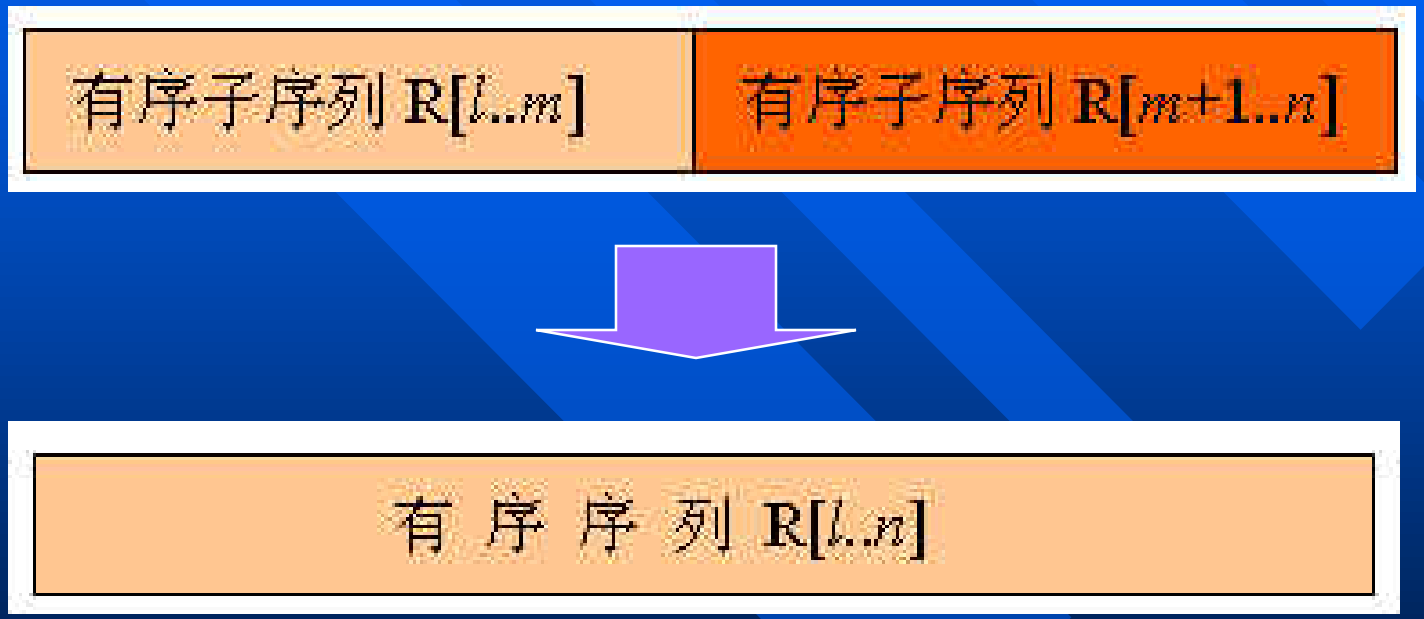
$$2(\lfloor \log_2(n-1) \rfloor+\lfloor \log_2(n-2) \rfloor+ \ \ldots+\log_2 2)\leq 2n(\lfloor \log_2 n \rfloor)$$

因此：堆排序的时间复杂度为O(nlogn)

# §9.5 归并分类方法

## 一、归并分类方法的基本思想

将两个或两个以上的有序子序列“归并”为一个有序序列。




常见的归并分类方法有2-路归并、多路归并等方法；

# §9.5 归并分类方法

## 二、2-路归并分类方法

1、基本思想：

待分类序列为r[1],r[2],...,r[n]。开始，每个元素看作一个有序序列，分类分为$\log_2 n$趟：

第1趟：r[1]与r[3]，r[3]与r[4],...,两两归并，得到n/2个有序序列；

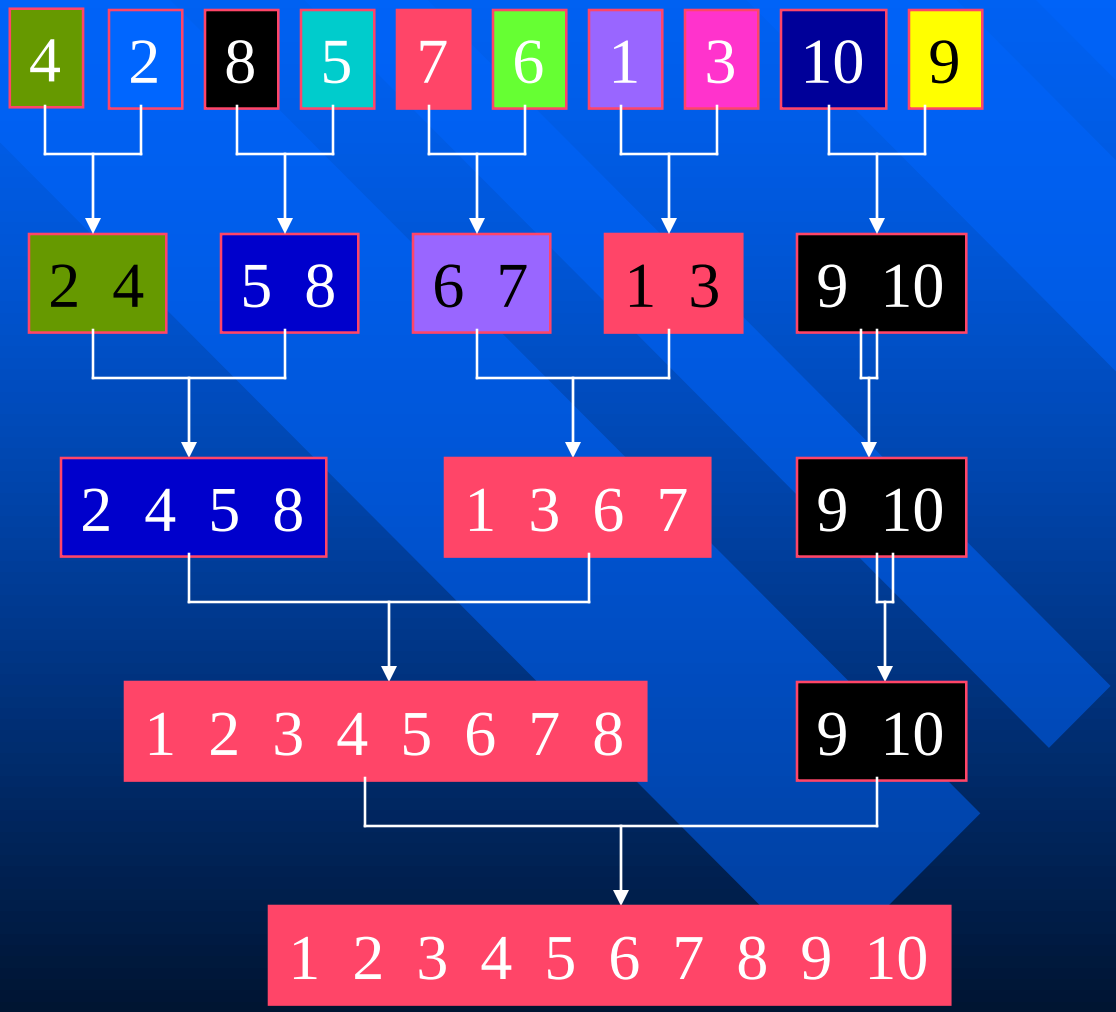
第2趟：对上一趟的n/2有序序列，再两两归并，得到n/4个有序序列；

…… ……

最后一趟：归并两个有序序列，得到最终的有序序列。

# §9.5 归并分类方法——2-路归并分类方法

2、举例： {4,2,8,5,7,6,1,3,10,9}



# §9.5 归并分类方法——2-路归并分类方法

## 3、算法：

略

## 4、效率分析：

（1）归并两个长度为m,n的有序序列，最大比较次数为m+n;

（2）归并长度为n的序列，共需要进行$\log_2 n$;

时间复杂性：$O(n\log_2 n)$

空间复杂性：O(n)，即r2的大小；

# §9.6 各种分类方法综合比较

## 一、时间性能

1、按平均的时间性能来分，有三类排序方法：

- 时间复杂度为O(nlogn)的方法有：快速排序、堆排序和归并排序，其中以快速排序为最好
- 时间复杂度为$O(n^2)$的有：直接插入排序、起泡排序和简单选择排序，其中以直接插入为最好，特别是对那些对关键字近似有序的记录序列尤为如此；
- 时间复杂度为O(n)的排序方法只有，基数排序。

2、按平均的时间性能来分，有三类排序方法：

当待排记录序列按关键字顺序有序时，直接插入排序和起泡排序能达到O(n)的时间复杂度;而对于快速排序而言，这是最不好的情况，此时的时间性能蜕化为$O(n^2)$，因此是应该尽量避免的情况

# §9.6 各种分类方法综合比较

3、简单选择排序、堆排序和归并排序的时间性能不随记录序列中关键字的分布而改变。

## 二、空间性能

指的是排序过程中所需的辅助空间大小。

1、所有的简单排序方法(包括：直接插入、起泡和简单选择)和堆排序的空间复杂度为O(1)；

2、快速排序为O(logn )，为栈所需的辅助空间；

3、归并排序所需辅助空间最多，其空间复杂度为O(n )；

4、链式基数排序需附设队列首尾指针，则空间复杂度为O(rd )

# §9.6 各种分类方法综合比较

## 三、排序方法的稳定性能

稳定的排序方法指的是，对于两个关键字相等的记录，它们在序列中的相对位置，在排序之前和经过排序之后，没有改变。

1、当对多关键字的记录序列进行LSD方法排序时，必须采用稳定的排序方法。

2、对于不稳定的排序方法，只要能举出一个实例说明即可。

3、快速排序和堆排序是不稳定的排序方法。

*END*