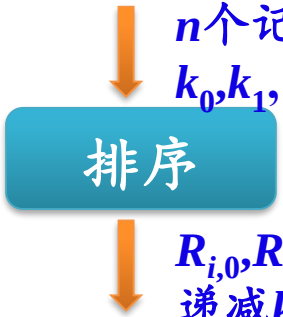
第10章　内 排 序

10.1 排序的概念

1、排序的定义

所谓排序，是整理表中的记录，使之按关键字递增（或递减）有序排列：

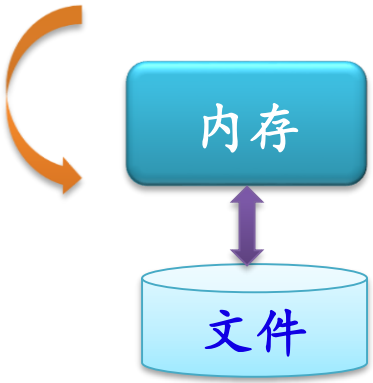


说明：排序数据中可以存在相同关键字的记录。本章仅考虑递增排序。

2、内排序和外排序

在排序过程中，若整个表都是放在内存中处理，排序时不涉及数据的内、外存交换，则称之为内排序；

反之，若排序过程中要进行数据的内、外存交换，则称之为外排序。



3、内排序的分类

- 内排序
  - 基于比较的排序算法
    - 插入排序
    - 交换排序
    - 选择排序
    - 归并排序
  - 不基于比较的排序算法
    - 基数排序

基于比较的内排序算法最快有多快？

假设有3个记录(R_1,R_2,R_3)，对应的关键字为(k_1,k_2,k_3)。

初始数据序列有 3! = 6 种情况：

- 1，2，3
- 1，3，2
- 2，1，3
- 2，3，1
- 3，1，2
- 3，2，1

📖 *n*个记录，初始数据序列有*n*!种情况

以(R_1,R_2,R_3) =（2,3,1）为例，一种基于比较的排序方法：

$R_1\ R_2\ R_3$

2　3　1

↓ $R_1 < R_2$为真，不交换

$R_1\ R_2\ R_3$

2　3　1

↓ $R_2 < R_3$为假，R_2、R_2交换

$R_1\ R_3\ R_2$

2　1　3

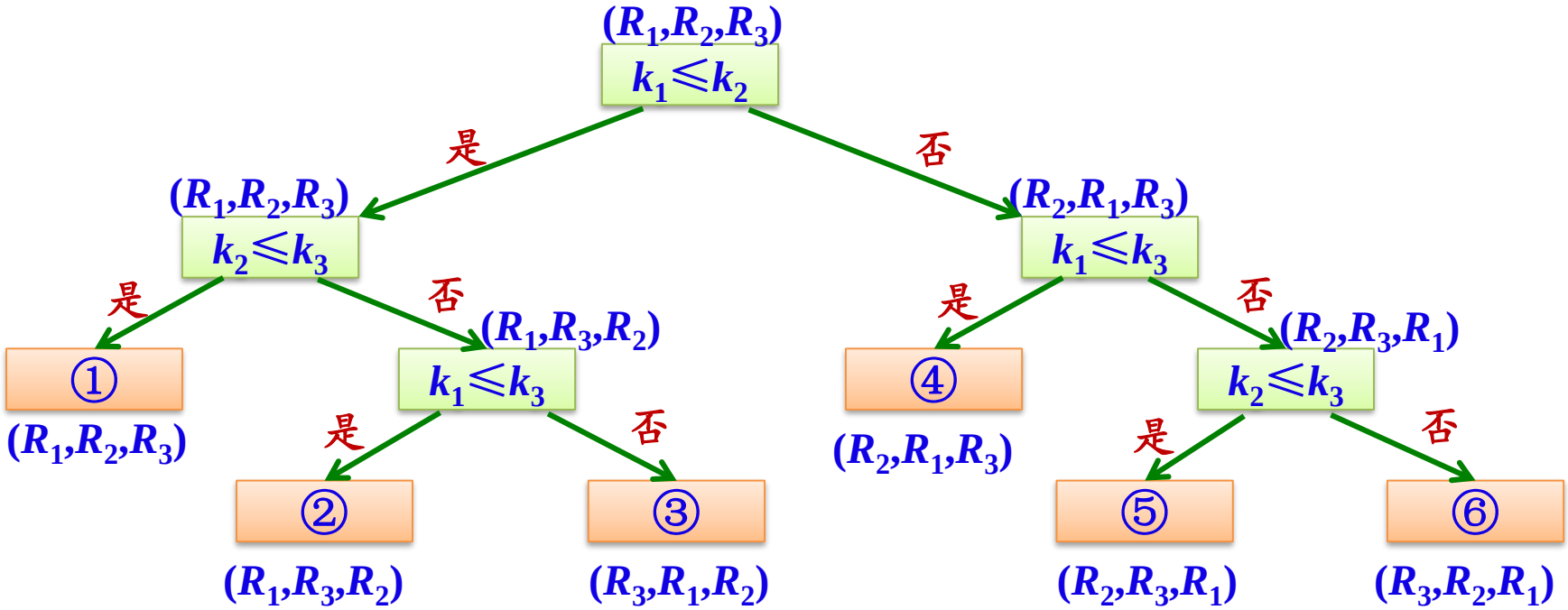
↓ $R_1 < R_3$为假，R_1、R_2交换

$R_3\ R_1\ R_2$

1　2　3

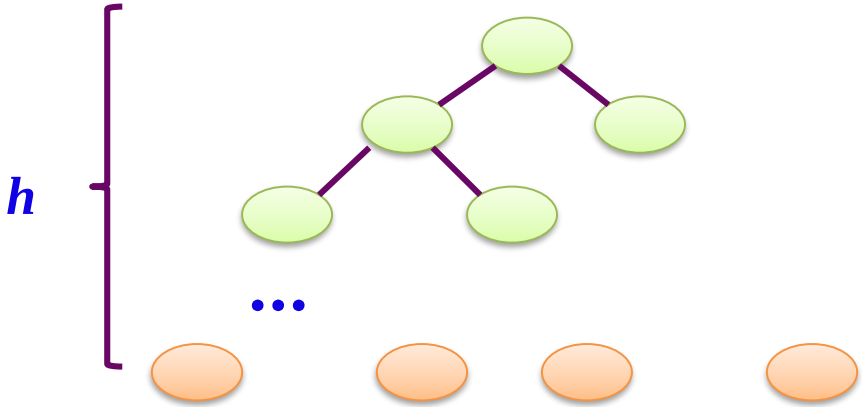
总共3次关键字比较

所有可能的初始序列的排序过程构成一个决策树：



决策树是一棵有$n!$个叶节点的二叉树。

决策树可以近似看成是一颗高度为h，叶节点个数为$n!$的满二叉树。



- 叶节点个数=$n!$
- 总节点个数=$2n!-1$
- $h=\log_2(\text{总节点个数}+1)=\log_2(n!)\approx n\log_2 n$
- 平均关键字比较次数=$h-1$
- 移动次数也是同样的数量级，即这样的算法最坏时间复杂度为$O(n\log_2 n)$。
- 同样可以证明平均时间复杂度也为$O(n\log_2 n)$。

结论：

n个记录采用基于比较的排序方法：

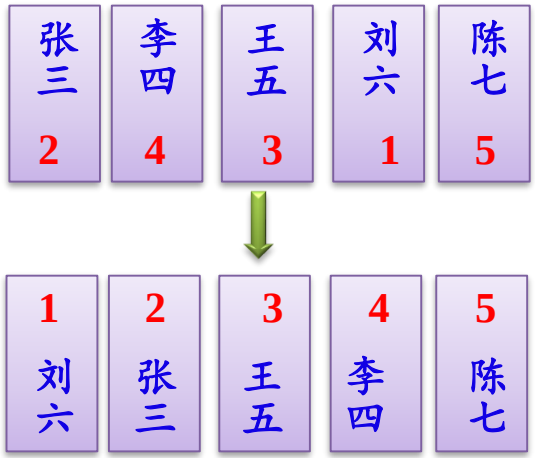
- 最好的平均时间复杂度为$O(n\log_2 n)$。
- 最好情况是排序序列正序，此时的时间复杂度为$O(n)$。

思考题

对n个记录按某个关键字排序，你能够采用基于比较的方法设计出平均时间复杂度好于为$O(n\log_2 n)$的排序算法吗？

4、内排序算法的稳定性

当待排序记录的关键字均不相同时，排序的结果是唯一的。



如果待排序的表中，存在有多个关键字相同的记录，经过排序后这些具有相同关键字的记录之间的相对次序保持不变，则称这种排序方法是稳定的。

3 4 3 1 5

1 2 3 3 5

反之，若具有相同关键字的记录之间的相对次序发生变化，则称这种排序方法是不稳定的。

3 4 3 1 5

1 2 3 3 5

5、正序和反序

若待排序的表中元素已按关键字排好序，称此表中元素为正序；

反之，若待排序的表中元素的关键字顺序正好和排好序的顺序相反，称此表中元素为反序。

有一些排序算法与初始序列的正序或反序有关，另一些排序算法与初始序列的情况无关。

6、 内排序数据的组织

待排序的顺序表的数据元素类型定义如下：

```
typedef int KeyType;          //定义关键字类型
typedef struct                //记录类型
{    KeyType key;             //关键字项
     InfoType data;           //其他数据项,类型为InfoType
} RecType;                    //排序的记录类型定义
```

——本讲完——