



# Data Structures

Ch7

# 查找 Searching

2023 年 12 月 5 日

学而不厌 诲人不倦

- ➡ 7.1 概述
- ➡ 7.2 线性表查找技术
- ➡ **7.3 树表的查找技术**
- ➡ 7.4 散列表查找技术
- ➡ 7.5 各种查找方法的比较
- ➡ 7.6 扩展与提高

本章的重点就是研究**查找表的存储方法**以及在此基础上的**查找方法**。

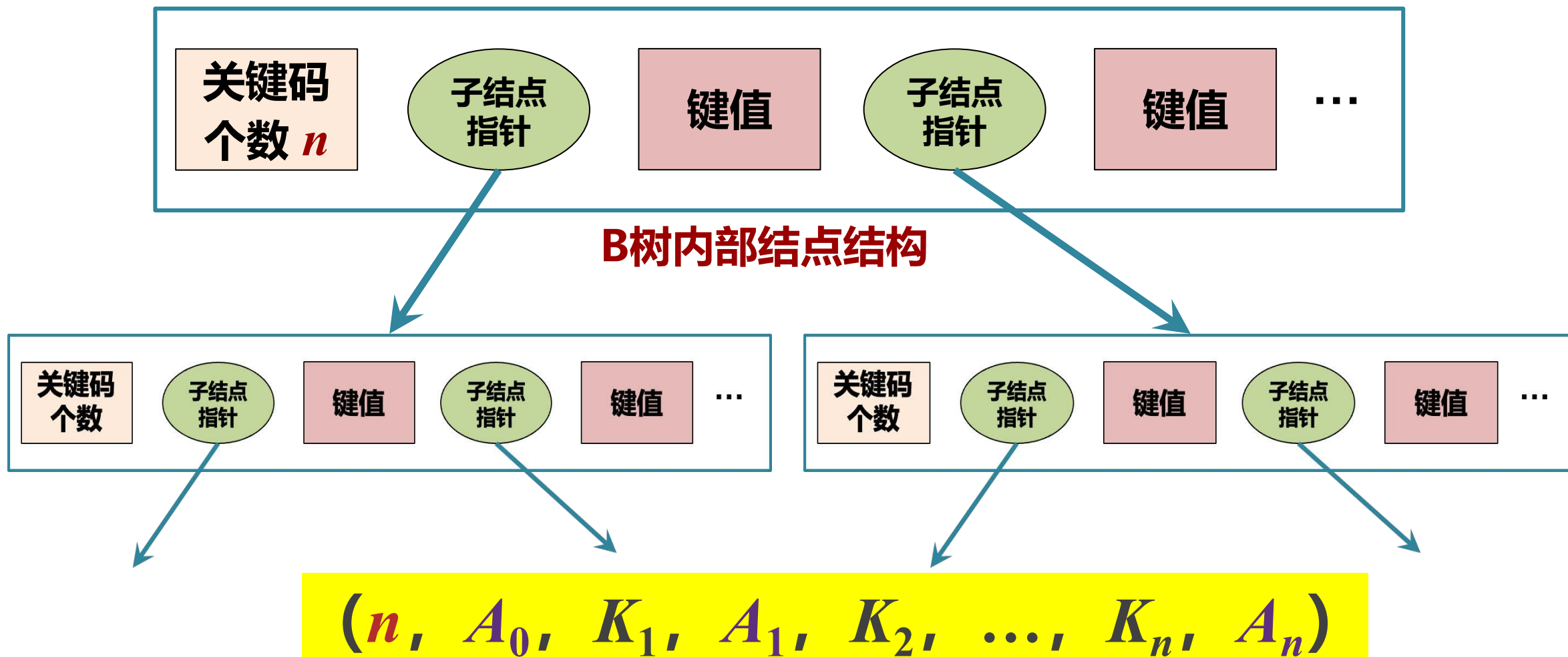
## 7.3 树表的查找技术

### 7-3-3 B树



### 1. B树的定义

Balanced Tree 平衡的多路查找树



## 7.3 树表的查找技术

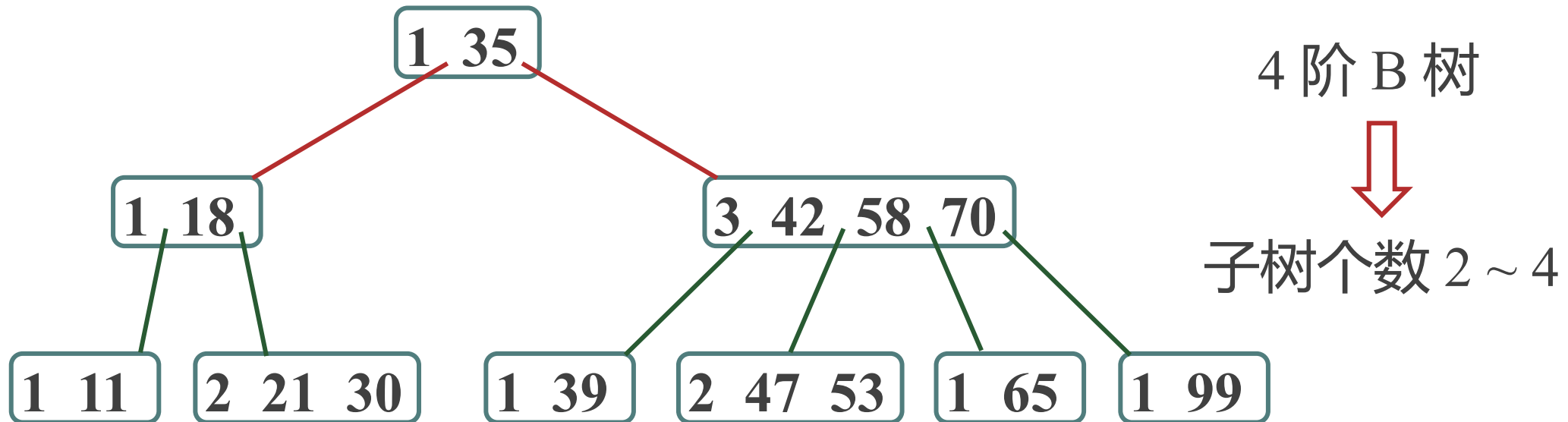
### 7-3-3 B树

#### 1. B树的定义

#### Balanced Tree 平衡的多路查找树

📌 **B树**：一棵 $m$ 阶的B树或者为空树，或者为满足下列特性的 $m$ 叉树：

- (1) 每个结点至多有  $m$  棵子树；
- (2) 根结点至少有两棵子树；
- (3) 除根结点和叶子结点外，所有结点至少有  $\lceil m/2 \rceil$  棵子树；



## 7.3 树表的查找技术

### 7-3-3 B树

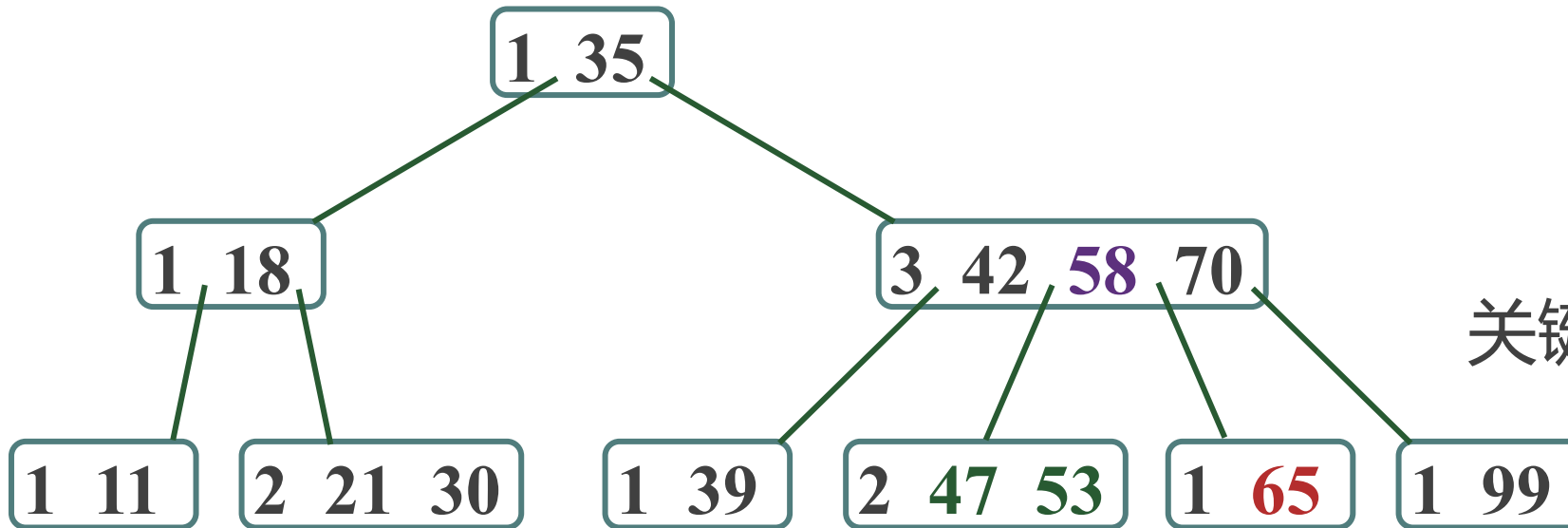
#### 1. B树的定义

Balanced Tree 平衡的多路查找树

✦ B树：一棵 $m$ 阶的B树或者为空树，或者为满足下列特性的 $m$ 叉树：  
(4) 所有结点都包含以下数据：

$(n, A_0, K_1, A_1, K_2, \dots, K_n, A_n)$

其中， $n$  ( $\lceil m/2 \rceil - 1 \leq n \leq m - 1$ ) 为**关键码的个数**， $K_i$  ( $1 \leq i \leq n$ ) 为**关键码**，且  $K_i < K_{i+1}$  ( $1 \leq i \leq n-1$ )， $A_i$  ( $0 \leq i \leq n$ ) 为**指向子树根结点的指针**，且指针 $A_i$ 所指子树中所有结点的关键码均小于 $K_{i+1}$ 大于 $K_i$ 。



4 阶 B 树



关键码个数 1 ~ 3

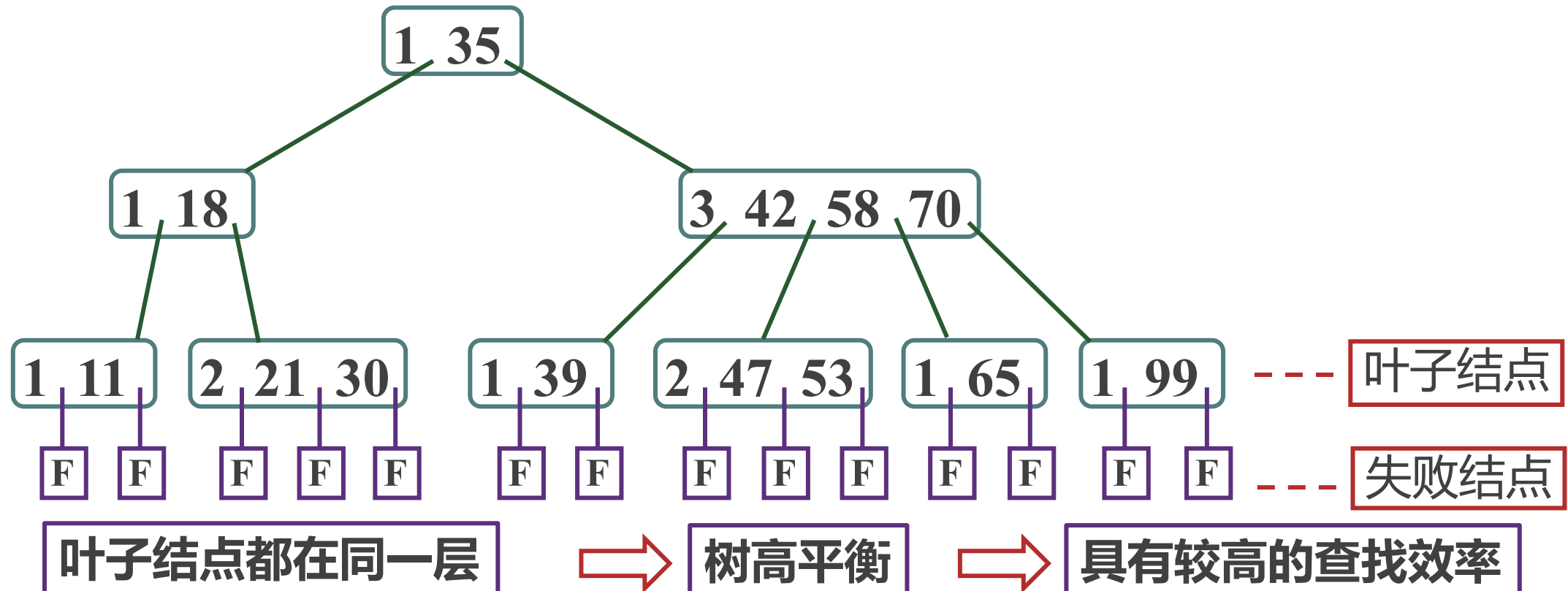
## 7.3 树表的查找技术

### 7-3-3 B树

#### 1. B树的定义

Balanced Tree 平衡的多路查找树

📌 B树：一棵 $m$ 阶的B树或者为空树，或者为满足下列特性的 $m$ 叉树：  
(5) 叶子结点都在同一层；



## 7.3 树表的查找技术

### 7-3-3 B树

#### 2. B树的插入

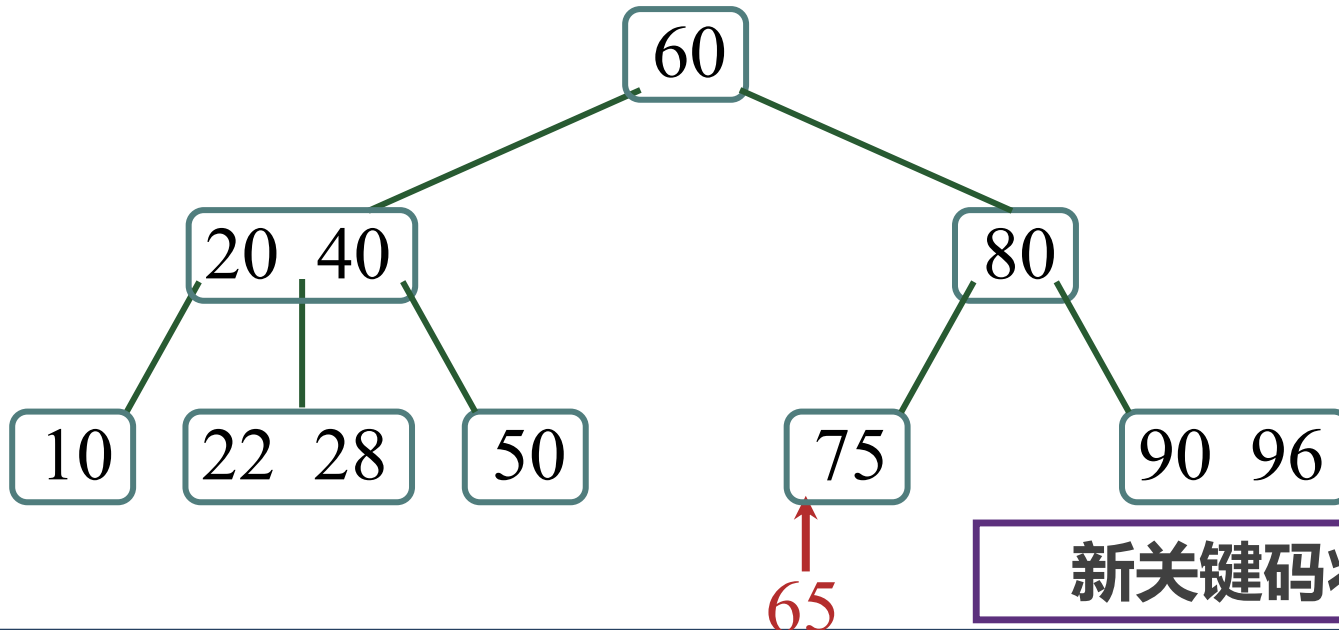
#### Balanced Tree 平衡的多路查找树

✈ 假定在  $m$  阶 B 树中插入关键码  $key$ ，设  $n=m-1$ ，插入过程如下：

(1) **定位**：确定关键码  $key$  应该插入哪个终端结点并返回该结点的指针  $p$ 。

若  $p$  中的关键码个数小于  $n$ ，则直接插入关键码  $key$ ；

否则，结点  $p$  的关键码个数**溢出**，执行“分裂——提升”过程。



🕒 B 树是多少阶？

$m = 3, n = 2$

新关键码将插入到相应的**叶子**结点中



## 7.3 树表的查找技术

### 7-3-3 B树

## 2. B树的插入

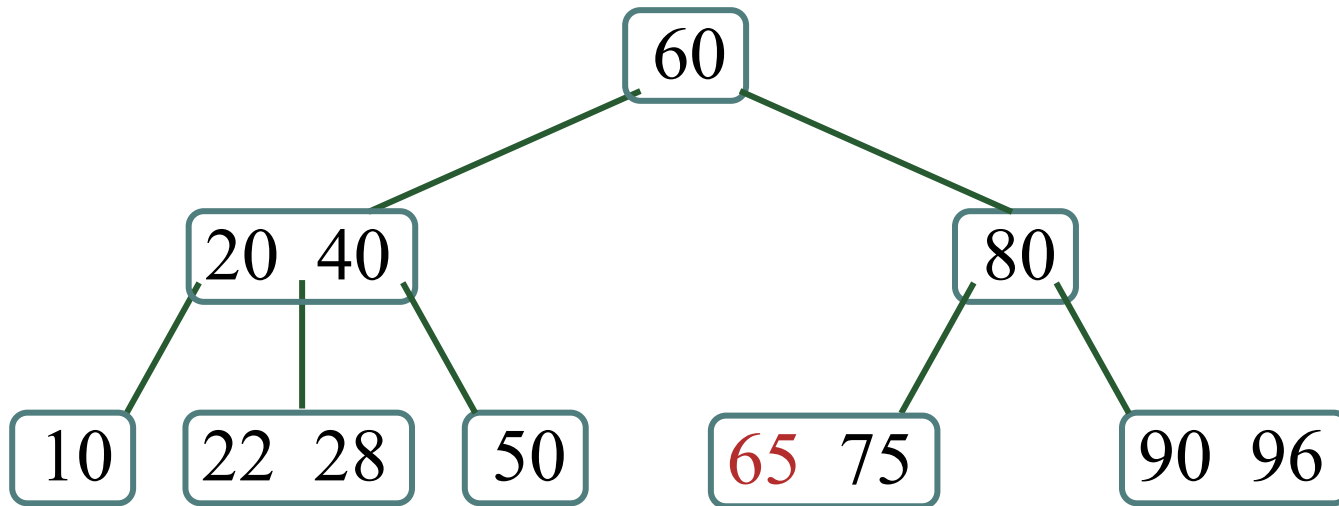
### Balanced Tree 平衡的多路查找树

✈ 假定在  $m$  阶 B 树中插入关键码  $key$ ，设  $n=m-1$ ，插入过程如下：

(1) **定位**：确定关键码  $key$  应该插入哪个终端结点并返回该结点的指针  $p$ 。

若  $p$  中的关键码个数小于  $n$ ，则直接插入关键码  $key$ ；

否则，结点  $p$  的关键码个数**溢出**，执行“分裂——提升”过程。



🕒 B 树是多少阶？

↓  
 $m = 3, n = 2$

## 7.3 树表的查找技术

### 7-3-3 B树

#### 2. B树的插入

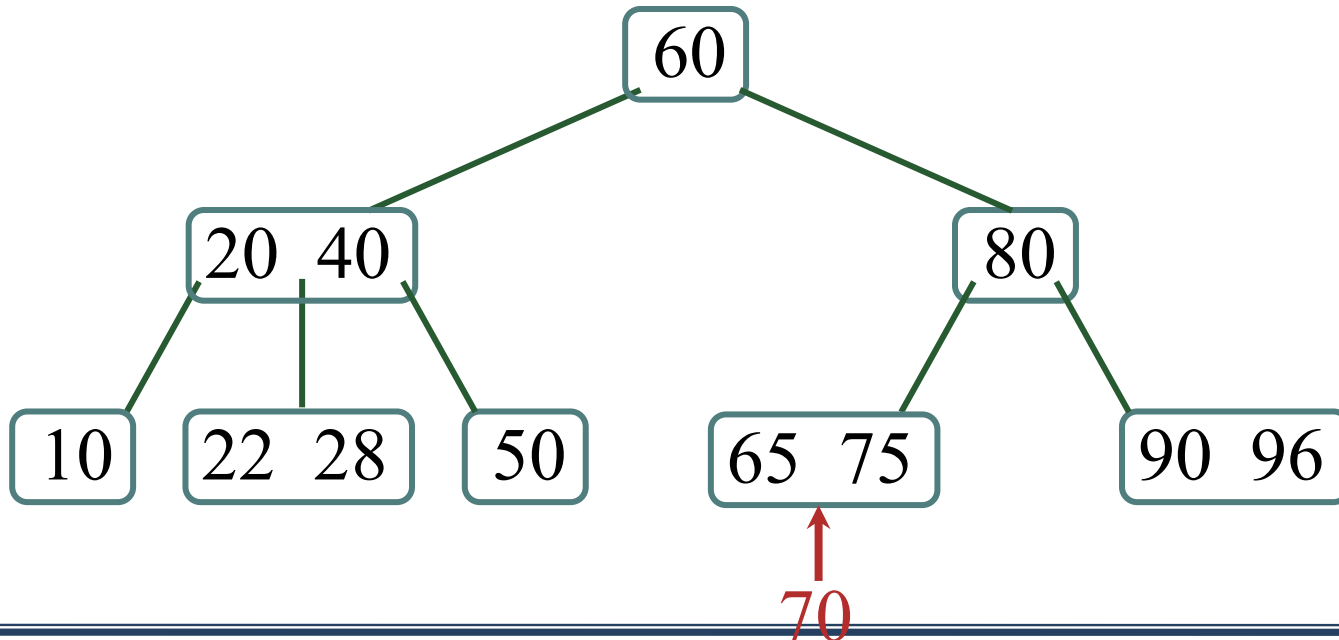
#### Balanced Tree 平衡的多路查找树

✈ 假定在  $m$  阶 B 树中插入关键码  $key$ ，设  $n=m-1$ ，插入过程如下：

(1) **定位**：确定关键码  $key$  应该插入哪个终端结点并返回该结点的指针  $p$ 。

若  $p$  中的关键码个数小于  $n$ ，则直接插入关键码  $key$ ；

否则，结点  $p$  的关键码个数**溢出**，执行“分裂——提升”过程。



🕒 B 树是多少阶？

$m = 3, n = 2$

发生溢出

## 7.3 树表的查找技术

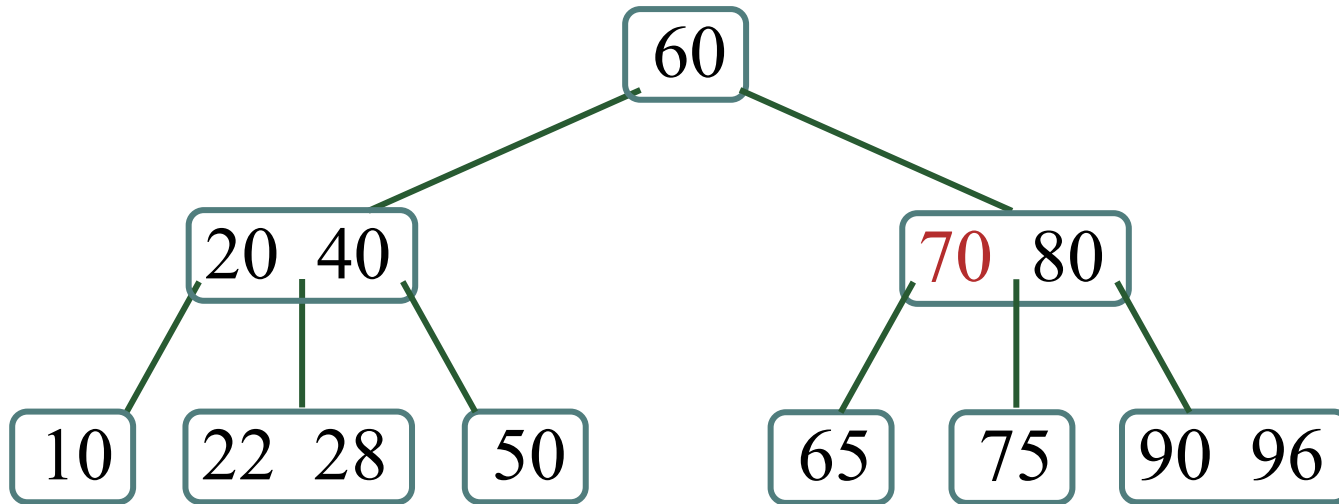
### 7-3-3 B树

#### 2. B树的插入

#### Balanced Tree 平衡的多路查找树

✈ 假定在  $m$  阶 B 树中插入关键码  $key$ ，设  $n=m-1$ ，插入过程如下：

(2) **分裂——提升**：将结点  $p$  “分裂” 成两个结点，分别是  $p_1$  和  $p_2$ ，把中间的关键码  $k$  “提升” 到父结点，并且  $k$  的左指针指向  $p_1$ ，右指针指向  $p_2$ 。



🕒 B 树是多少阶？



$$m = 3, n = 2$$

分裂——提升

## 7.3 树表的查找技术

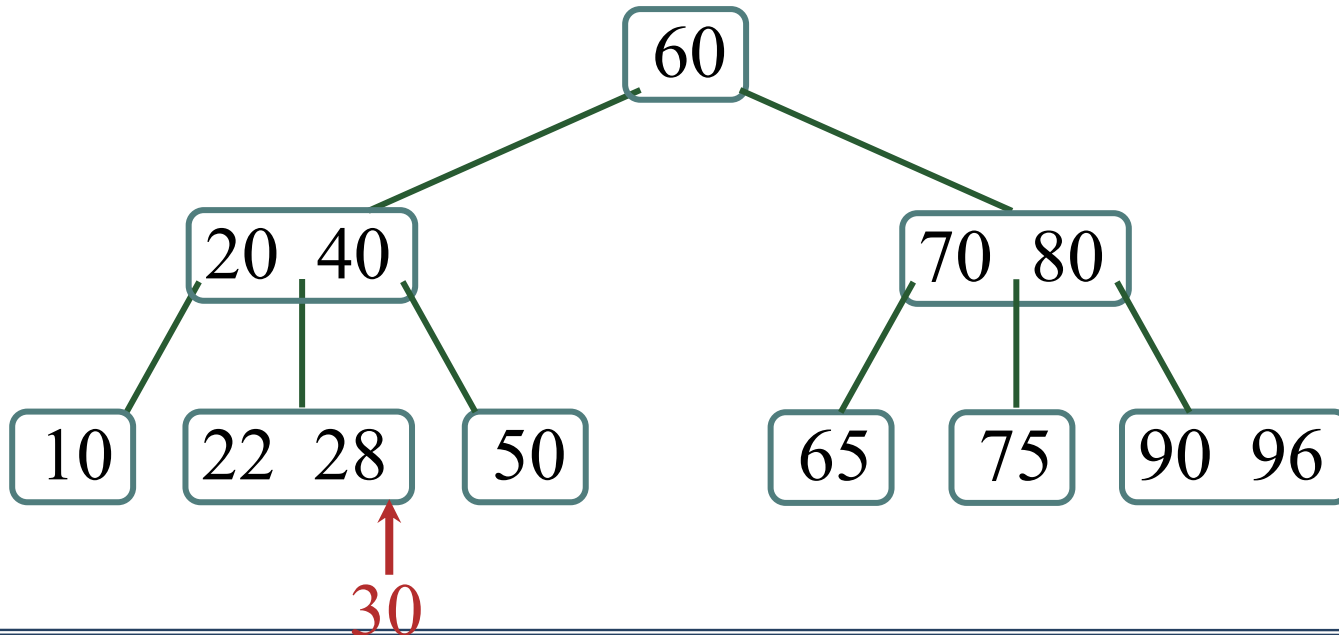
### 7-3-3 B树

#### 2. B树的插入

#### Balanced Tree 平衡的多路查找树

✈ 假定在  $m$  阶 B 树中插入关键码  $key$ ，设  $n=m-1$ ，插入过程如下：

(2) **分裂——提升**：如果父结点的关键码个数也溢出，则继续执行“分裂——提升”过程。显然，这种分裂可能一直上传，如果根结点也分裂了，则树的高度增加了一层。



🕒 B 树是多少阶？

$m = 3, n = 2$

发生溢出

## 7.3 树表的查找技术

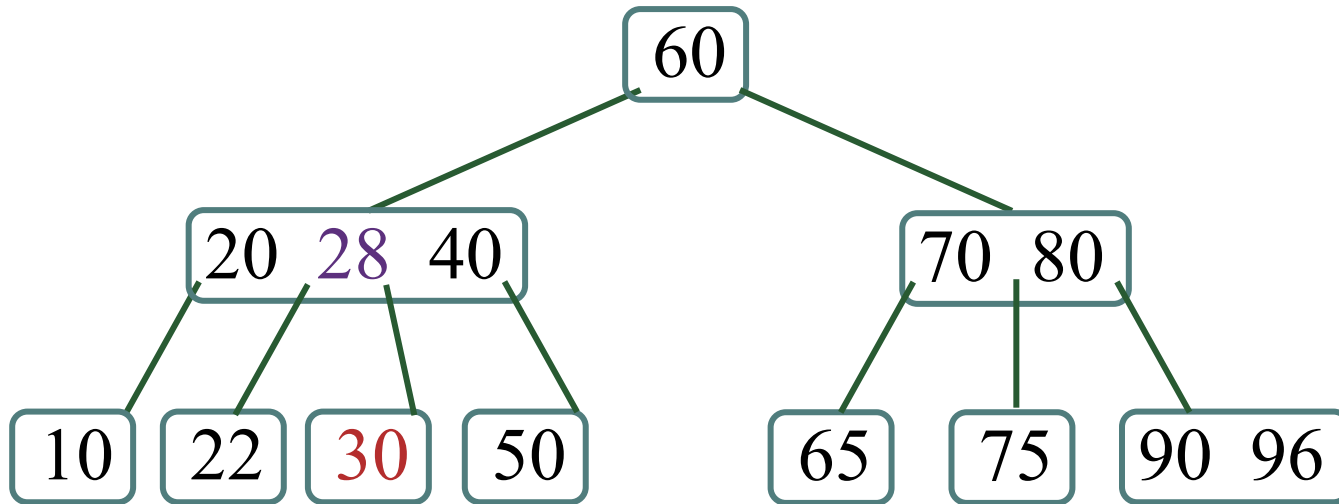
### 7-3-3 B树

#### 2. B树的插入

#### Balanced Tree 平衡的多路查找树

✈ 假定在  $m$  阶 B 树中插入关键码  $key$ ，设  $n=m-1$ ，插入过程如下：

(2) **分裂——提升**：如果父结点的关键码个数也溢出，则继续执行“分裂——提升”过程。显然，这种分裂可能一直上传，如果根结点也分裂了，则树的高度增加了一层。



🕒 B 树是多少阶？



$$m = 3, n = 2$$

分裂——提升  
再次溢出

## 7.3 树表的查找技术

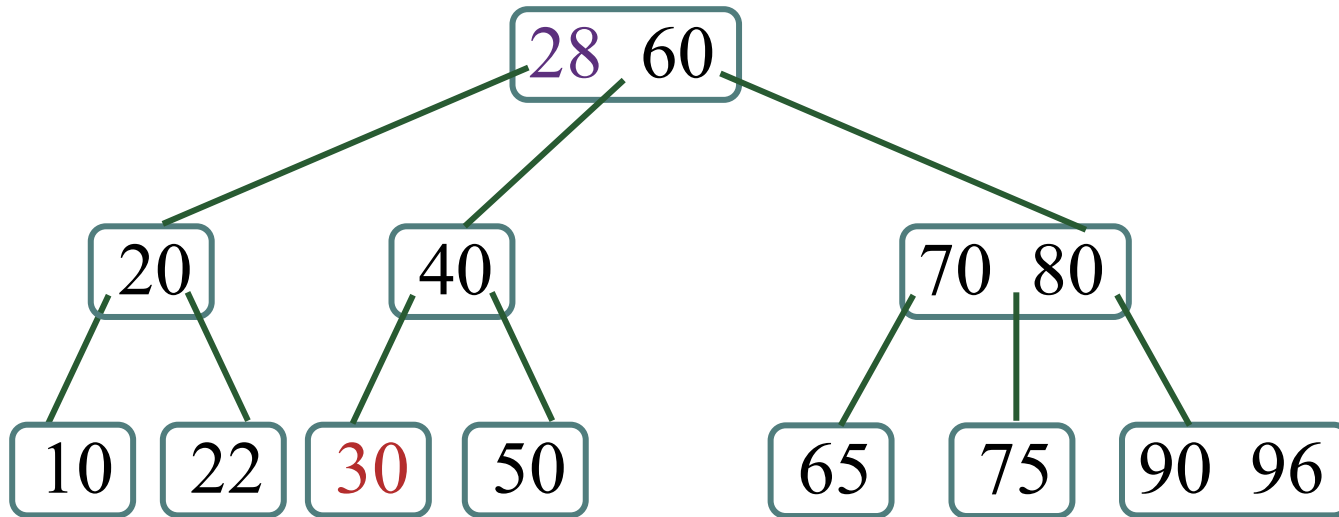
### 7-3-3 B树

## 2. B树的插入

### Balanced Tree 平衡的多路查找树

✈ 假定在  $m$  阶 B 树中插入关键码  $key$ ，设  $n=m-1$ ，插入过程如下：

(2) 分裂——提升：如果父结点的关键码个数也溢出，则继续执行“分裂——提升”过程。显然，这种分裂可能一直上传，如果根结点也分裂了，则树的高度增加了一层。



🕒 B 树是多少阶？



$$m = 3, n = 2$$

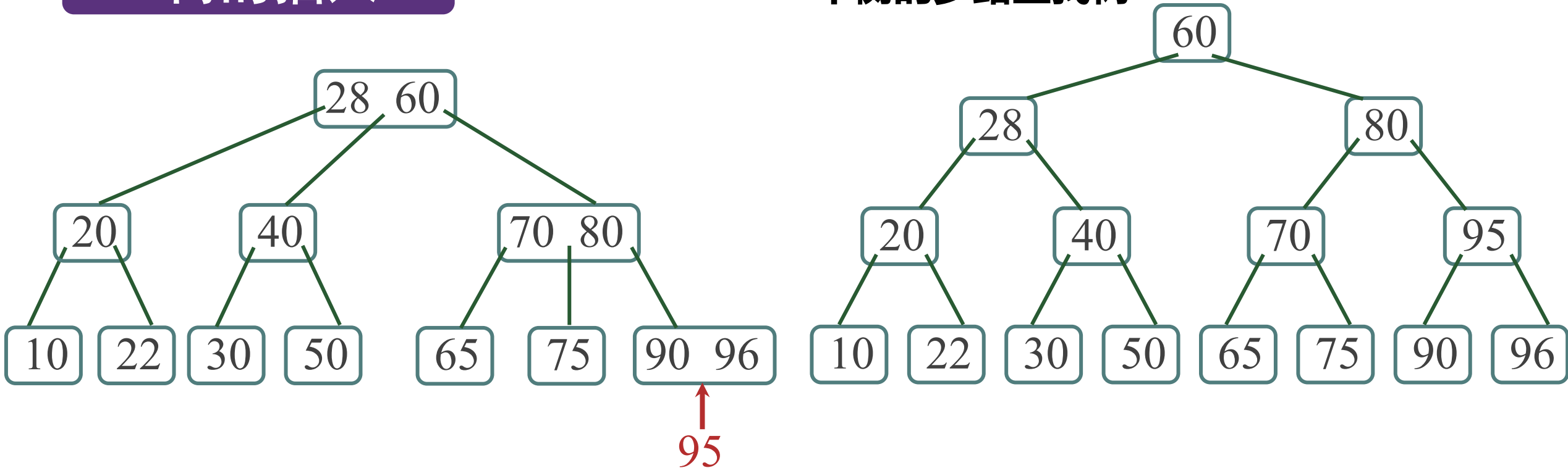
再次分裂——提升

## 7.3 树表的查找技术

### 7-3-3 B树

#### 2. B树的插入

Balanced Tree 平衡的多路查找树



 为什么 B 树的根结点最少有两棵子树?

如果在 B 树中插入一个元素时导致根结点发生溢出，则 B 树产生一个新的根结点并且树高增加了一层，此时，新根只有一个关键码和两棵子树。



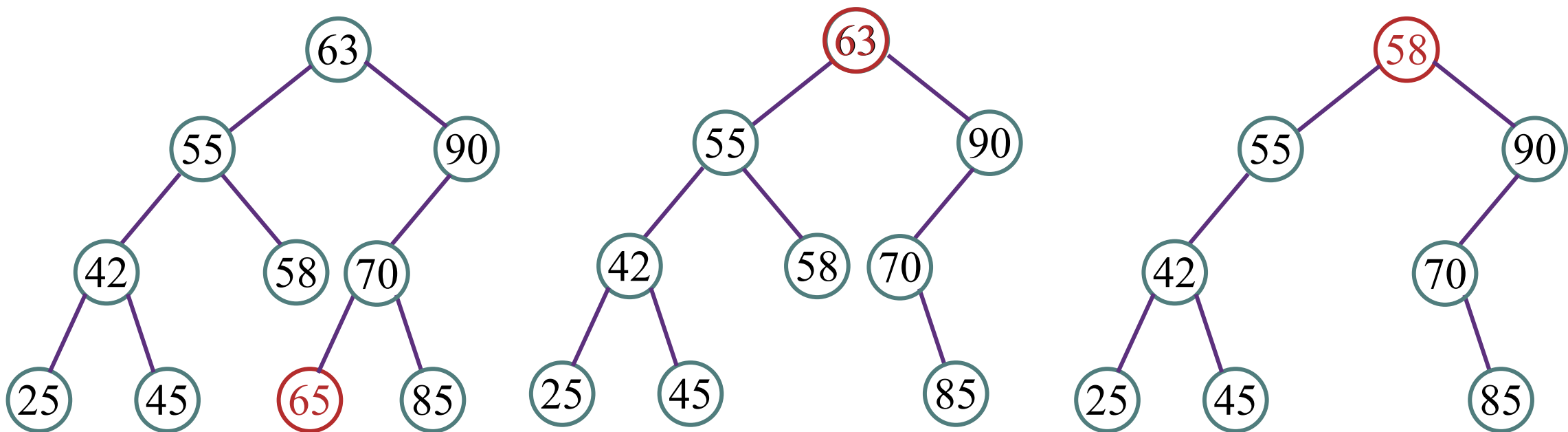
### 3. B树的删除

🕒 如何删除二叉排序树中的一个结点？

📎 情况 1——被删除的结点是叶子结点

📎 情况 3——被删除的结点既有左子树也有右子树

回顾





## 7.3 树表的查找技术

### 7-3-3 B树

#### 3. B树的删除

✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

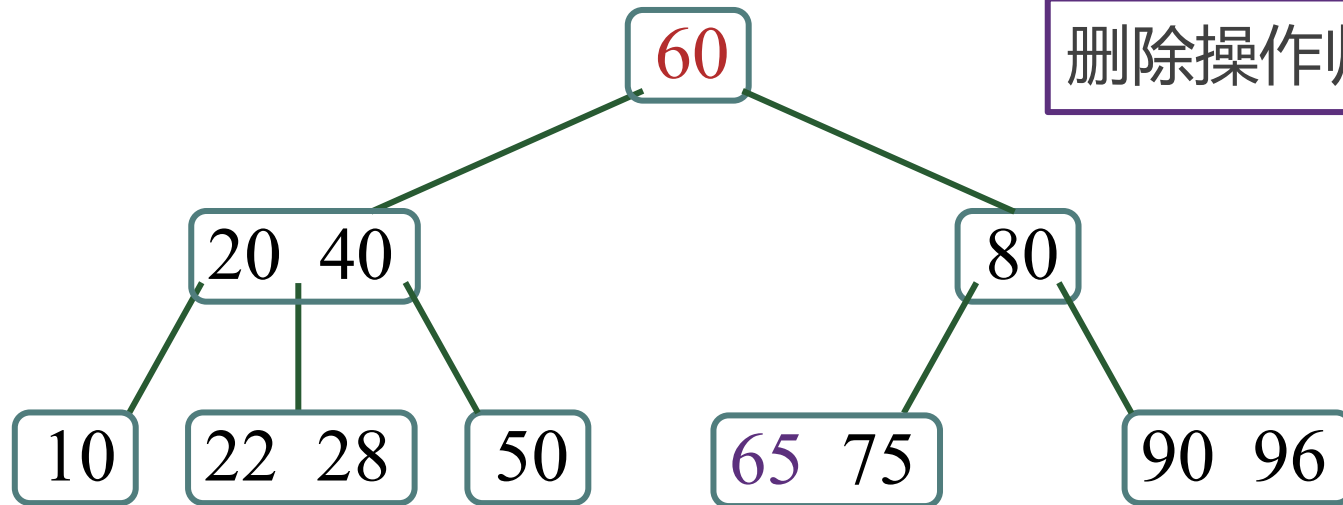
(1) **定位**：确定关键码  $key$  在哪个结点并返回该结点的指针  $q$ 。

假定  $key$  是结点  $q$  中的第  $i$  个关键码  $K_i$ ，有以下两种情况：

1.1 若结点  $q$  是叶子结点，则删除  $key$ ；

1.2 若结点  $q$  不是叶子结点，则用  $A_i$  所指子树中的最小值  $x$  替换  $K_i$ ；删除  $x$ ；

删除操作归结为在叶子结点中删除关键码



## 7.3 树表的查找技术

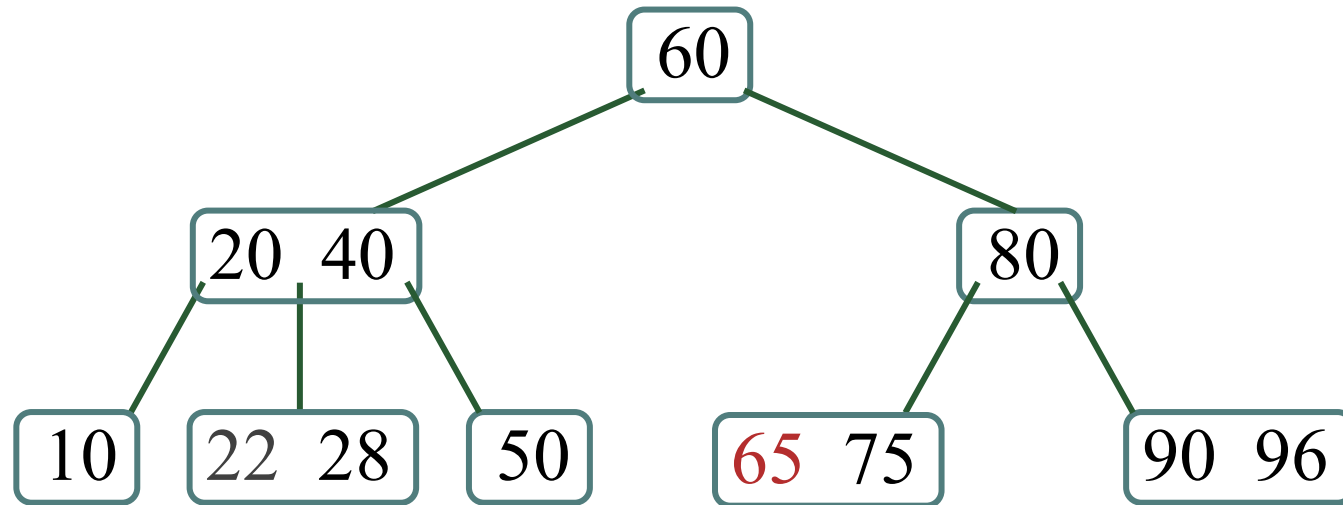
### 7-3-3 B树

#### 3. B树的删除

✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(2) 判断是否下溢

2.1 如果叶子结点中关键码的个数 **大于**  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；



🕒 B 树是多少阶？

↓

$$\left\lceil \frac{m}{2} \right\rceil - 1 = 1$$

## 7.3 树表的查找技术

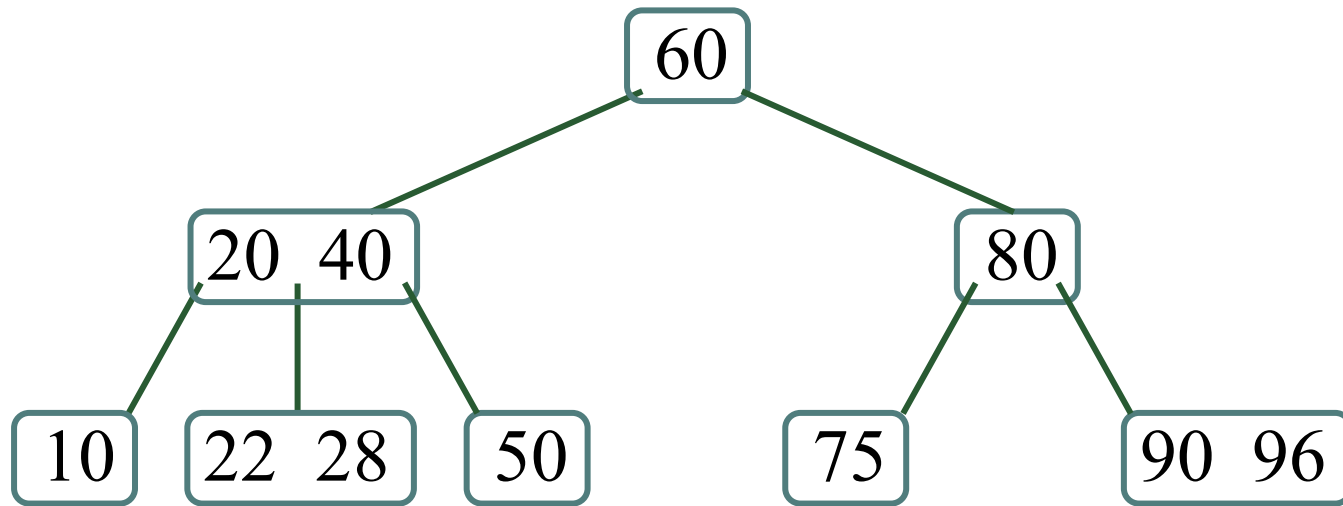
### 7-3-3 B树

#### 3. B树的删除

✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(2) 判断是否下溢

2.1 如果叶子结点中关键码的个数大于  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；



## 7.3 树表的查找技术

### 7-3-3 B树

#### 3. B树的删除

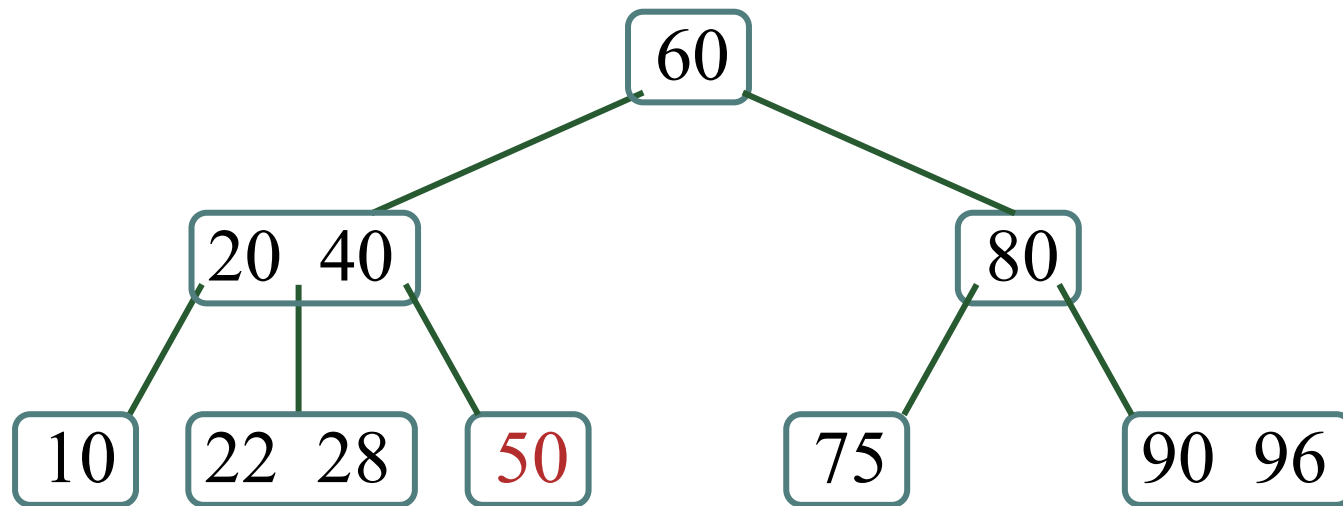
✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(2) 判断是否下溢

2.1 如果叶子结点中关键码的个数大于  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；

2.2 否则，删除操作涉及到兄弟结点

2.2.1 兄弟结点的关键码个数 **大于**  $\left\lceil \frac{m}{2} \right\rceil$ ，则向兄弟结点借一个关键码；



## 7.3 树表的查找技术

### 7-3-3 B树

#### 3. B树的删除

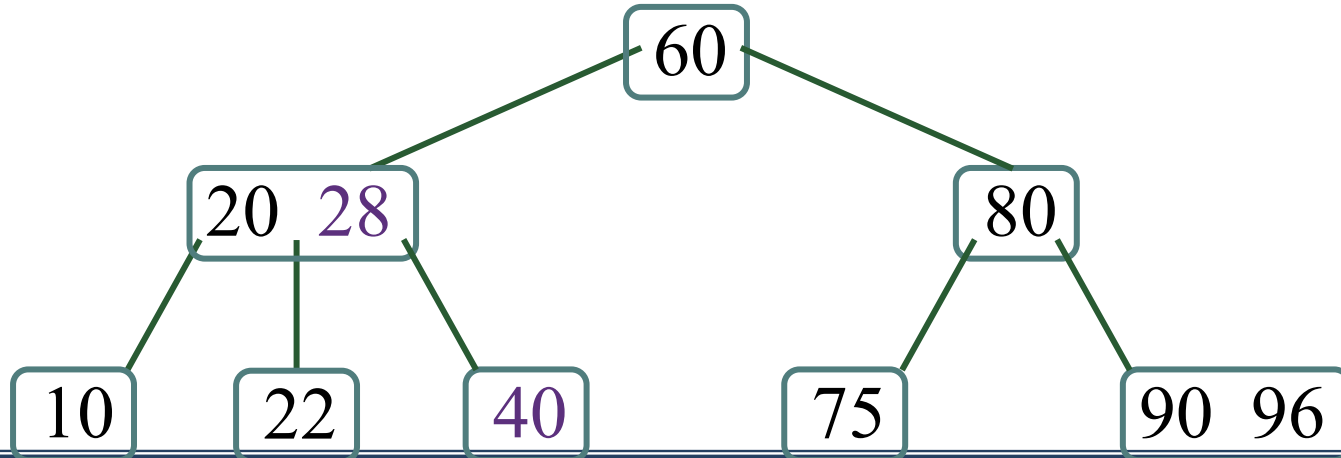
✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(2) 判断是否下溢

2.1 如果叶子结点中关键码的个数大于  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；

2.2 否则，删除操作涉及到兄弟结点

2.2.1 兄弟结点的关键码个数大于  $\left\lceil \frac{m}{2} \right\rceil$ ，则向兄弟结点借一个关键码，并且借来的关键码“上移”到双亲结点，双亲结点相应关键码“下移”；



## 7.3 树表的查找技术

### 7-3-3 B树

#### 3. B树的删除

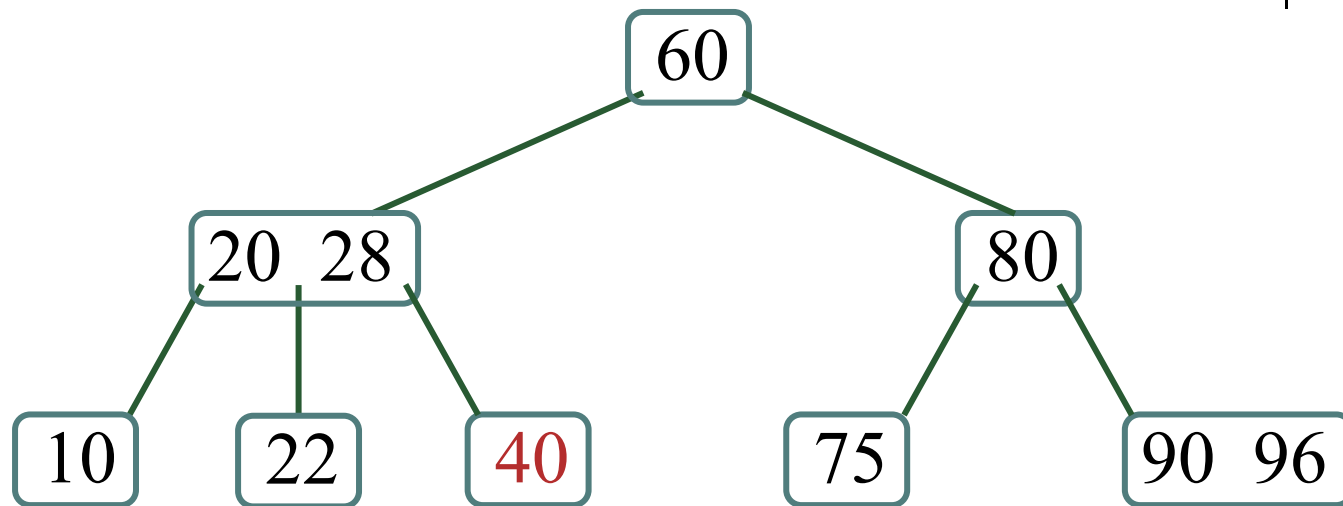
✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(2) 判断是否下溢

2.1 如果叶子结点中关键码的个数大于  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；

2.2 否则，删除操作涉及到兄弟结点

2.2.2 兄弟结点的关键码个数 **不大于**  $\left\lceil \frac{m}{2} \right\rceil$ ，则执行“合并”兄弟操作；



## 7.3 树表的查找技术

### 7-3-3 B树

#### 3. B树的删除

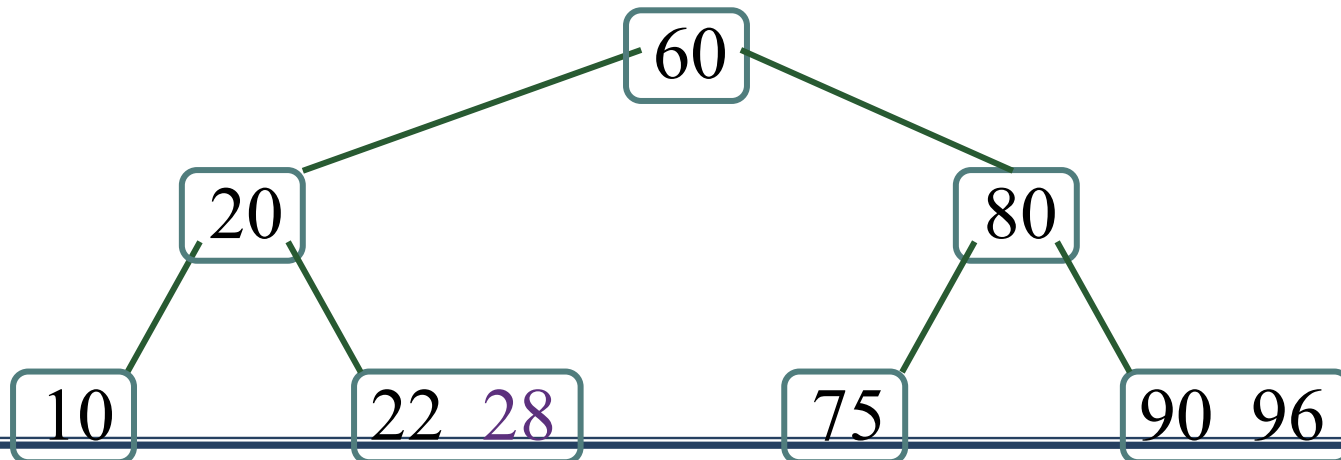
✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(2) 判断是否下溢

2.1 如果叶子结点中关键码的个数大于  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；

2.2 否则，删除操作涉及到兄弟结点

2.2.2 兄弟结点的关键码个数 **不大于**  $\left\lceil \frac{m}{2} \right\rceil$ ，则执行“合并”兄弟操作；  
删除空结点，并将双亲结点中的相应关键码“下移”到合并结点中；



## 7.3 树表的查找技术

### 7-3-3 B树

#### 3. B树的删除

✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(2) 判断是否下溢

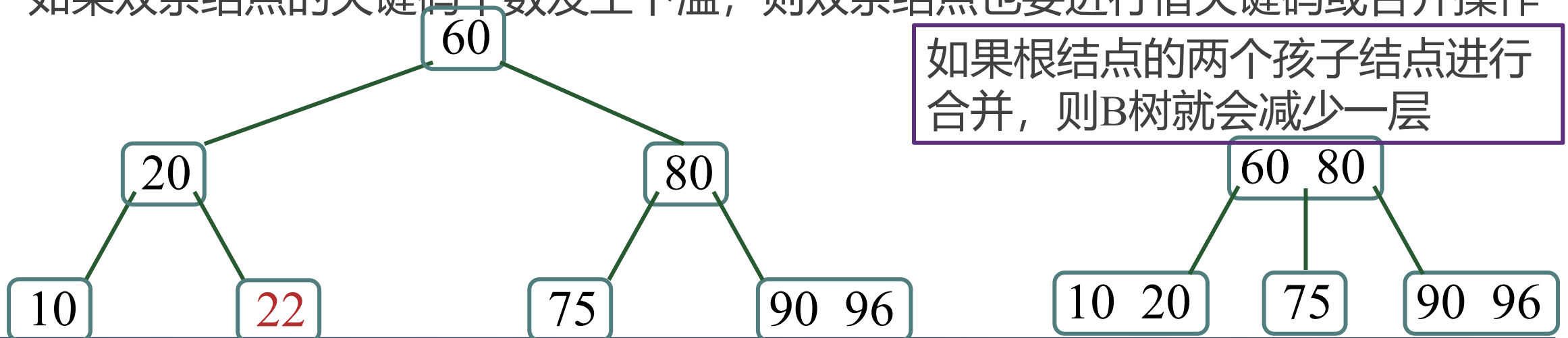
2.1 如果叶子结点中关键码的个数大于  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；

2.2 否则，删除操作涉及到兄弟结点

2.2.2 兄弟结点的关键码个数 **不大于**  $\left\lceil \frac{m}{2} \right\rceil$ ，则执行“合并”兄弟操作；

如果双亲结点的关键码个数发生下溢，则双亲结点也要进行借关键码或合并操作

如果根结点的两个孩子结点进行合并，则B树就会减少一层





## 7.3 树表的查找技术

### 7-3-3 B树

#### 3. B树的删除

✈ 假定在  $m$  阶 B 树中删除关键码  $key$ ，删除过程如下：

(1) 定位：确定关键码  $key$  在哪个结点并返回该结点的指针  $q$ 。

假定  $key$  是结点  $q$  中的第  $i$  个关键码  $K_i$ ，有以下两种情况：

1.1 若结点  $q$  是叶子结点，则删除  $key$ ；

1.2 否则用  $A_i$  所指子树中的最小值  $x$  替换  $K_i$ ；删除  $x$ ；

(2) 判断是否下溢

2.1 如果叶子结点中关键码的个数大于  $\left\lceil \frac{m}{2} \right\rceil - 1$ ，则直接删除；

2.2 否则，删除操作涉及到兄弟结点

2.2.1 兄弟结点的关键码个数大于  $\left\lceil \frac{m}{2} \right\rceil$ ，则向兄弟结点借一个关键码，并且借来的关键码“上移”到双亲结点，双亲结点相应关键码“下移”；

2.2.2 否则执行“合并”兄弟操作；删除空结点，并将双亲结点中的相

应关键码“下移”到合并结点中；

## 小结

1. 掌握二叉排序树的构建、查找、插入、删除原理
2. 掌握二叉排序树的实现方法和性能分析方法
3. 掌握平衡二叉树的相关概念和不同类型平衡调整方法
4. 了解平衡二叉树的性能分析方法
5. 掌握B树的定义和查找方法
6. 理解B树的插入和删除方法



*Thank You !*

*Q & A*