



## 第15章 搜索树(二)

---



计算机学院

# 主要内容

---

- B树
  - 定义
  - 搜索
  - 插入
  - 删除
  - B<sup>+</sup>树
- 红-黑树



# 动机：建立索引

---

- 当数据**元素数目较大**时，不能一次读入内存，可以把所有 $n$ 个元素分为 $b$ 个子块存放
- 建立索引表，其中每一项是索引项，记录了各子块中最大的关键值
- 此时，索引项与子块一一对应，**便于查找**！
- 当**元素数目特别大**时，索引也不能一次读入内存，可以建立索引的索引（**多级索引**）



# 索引示例



## 病房楼层索引图

北 区	楼 层	南 区
上 下 楼 安 全		⑨ 麻醉科手术室
		⑧ 儿 科 二 病 区
		⑦ 外 二 病 区 肛肠、泌尿、烧伤、耳鼻喉
		⑥ 妇 产 科 病 区
		⑤ 儿 科 病 区
		④ 神 经 内 科 病 区
		③ 内 科 病 区
		② 外 一 病 区
		① 骨 科 病 区
药房 新农合办公室 被褥发放处 治安室		

免费接诊电话：7100000

昵图网 [www.nipic.com](http://www.nipic.com) BY: ylxgyq NO:201100523104048009087



# 索引

Taq PCR Kit	97
Taq PCR Kit with Controls	97
T-Cell Protein Phosphatase	254
Technical Support	288
Terminal Transferase	114
Tilil	77
The Answers are Blowing in the Wind	286
The Hydrogen Economy	258
The River in the Ocean	196
Thermitase	240
Terminator DNA Polymerase	110
Terminator II DNA Polymerase	110
Terminator III DNA Polymerase	111
Terminator $\gamma$ DNA Polymerase	111
Thermophiles at 37°C, Activity of	302
Thermophilic Polymerases, Guidelines for PCR Optimization	354
Thermophilic Polymerases, Diluent Buffers	113
Thermophilic Polymerases, Reaction Buffers	112
Thermostable DNA Ligase (Taq)	126
Thermostable DNA Polymerases	96-113
Thermostable Inorganic Pyrophosphatase	153
Time-Saver Qualified Restriction Enzymes	304-305
Tilil	77
Tma Endonuclease III	139
Topoisomerase	
Cre Recombinase	156
Topoisomerase I	146
TR 1 Reverse Primer (19-mer)	267
Transfection Reagents	272-274
Transformation Efficiencies, Enhancing	346
TransPass Transfection Reagents	
COS/293	273
D1, D2	272
Fluorescein-siRNA Transfection Control	274
HeLa	273
HUVEC	273
P Protein	274
R1, R2	274
V	273
TriDye DNA Ladders	171
Troubleshooting Guide, Restriction Enzymes	296
Troubleshooting Guide, SNAP-tag Technology	360
Trypsin-digested BSA MS Standard (CAM-modified)	237
Trypsin, Modified (TPCK-treated)	238
Tsel	78
Tsp45I	78
Tsp509I	78
TspMI	78
TspRI	79
Tth111I	79
Tth Endonuclease IV	140
Tth RecA	154

## U

UDG	144
Universal miRNA Cloning Linker	189
Universal USER Cassette	158
Uracil-DNA Glycosylase (UDG)	144
Uracil Glycosylase Inhibitor (UGI)	145
USER Enzyme	158

## V

(Vanadate) Sodium Orthovanadate	254
Vectors	
Gaussia Luciferase	270
IMPACT	205
<i>K. lacis</i>	207
pMAL	202
pMAL pIII	242
SNAP/CLIP/ACP/MCP	267
Vent DNA Polymerase	98
Vent (exo-) DNA Polymerase	98
Vista Label	268

## W

Web site	290
----------	-----

## X

XbaI	79
XcmI	79
XhoI	80
XmaI	80
XmnI	80
XRN-1	194

## Y

Yeast Medium Pack	206
Yeast Chromosome PFG Marker	173
Yeast Expression	206-210

## Z

ZrAI	80
------	----





# 倒排索引示例

文档 1:

I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

文档 2:

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

词项	docID	词项	docID
I	1	ambitious	2
did	1	be	2
enact	1	brutus	1
julius	1	brutus	2
caesar	1	capitol	1
I	1	caesar	1
was	1	caesar	2
killed	1	caesar	2
i'	1	did	1
the	1	enact	1
capitol	1	hath	1
brutus	1	I	1
killed	1	I	1
me	1	i'	1
so	2	it	2
let	2	julius	1
it	2	killed	1
be	2	killed	1
with	2	let	2
caesar	2	me	1
the	2	noble	2
noble	2	so	2
brutus	2	the	1
hath	2	the	2
told	2	told	2
you	2	you	2
caesar	2	was	1
was	2	was	2
ambitious	2	with	2

词项	文档频率	倒排记录表
ambitious	1	→ [2]
be	1	→ [2]
brutus	2	→ [1] → [2]
capitol	1	→ [1]
caesar	2	→ [1] → [2]
did	1	→ [1]
enact	1	→ [1]
hath	1	→ [2]
I	1	→ [1]
i'	1	→ [1]
it	1	→ [2]
julius	1	→ [1]
killed	1	→ [1]
let	1	→ [2]
me	1	→ [1]
noble	1	→ [2]
so	1	→ [2]
the	2	→ [1] → [2]
told	1	→ [2]
you	1	→ [2]
was	2	→ [1] → [2]
with	1	→ [2]



# m叉搜索树

---

- **定义：** **m叉搜索树** ( $m$ -way search tree)  
可以是一棵空树，  
如果非空，它必须满足：
  - 1) 在相应的扩充搜索树中（用外部节点替换零指针），**每个内部节点最多可以有 $m$ 个子女及 $1 \sim m-1$ 个元素**（外部节点不含元素和子女）



# m叉搜索树

---

2) 每个含 $p$ 个元素的节点，有 $p+1$ 个子女

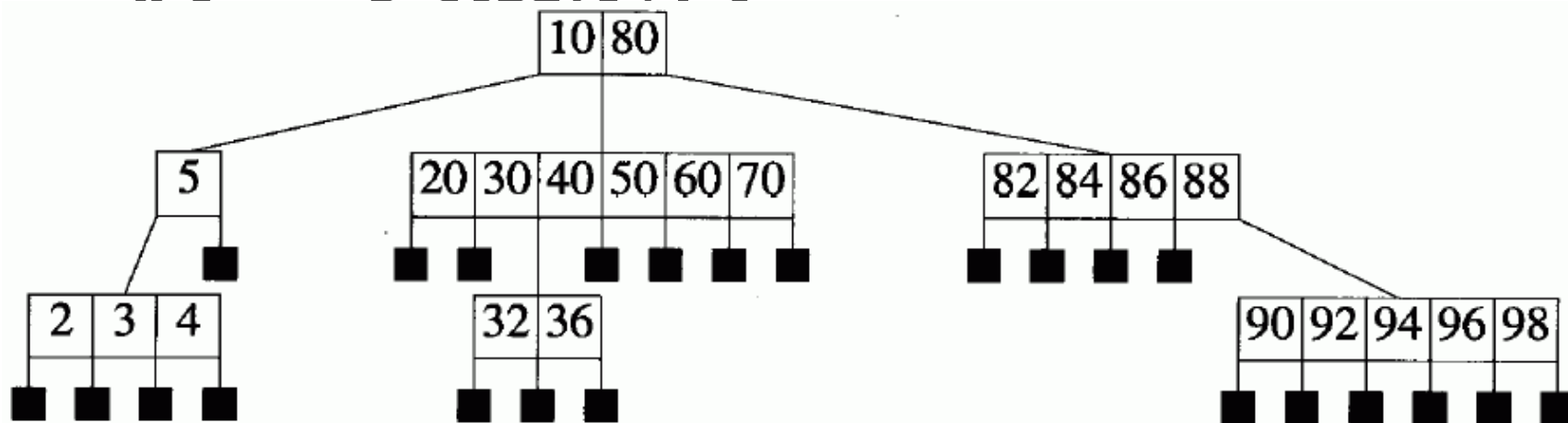
3) 考察含 $p$ 个元素的任意节点

- 设 $k_1, \dots, k_p$ 是这些元素的关键值，元素按关键字**升序**排列，即有 $k_1 < k_2 < \dots < k_p$   
设 $c_0, c_1, \dots, c_p$ 是节点的 $p+1$ 个孩子
  - 以 $c_0$ 为根的子树中的元素关键值小于 $k_1$
  - 以 $c_p$ 为根的子树中的元素关键值大于 $k_p$
  - 并且以 $c_i$ 为根的子树中的元素关键值会大于 $k_i$ 而小于 $k_{i+1}$ ，其中 $1 \leq i \leq p$





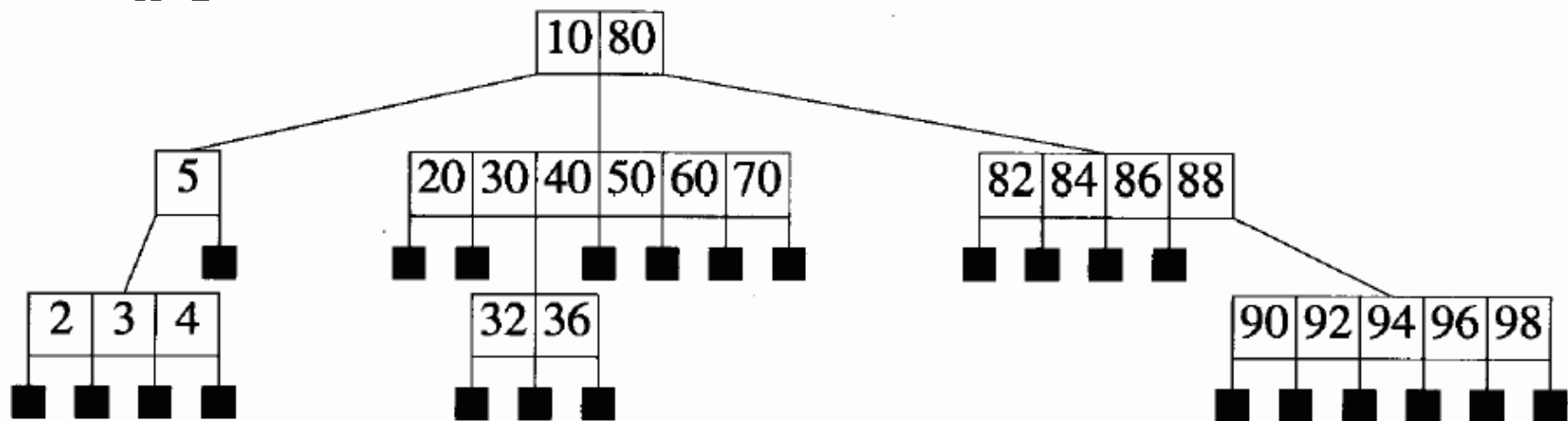
### 检验条件1、2、3



- 从根节点开始，向下搜索
- 对每个节点，若目标关键字 $k$ 
  - $=k_i$ ，搜索成功
  - $k_i < k < k_{i+1}$ ，在子树 $i$ 中继续寻找
- 到达外部节点——失败：搜索31



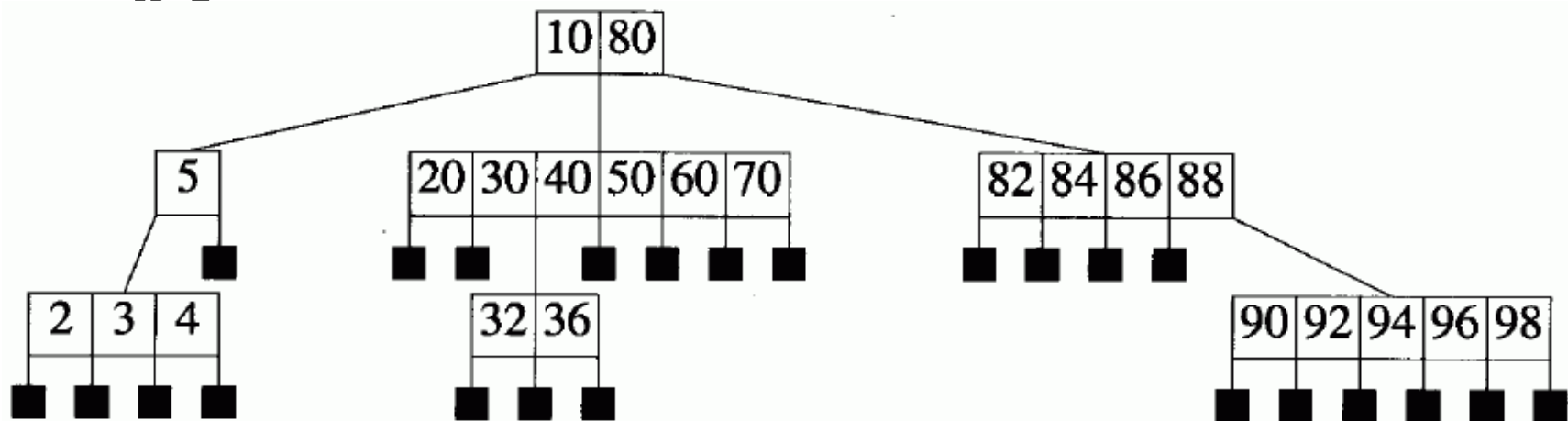
# 例



## • 插入操作

- 先进行搜索：成功，插入失败；否则，在失败位置（最后的一个内部节点）进行插入
- 元素数  $< m-1$ ，直接插入：插入31
- $\geq m-1$ ，生成新孩子节点：插入65

# 例



## • 删除操作

- 先进行搜索，找到元素后进行删除
- 左右子树均为空，直接进行删除：删除20
- 不都为空，子树中元素提升，替代被删除元素：删除10



# m叉搜索树的高度

- 高度为h，最少h个元素，最多 $m^h - 1$

- 1层~h-1层的每个节点孩子数都是m

- h层节点无孩子，节点总数

- 每个节点m-1个元素

- 元素总数 $m^h - 1$

$$\sum_{i=0}^{h-1} m^i = (m^h - 1) / (m - 1)$$

- n个元素，高度 $n \sim \log_m(n+1)$
- 与二叉搜索树类似，最坏情况很差——  
**m叉平衡搜索树，保证总有对数的复杂性**



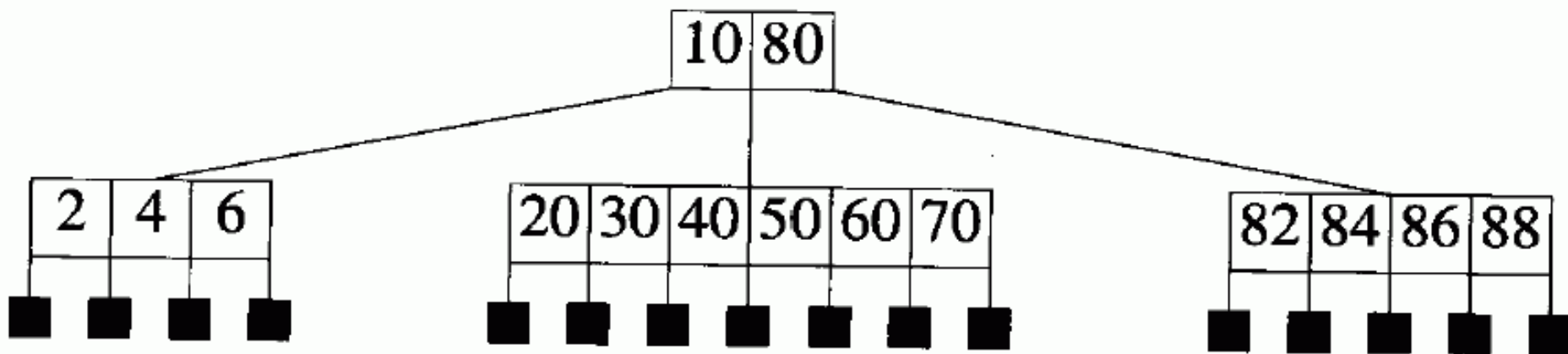
# H1. m阶B树

---

- **定义：**  $m$ 阶B树 (B-Tree of order  $m$ ) 是一棵  $m$ 叉搜索树，如果B树非空，那么相应的扩充树满足下列特征：
  - 1) 根节点至少有2个孩子
  - 2) 除根节点外，所有内部节点至少有  $\lceil m/2 \rceil$  个孩子
  - 3) 所有外部节点位于同一层上



# 7阶B树例



- 1) 在扩充搜索树中，每个内部节点最多可以有 $m$ 个子女及 $1 \sim m-1$ 个元素
- 2) 每个含 $p$ 个元素的节点，有 $p+1$ 个子女
- 3) 考察含 $p$ 个元素的任意节点以 $c_i$ 为根的子树中的元素关键值会大于 $k_i$ 而小于 $k_{i+1}$ ，其中 $1 \leq i \leq p$

- 1) 根节点至少有2个孩子
- 2) 除根节点外，所有内部节点至少有 $\lceil m/2 \rceil$ 个孩子
- 3) 所有外部节点位于同一层上

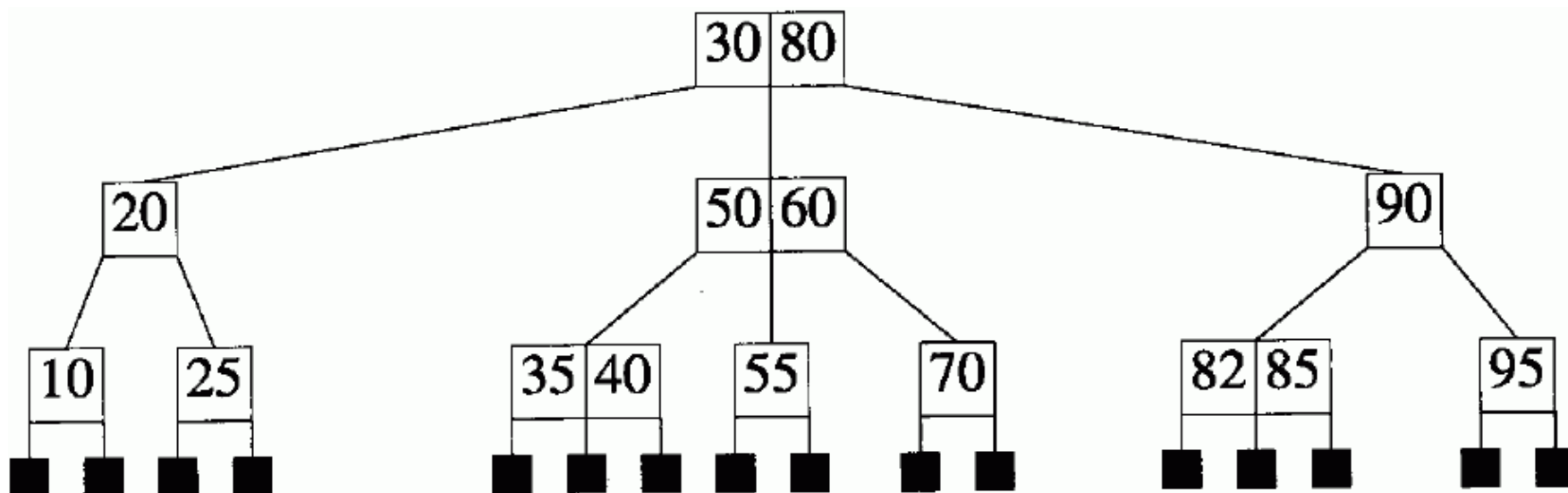




# 例

- 二阶B树：满二叉树
- 三阶B树：2-3树
- 四阶B树：2-3-4树

问题1：2-3-4树中有几种节点？  
问题2：这些节点如何存储？



# 重新整理m阶B树特征

1. 每个节点至多有m棵子树
2. 若根节点有子树，则至少有2棵子树
3. 除根节点外，每个节点至少有 $\text{ceil}(m/2)$ 棵子树
4. 子树与它的索引有确定的大小关系
5. 所有叶节点在同一层上

个数要求

大小要求

层次要求



# B树的高度

---

- 定理11-3

设  $T$  是一棵高度为  $h$  的  $m$  阶 B 树

$d = \lceil m/2 \rceil$  且  $n$  是  $T$  中的元素个数，则

1)  $2d^{h-1} - 1 \leq n \leq m^h - 1$

2)  $\log_m(n+1) \leq h \leq \log_d((n+1)/2) + 1$



# 证明

$n$ 的上限，由  $T$  是一棵  $m$  叉搜索树，得证

对于下限，扩充B树的外部节点都在  $h+1$  层。1

， 2， 3， 4，  $\dots$ ，  $h+1$  层的节点最小数目是1

， 2，  $2d$ ，  $2d^2$ ，  $\dots$ ，  $2d^{h-1}$ ，

→ B树中外部节点的最小数是  $2d^{h-1}$

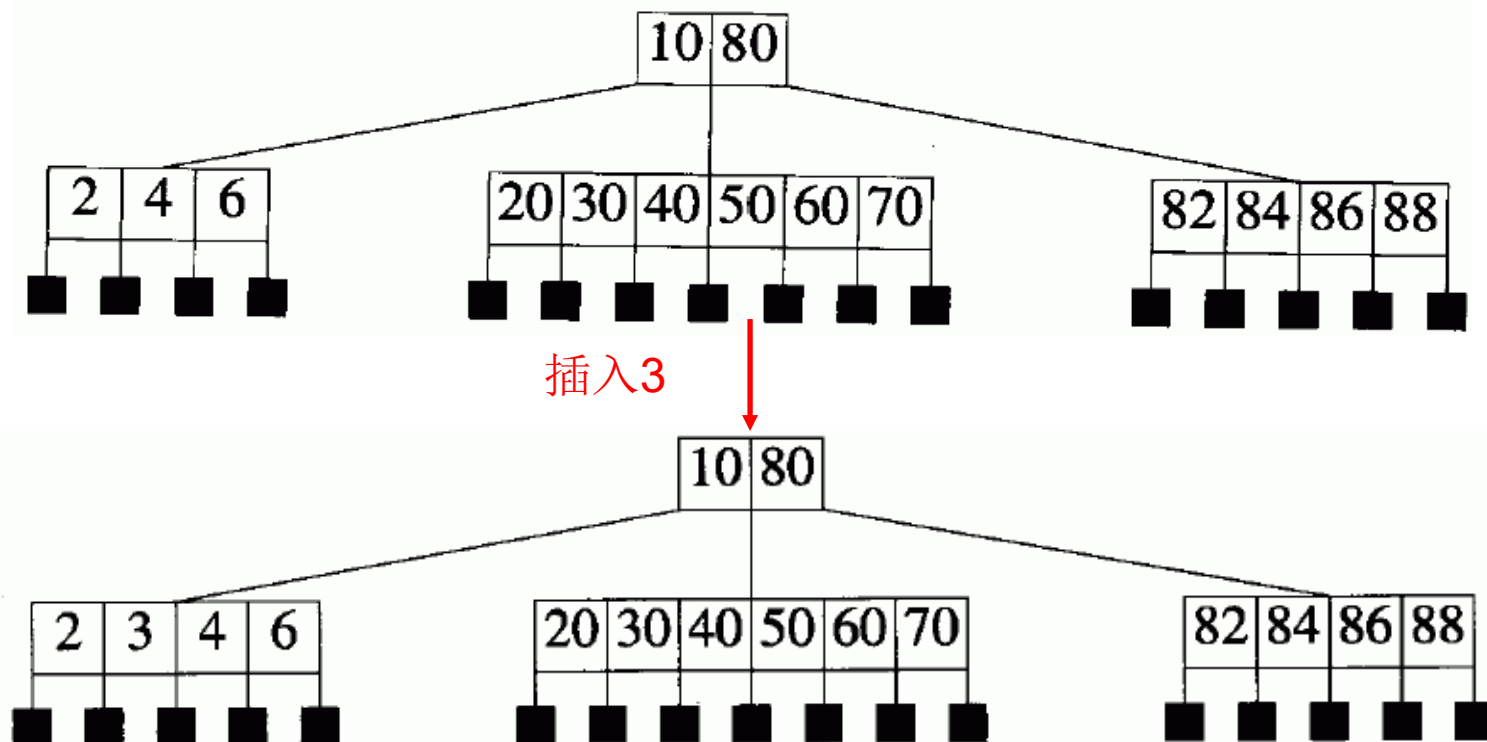
**外部节点的数量比元素的个数多1**

→  $n \geq 2d^{h-1} - 1$  从1) 直接可以得到2 )



# 插入操作——简单情况

- 插入节点的元素数  $< m-1$ ，直接插入



# 插入操作——分裂

- 插入节点的元素数 $=m-1$ ，再添加新的元素，必然超出限制，怎么办？分裂！
- 带空指针的新元素 $e$ 插入饱和节点 $P$   
→成为溢出节点

$m, c_0, (e_1, c_1), \dots, (e_m, c_m)$





# 插入操作——分裂

---

- 从  $e_d$  处分裂出新节点  $Q$ :

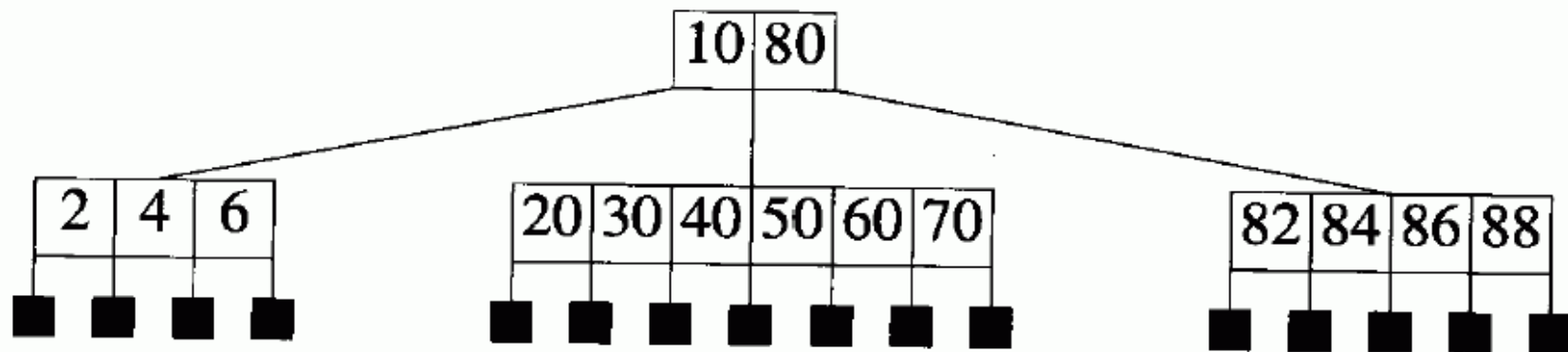
$P$ :  $d-1, c_0, (e_1, c_1), \dots, (e_{d-1}, c_{d-1})$

$Q$ :  $m-d, c_d, (e_{d+1}, c_{d+1}), \dots, (e_m, c_m)$

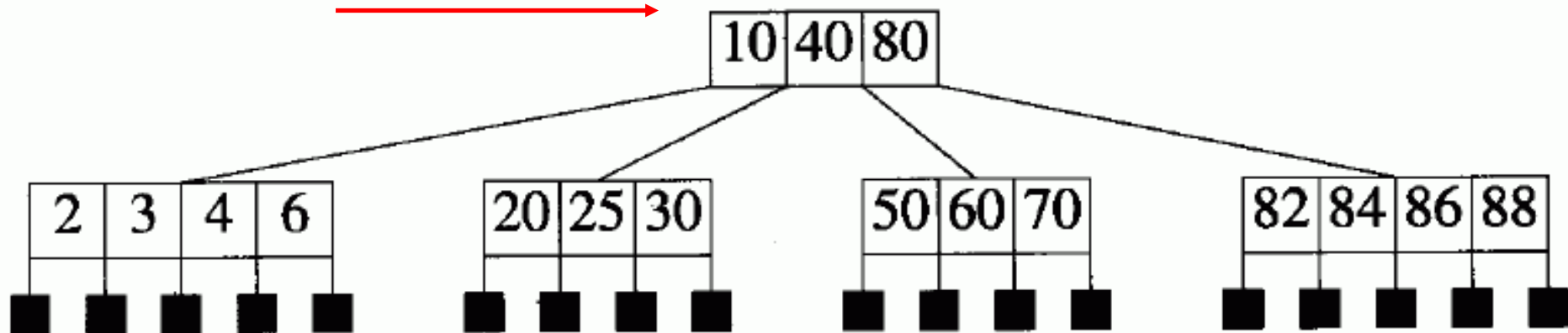
- $(e_d, Q)$  提升到  $P$  的父节点中



# 分裂例



插入25



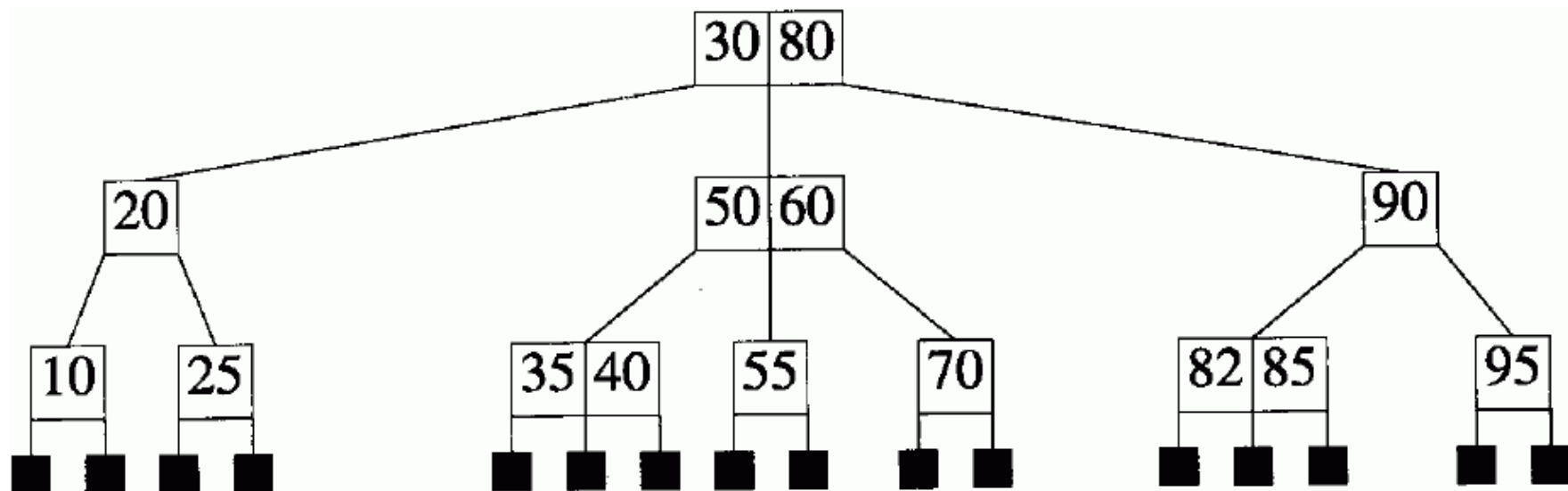
# 插入操作——分裂回溯

---

- 分裂后，中心节点提升至父节点——  
可能导致父节点溢出——  
继续分裂，...，直至根节点



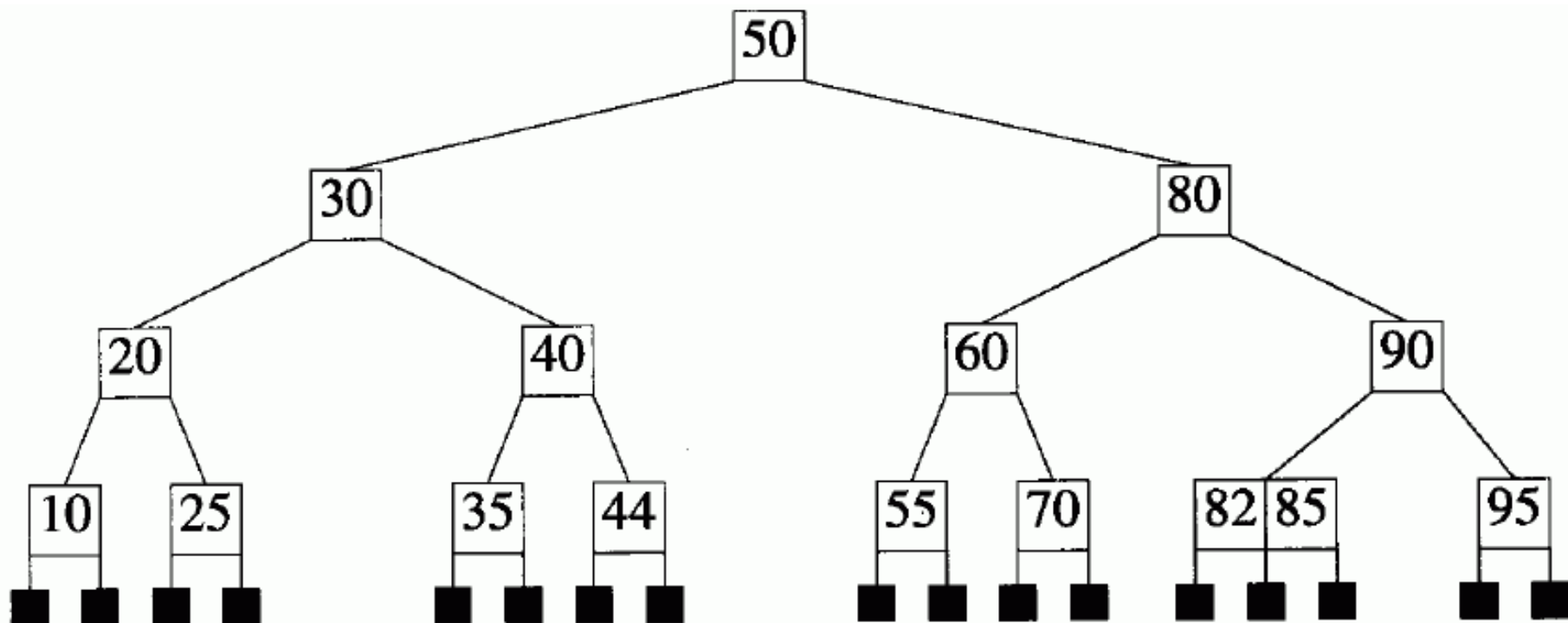
# 分裂回溯例——三阶B树



插入44



# 分裂回溯例（续）



# 删除操作

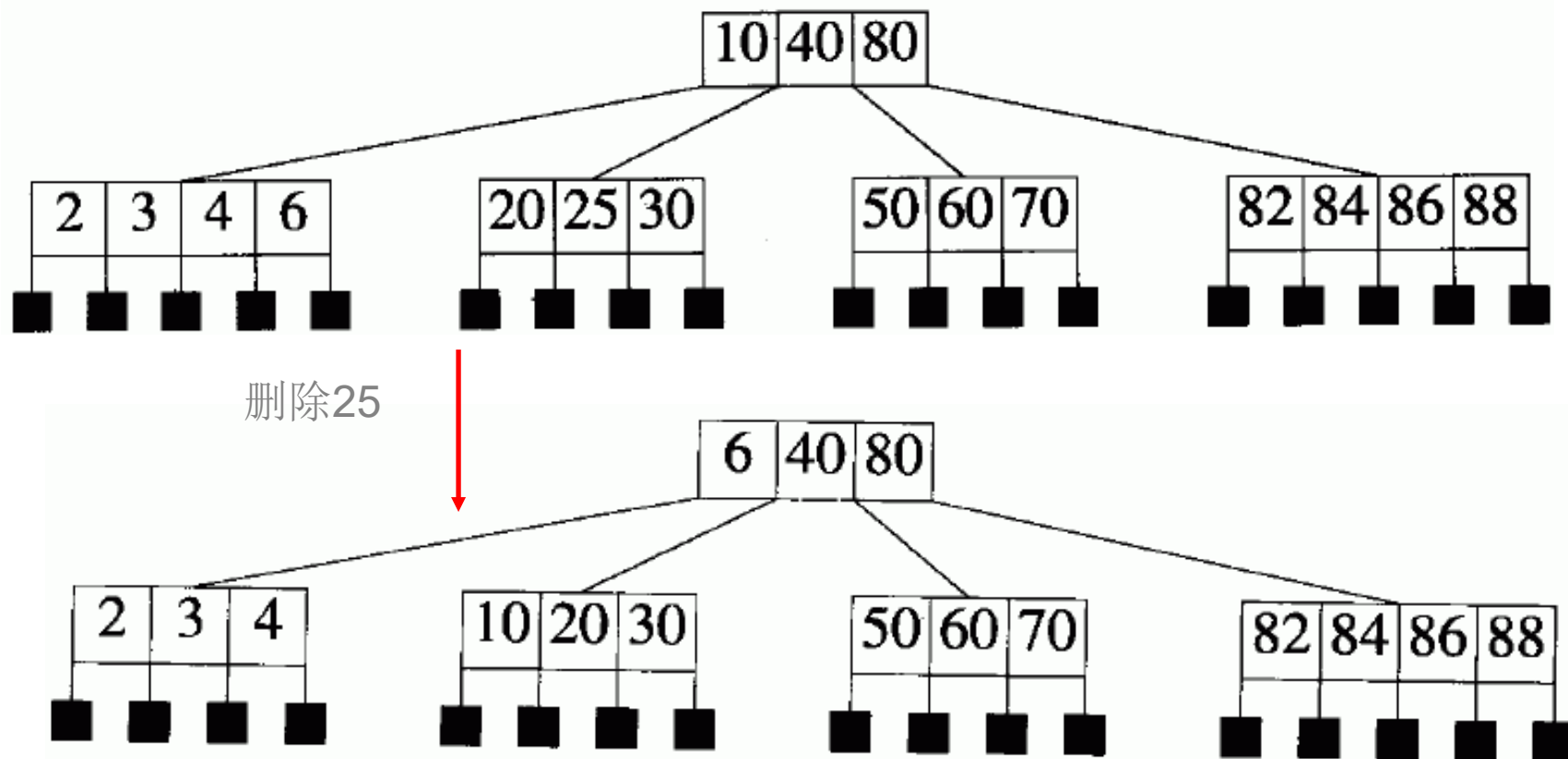
---

- 1) 删除叶节点中元素
- 2) 删除非叶节点元素，类似AVL树，子树叶节点元素与之交换，转换为1)
  - 元素数目 $>d-1$ ，直接删除即可
  - 元素数目 $=d-1$ ，删除后小于限制值
    - 借用兄弟节点“多余”的元素
    - 兄弟节点也无多余元素，合并





# 删除操作——借用兄弟节点 $m=7$



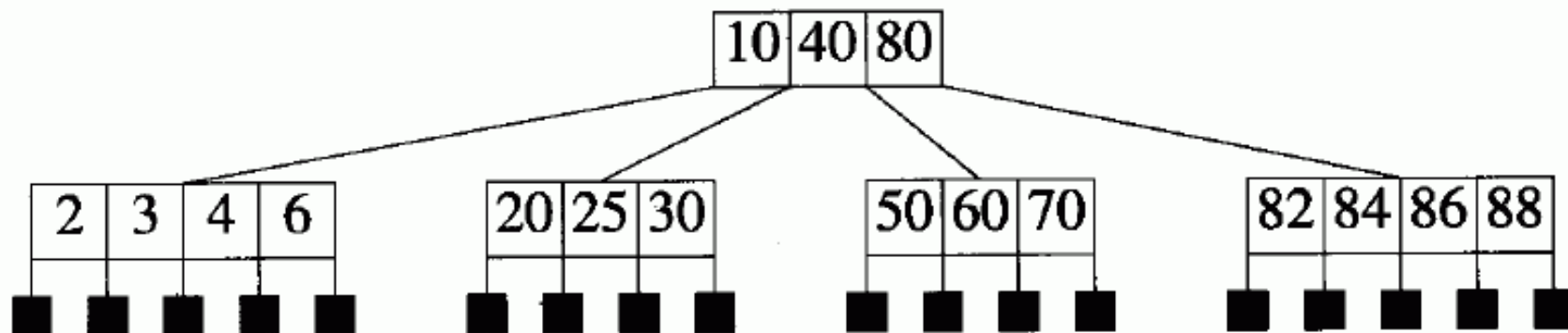
# 删除操作——借用兄弟节点

---

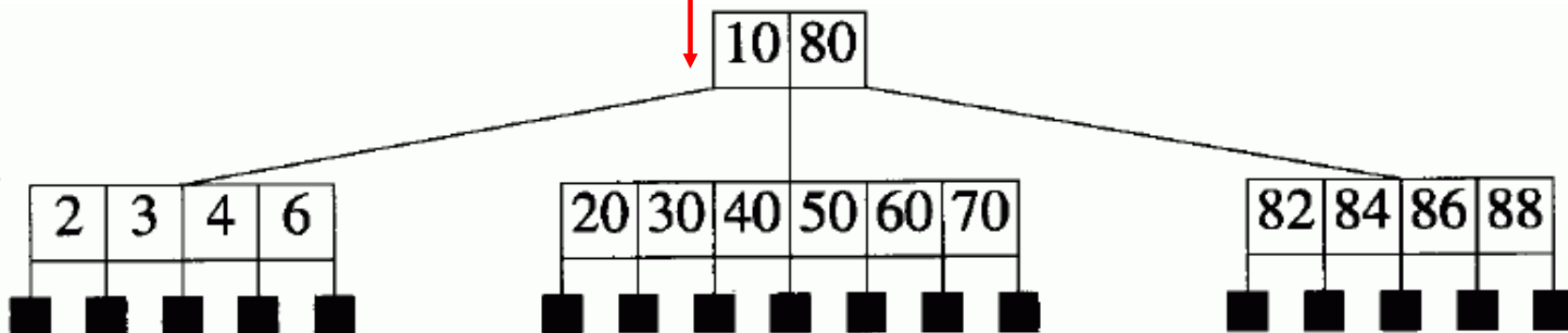
- 左（右）兄弟节点元素数 $>d-1$
- 将其最右（左）元素提升至父节点，  
父节点相应元素下降到删除节点  
→元素数目 $\geq d-1$



# 删除操作——与兄弟节点合并 $m=7$



删除25



# 删除操作——与兄弟节点合并

- 本节点元素数：  $d-2+$   
兄弟节点元素数：  $d-1+$   
父节点中介于两者之间的元素： 1  
 $= 2d - 2 \leq m - 1$



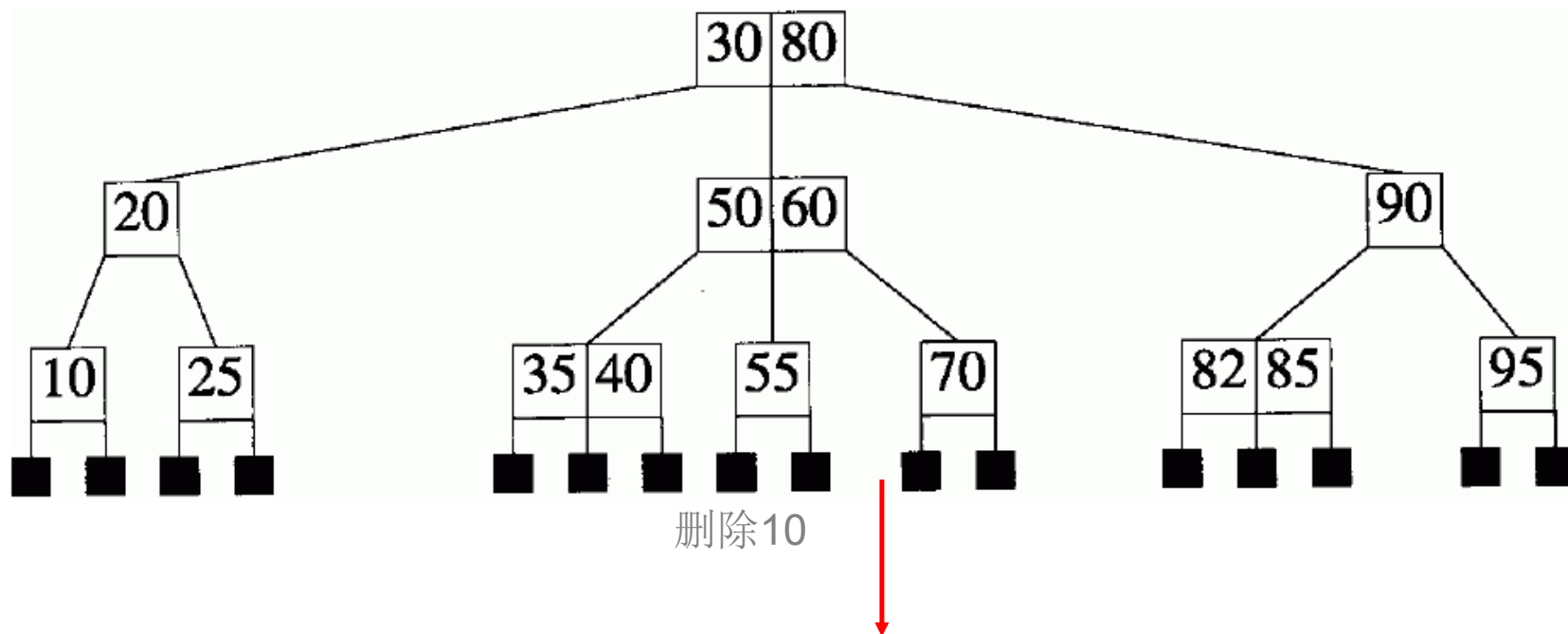
# 删除操作——合并的回溯

---

- 合并：父节点元素数减少，可能 $< d-1$
- 继续前面所述的处理方法
- 可能需合并——涉及祖父节点...
- 最坏情况回溯到根节点

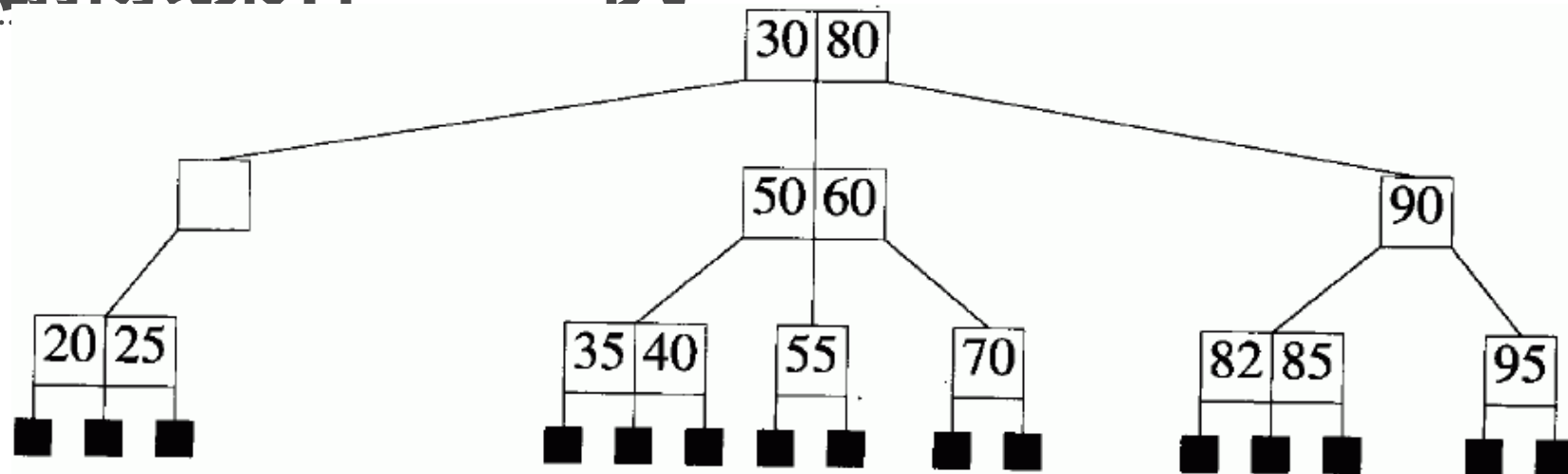


# 删除操作——合并的回溯 $m=3$





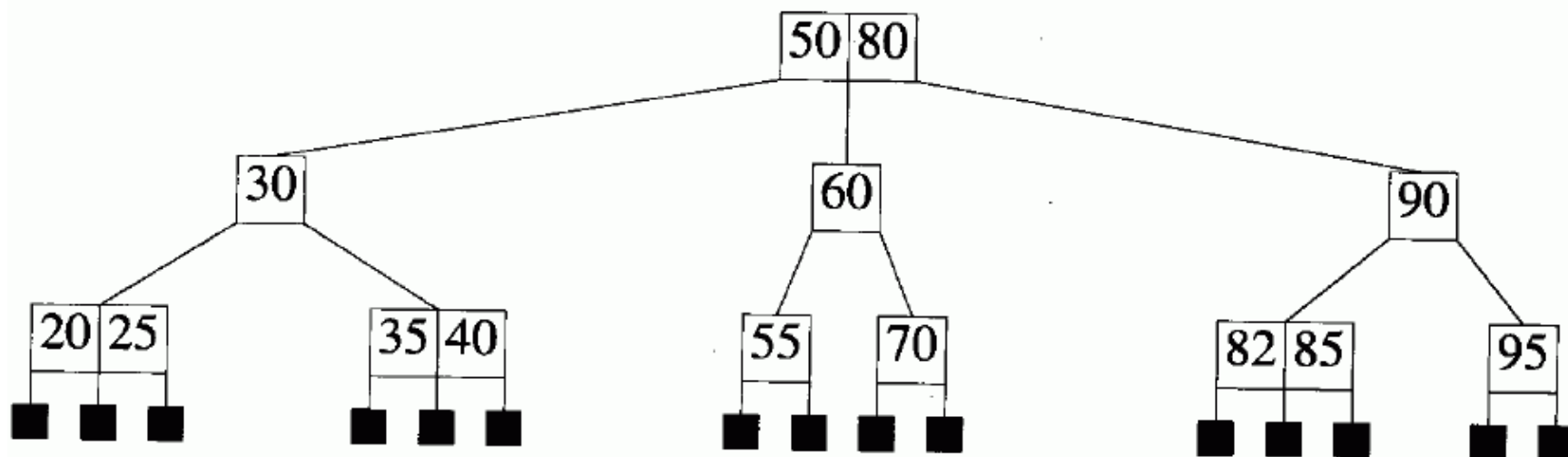
# 删除操作——例



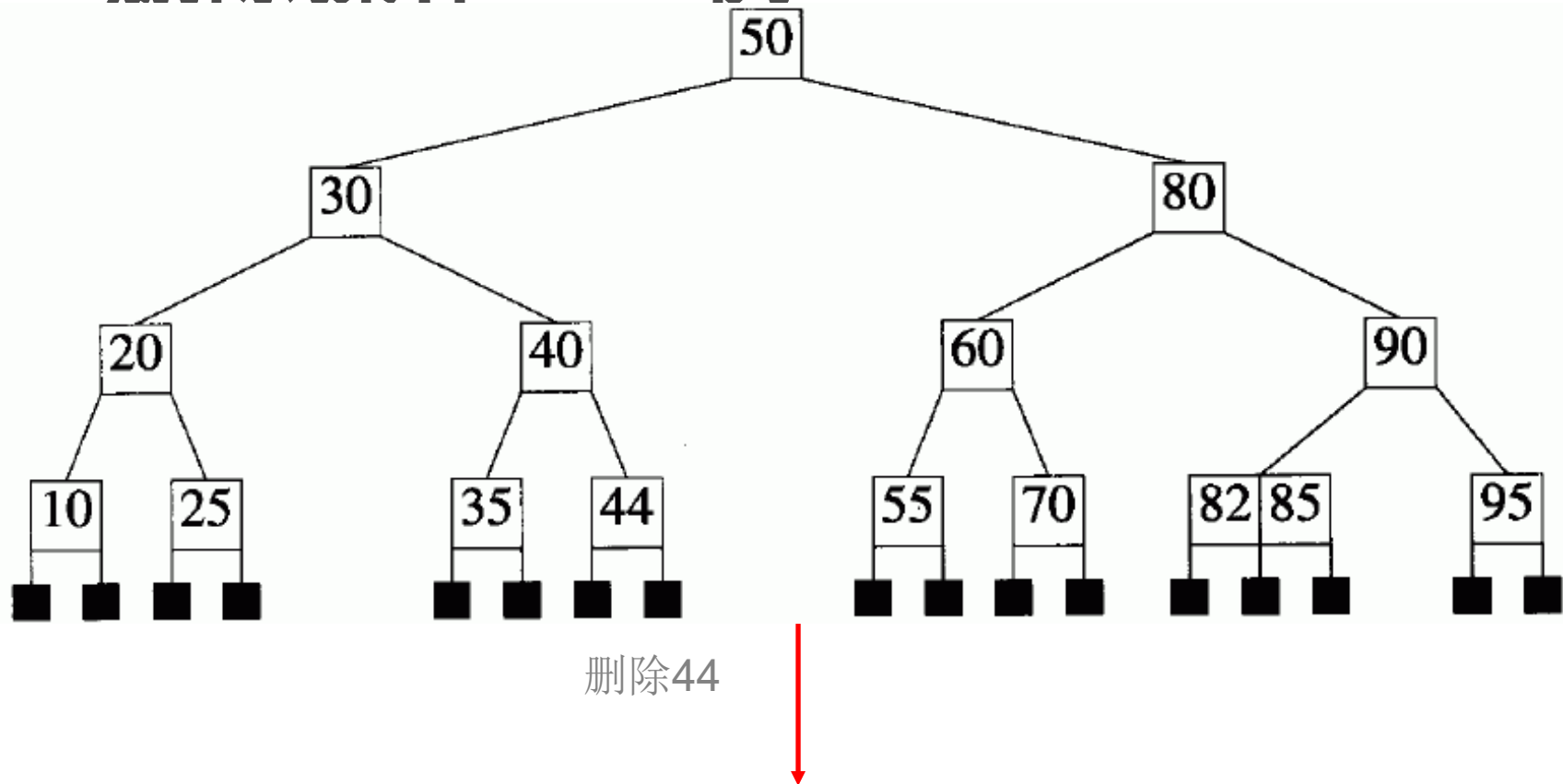
父节点采取“借用法”



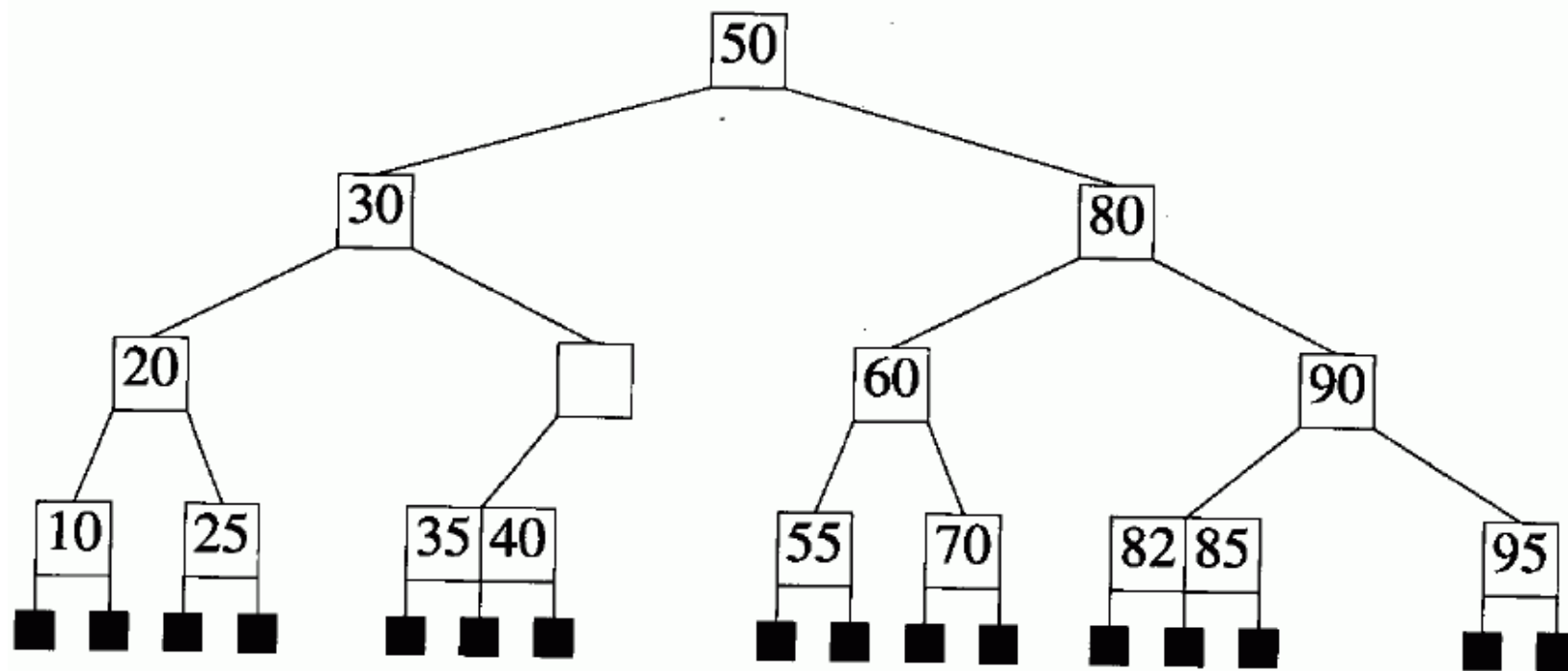
# 删除操作——例



# 删除操作——例



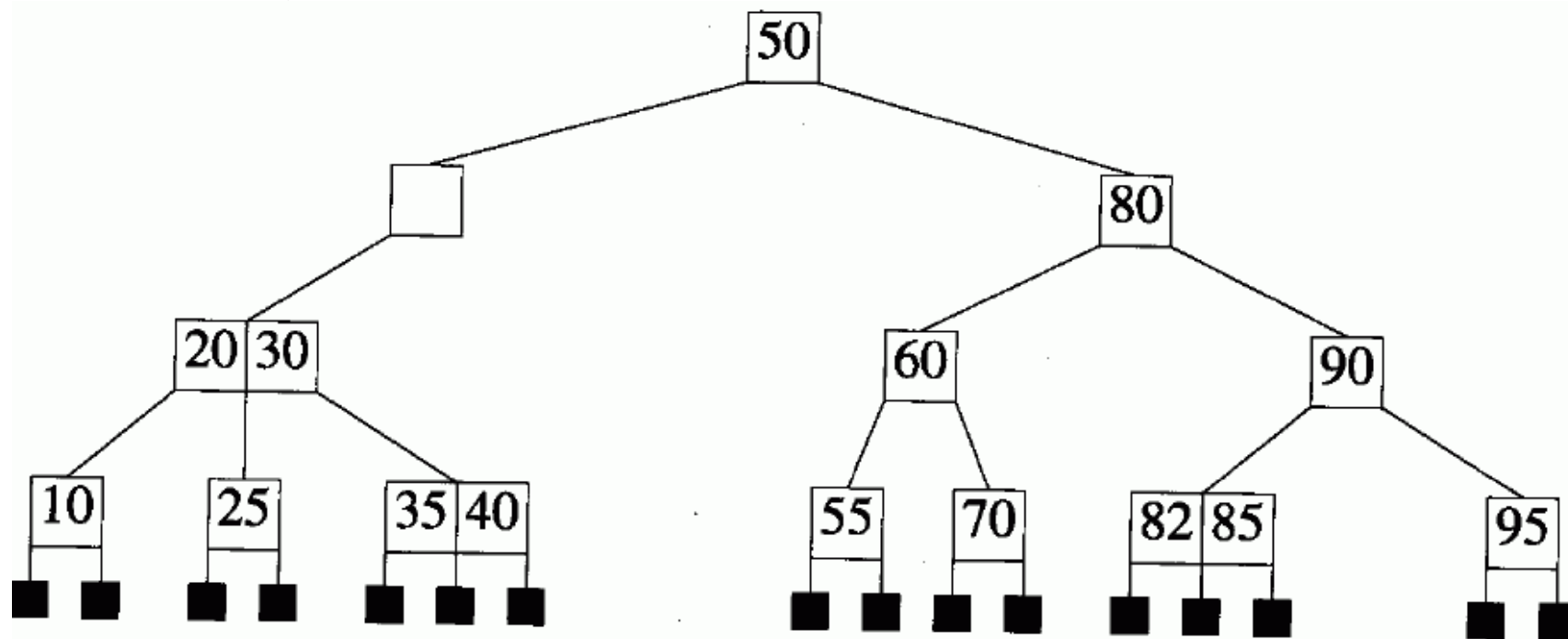
# 删除操作——例



20、30合并



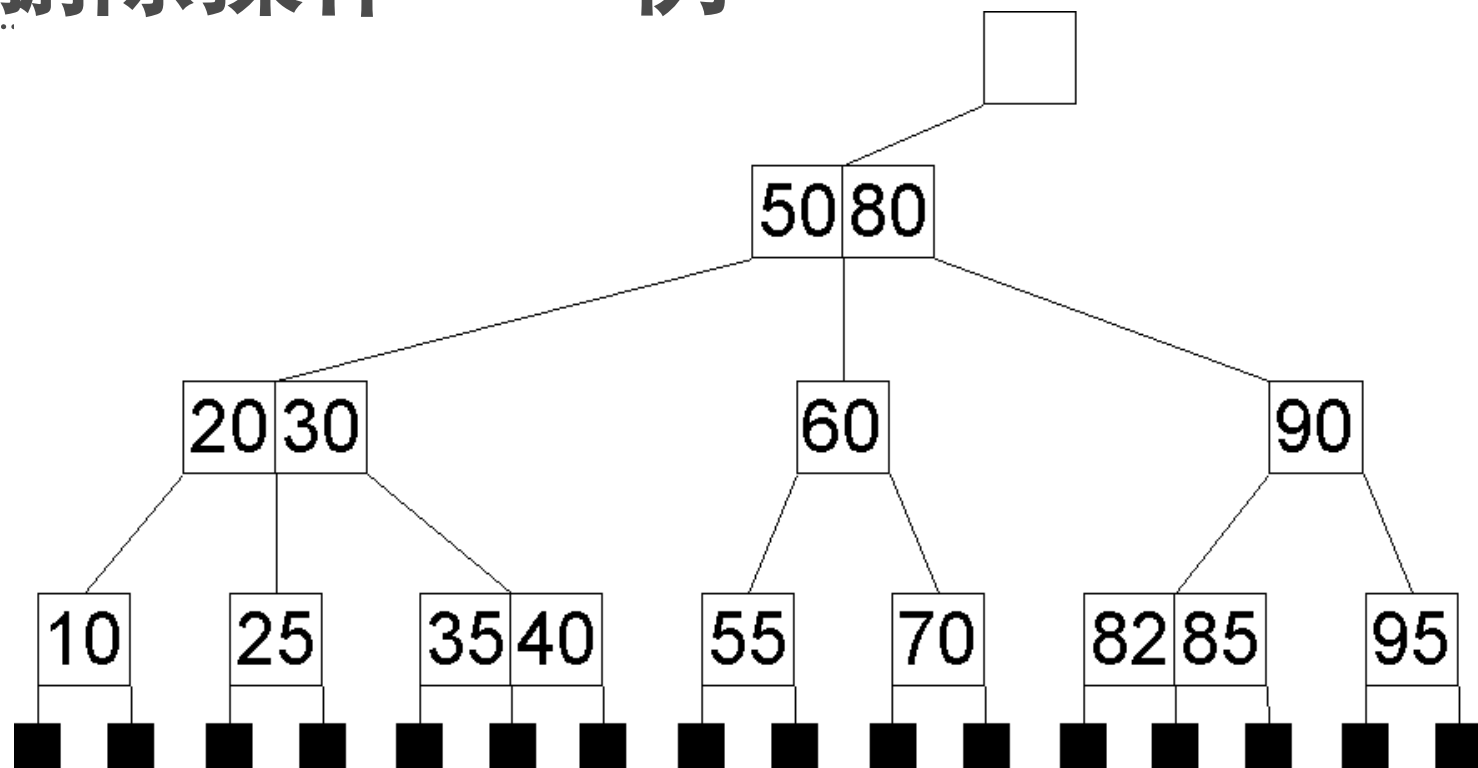
# 删除操作——例



继续合并，根节点变空

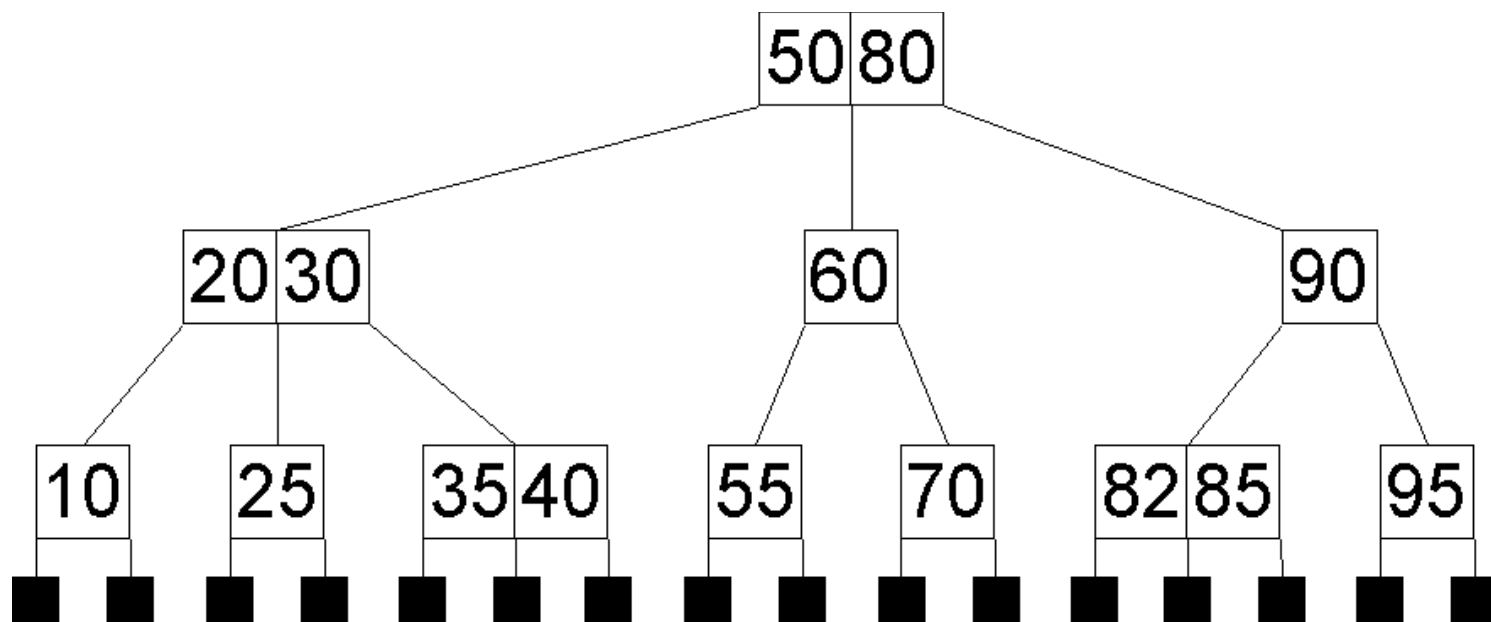


# 删除操作——例



# 删除操作——例

丢弃根节点，最终结果



# H1 小结

---

- n阶B树的插入
  - 需考虑违背n阶上限的情况
  - 处理方法：分裂
- n阶B树的删除
  - 需考虑违背n阶下限的情况
  - 处理方法：能借就借，不能借合并





# B<sup>+</sup>树定义

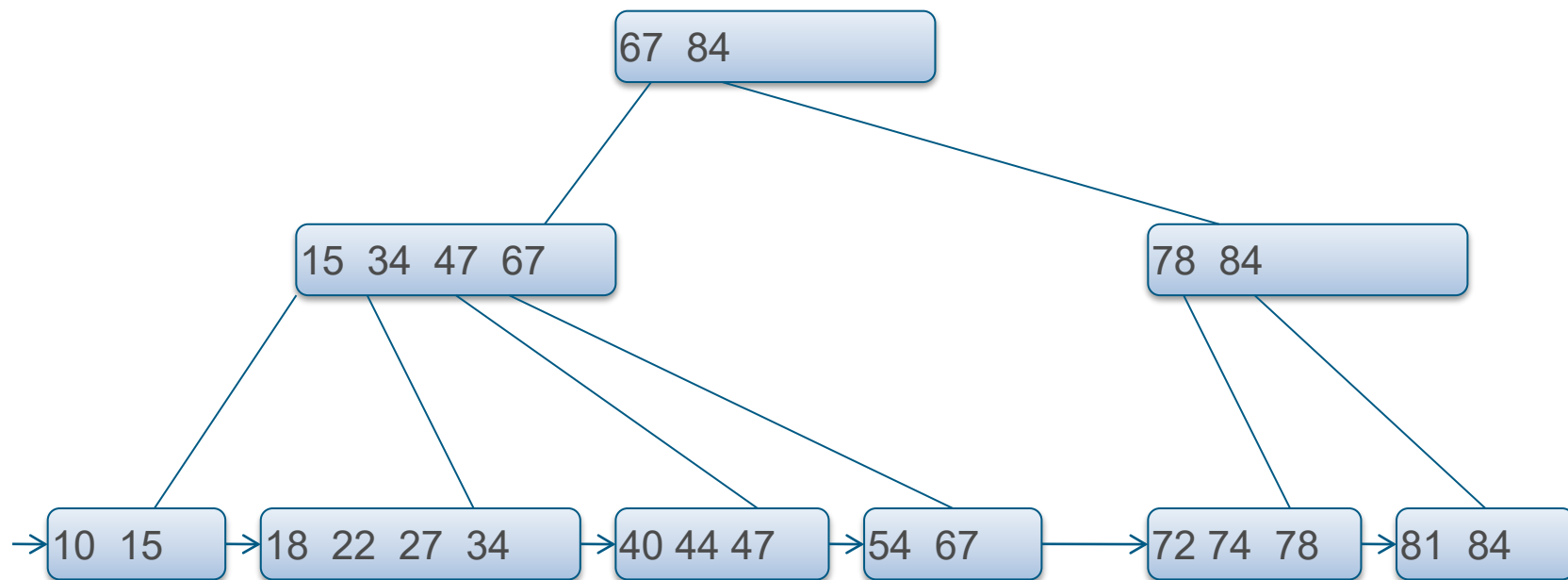
---

- B<sup>+</sup>树与B树的区别

- 所有关键字都放在叶节点中，上层的非叶结点的关键字是其子树中最大关键字的复写
- 叶节点包含了全部关键字，且叶节点本身按关键字值从小到大链接



# 4阶B+树示例



# B<sup>+</sup>树插入

---

- 通过查找，在叶节点的适当位置插入元素
- 如果插入后合法，结束
- 如果插入后非法，将该叶节点均匀分裂，更新其父节点
- 递归检查其父节点在更新后是否合法，直至结束。



# B+树插入示例

---

- 要求：连续插入十三个数，形成4阶B+树  
24, 72, 1, 39, 53, 63, 90, 88, 15, 10, 44, 68, 18

第1步：插入24,72,1,39

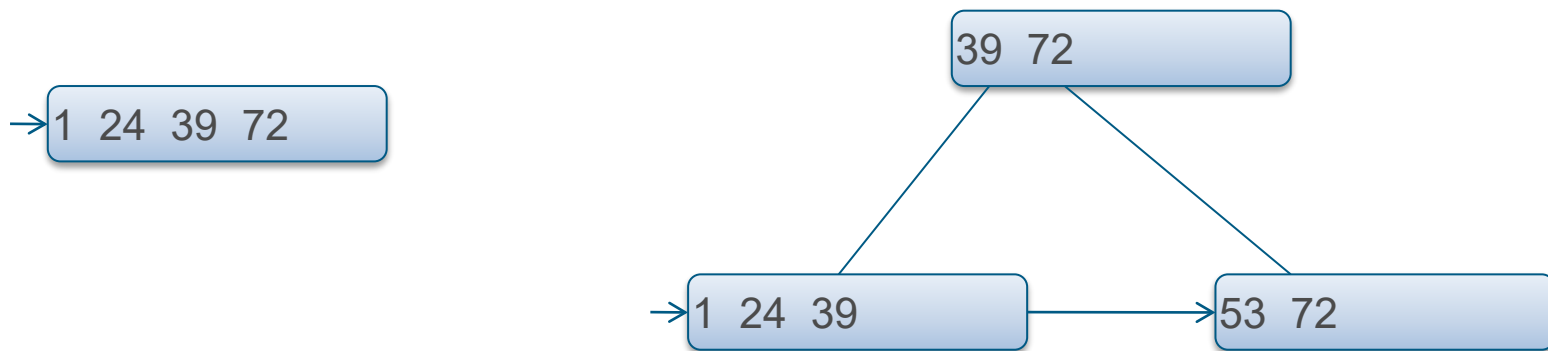
→ 1 24 39 72



# B+树插入示例

- 要求：连续插入十三个数，形成4阶B+树  
24, 72, 1, 39, 53, 63, 90, 88, 15, 10, 44, 68, 18

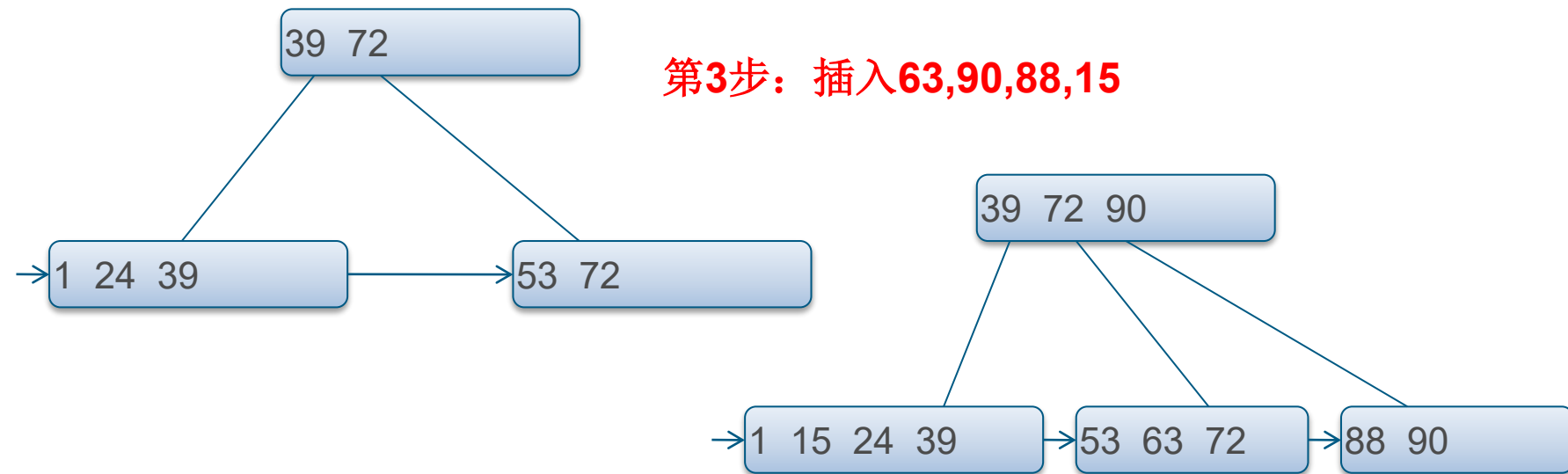
第2步：插入53



# B+树插入示例

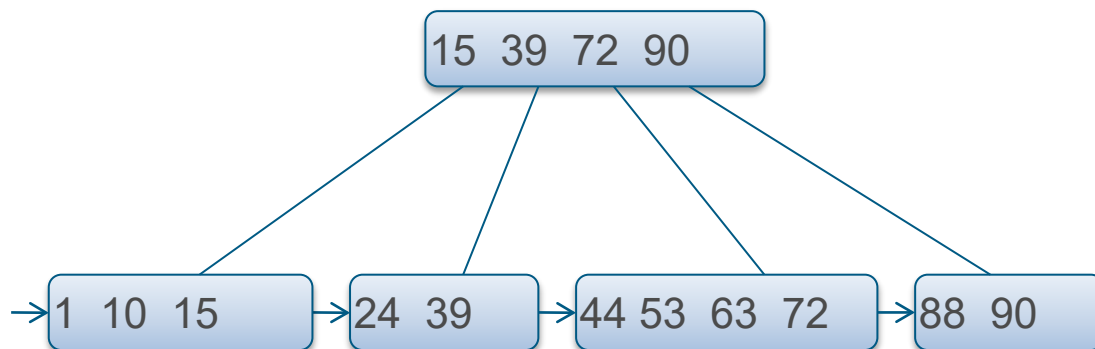
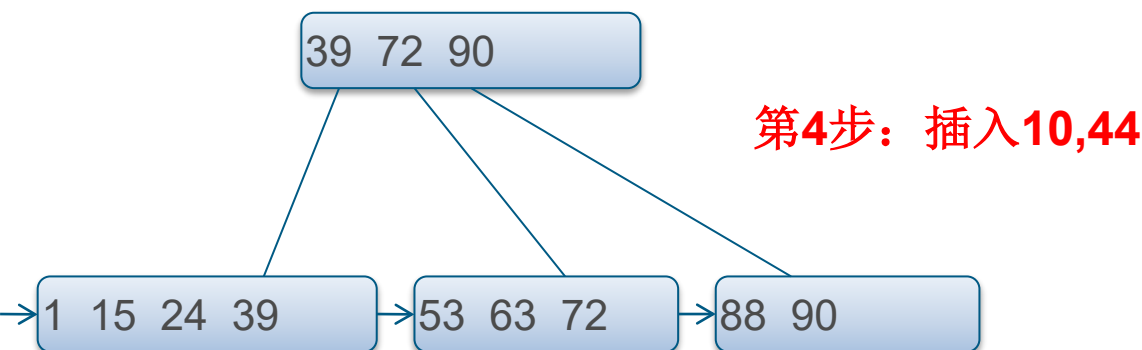
- 要求：连续插入十三个数，形成4阶B+树  
24, 72, 1, 39, 53, 63, 90, 88, 15, 10, 44, 68, 18

第3步：插入63,90,88,15



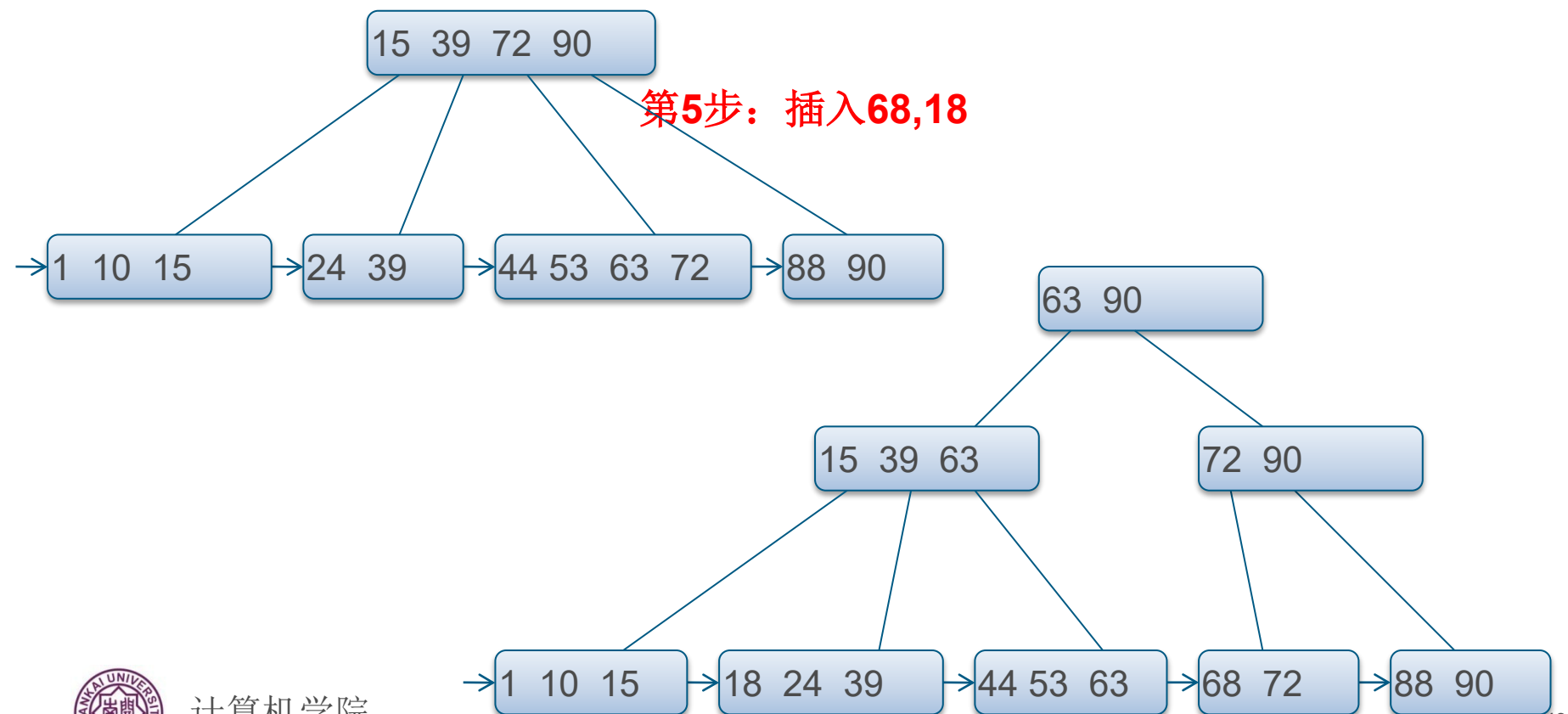
# B+树插入示例

- 要求：连续插入十三个数，形成4阶B+树  
24, 72, 1, 39, 53, 63, 90, 88, 15, 10, 44, 68, 18



# B+树插入示例

- 要求：连续插入十三个数，形成4阶B+树  
24, 72, 1, 39, 53, 63, 90, 88, 15, 10, 44, 68, 18





# B<sup>+</sup>树删除

---

- 首先在叶节点上进行删除
- 如果删除后该节点仍满足最小元素数要求，停止
- 如果不满足了，从兄弟节点借元素
- 如果没法借，则合并兄弟节点
- 递归考察父节点，直至结束



# 主要内容

---

- B树
- 红-黑树
  - 定义
  - 搜索
  - 插入
  - 删除



# 红-黑树定义

- 特殊的二叉搜索树，对扩充（外部节点）的二叉搜索树，满足以下条件
  - RB1：根节点和所有外部节点的颜色是黑的
  - RB2：根至外部节点的路径上没有连续红节点
  - RB3：所有根至外部节点的路径具有相同数目的黑节点



# 另一种定义方式

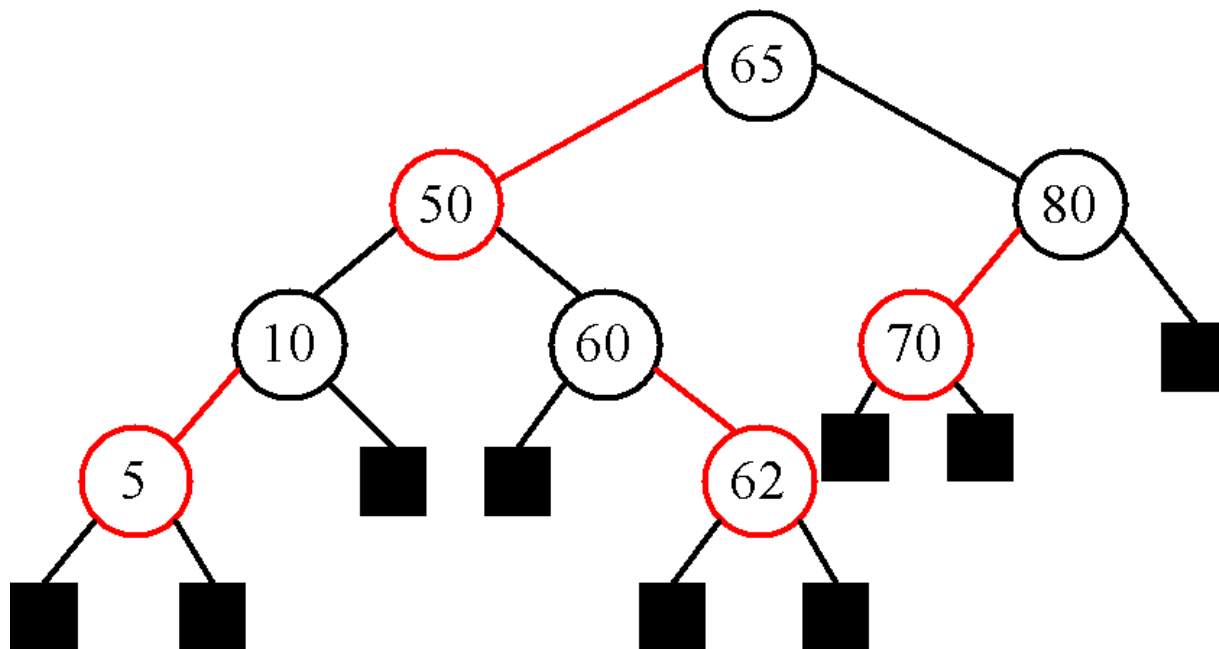
---

- 边（指针）的颜色
  - RB1' : 从内部节点指向外部节点的指针是黑的
  - RB2' : 从根至外部节点的路径上没有连续红指针
  - RB3' : 所有根至外部节点的路径具有相同数目的黑指针
- 黑指针指向的孩子节点是黑的  
红指针指向的孩子节点是红的



# 红-黑树例

- 节点的**阶**（rank）：从该节点到其子树中任一外部节点的路径上的黑色指针数目



# 红—黑树是怎么来的

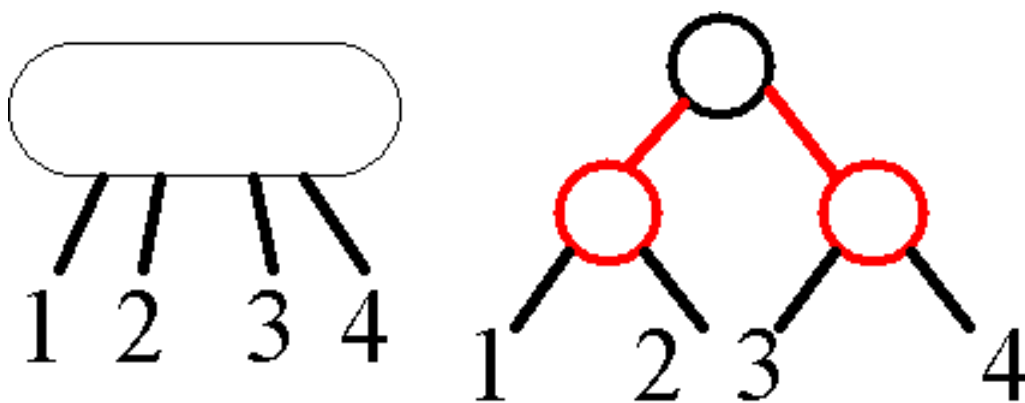
---

- 实际上因为2-3-4树实现比较复杂，仍然想通过二叉树结构描述2-3-4树
  - 2节点（一个关键字，两个孩子）用二叉树结构表示当然没问题
  - 3节点、4节点怎么办？多个关键字在二叉树中只能形成多个节点
  - 用红边描述——红边连接的节点中的关键字，在2—3—4树中是一个节点内的



# 2-3-4树的红-黑树表示

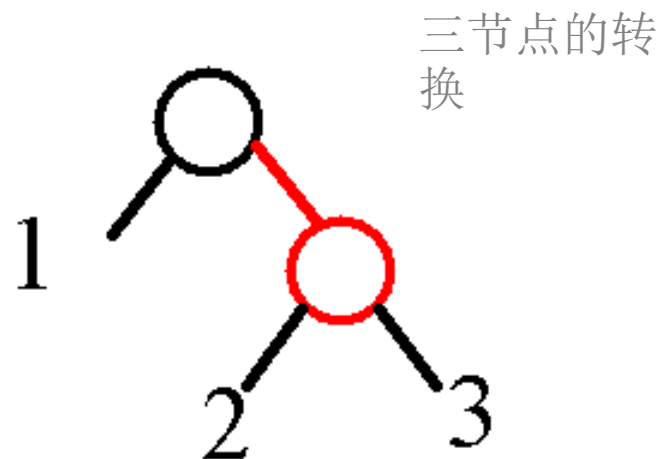
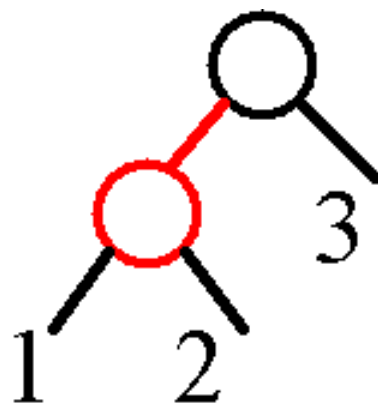
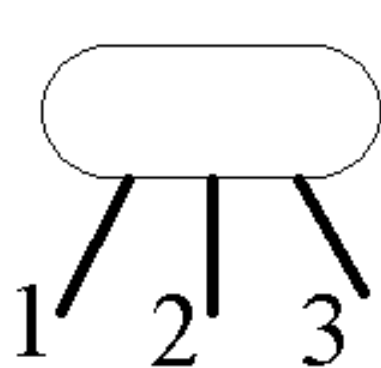
---



四节点的转  
换



# 2-3-4树的红-黑树表示



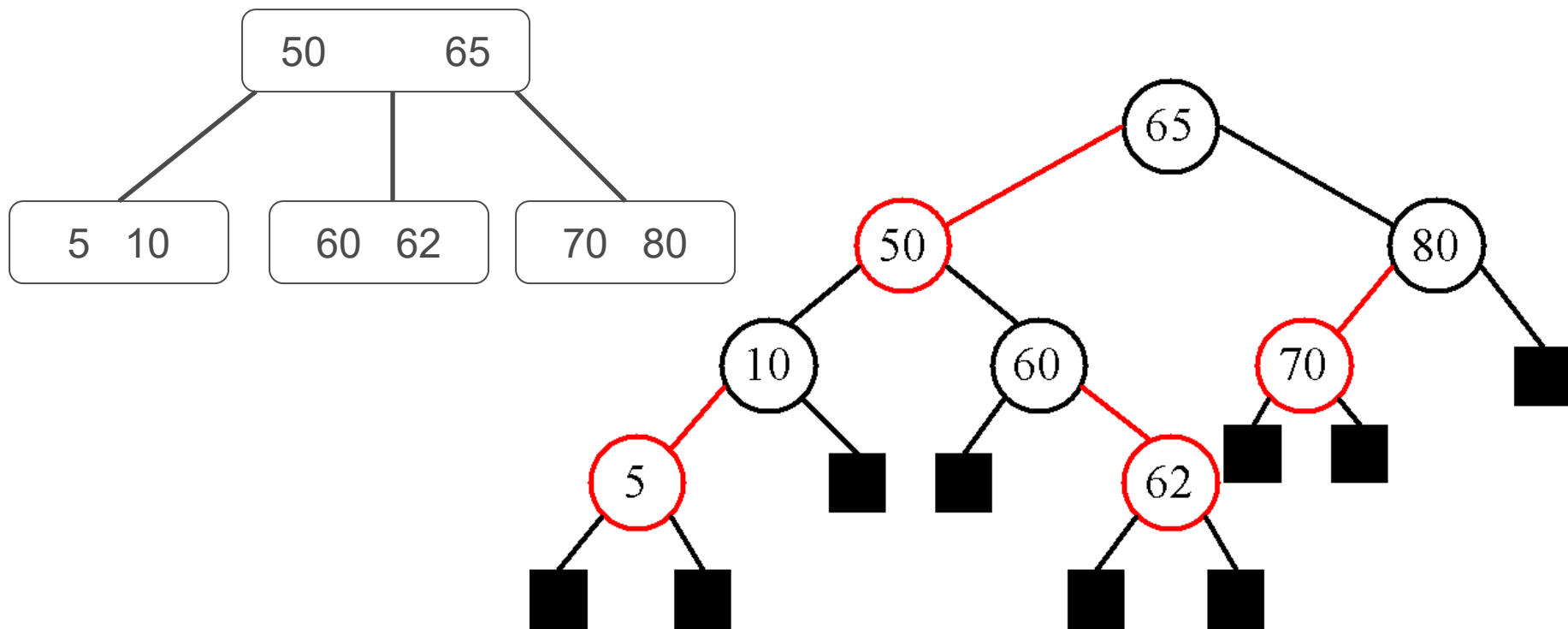
三节点的转换

即：红边上表示的是2-3-4树的同节点关系

黑边表示父子关系



# 2-3-4树的红-黑树表示



# 几个问题

---

- 为什么用节点和边的颜色表示是等价的？
- 为什么根和外部节点都是黑的？
- 为什么路径上没有连续红节点？
- 为什么每条路径上具有相同数量的黑指针？
- 了解了红-黑树的来历就一清二楚了



# 红—黑树特性1

- 定理11-1：设从根到外部节点的路径的长度（length）是该路径中指针的数量，若P、Q是红—黑树中两条从根至外部节点的路径，那么 $\text{length}(P) \leq 2\text{length}(Q)$

证明：

设根的阶为r——黑指针数为r



# 红—黑树特性1

---

由RB1' 知：每条路径最后一个指针为黑色的

由RB2' 知：所有路径都不包含连续红指针

→每个红指针后面都会紧跟一个黑指针

→红指针数目最多为 $r$  → 指针总数在 $r \sim 2r$ 之间

定理得证



# 红—黑树特性2

- 定理11-2：设 $h$ 是一棵红—黑树的高度（不包括外部节点）， $n$ 是树中内部节点的数目，而 $r$ 是根节点的阶，则有
  - 1)  $h \leq 2r$ ——由定理11-1显然
  - 2)  $n \geq 2^r - 1$   $\longleftarrow$  根的阶为 $r \rightarrow$  树高至少为 $r$
  - 3)  $h \leq 2 \log_2(n+1)$   $\longleftarrow$  由2可得 $r \leq \log_2(n+1)$ ，再结合1即可得证



# 红-黑树的描述

---

- 外部节点无需实际保存
- 每个节点需保存其颜色和两个指针的颜色——最多需3个二进制位



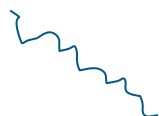
# 红黑树如何绘制？

---

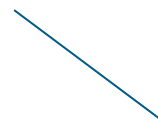
- 节点表示



- 指针表示（可选）



红色



黑色



# 红-黑树的搜索

---

- 与二叉搜索树相同
- 复杂性 $O(\log n)$





## H2. 红—黑树的插入

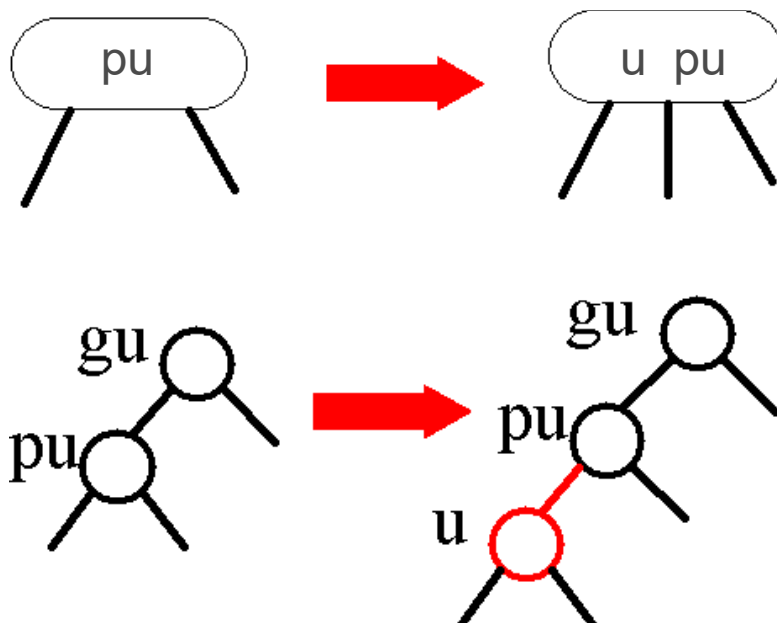
---

- 首先还是简单的二叉搜索树插入方法
- 然后有一个着色的问题——新节点应该是黑色还是红色？
- 应该是红色，为什么？——考虑2-3-4树（B-树）的插入
- 可能会出现连续红边，显然要调整树的结构，如何调整？——考虑对应的2-3-4树（B-树）插入的情况



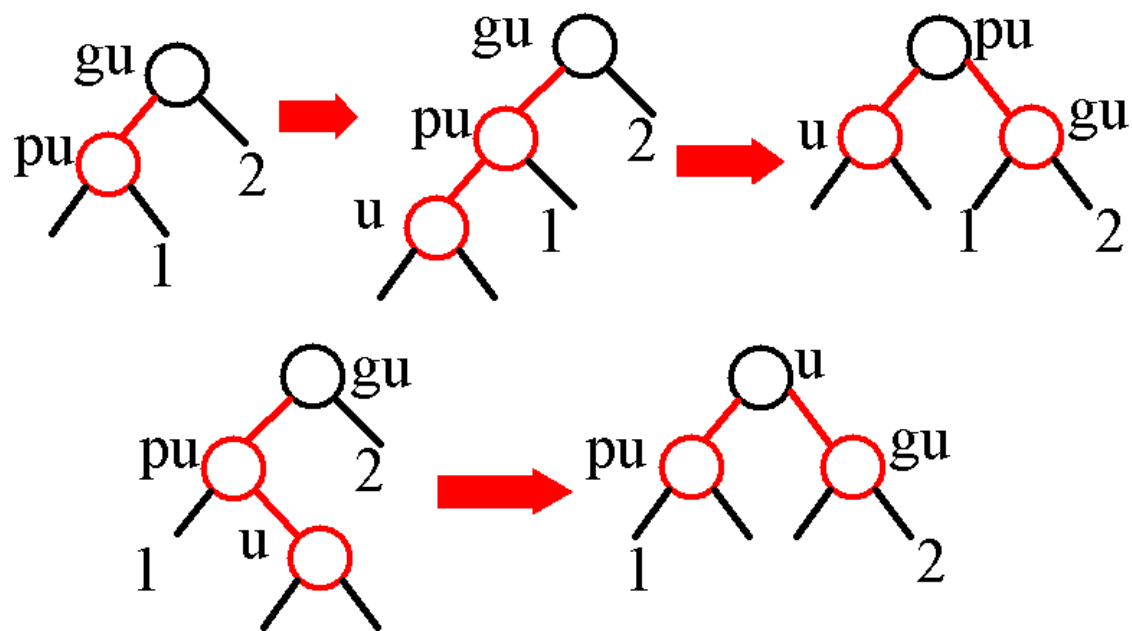
# 没有连续红边的情况（最理想）

- 对应2-3-4树的情况应该是2节点变为3节点



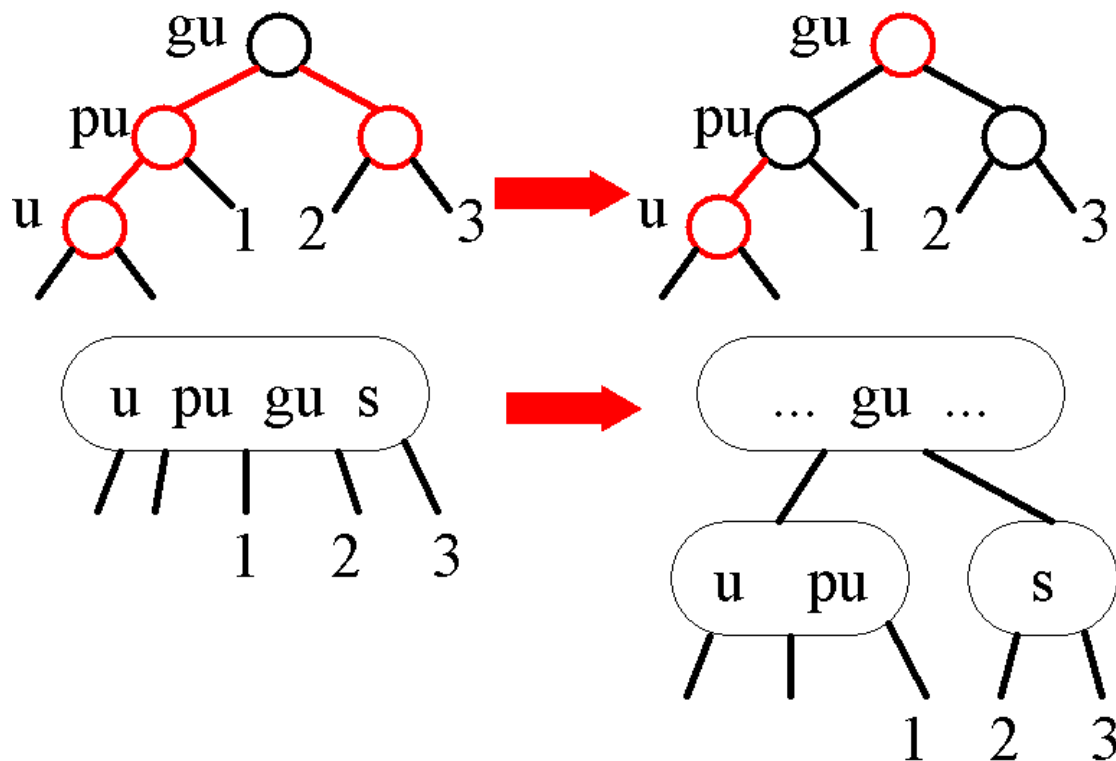
# 连续红边情况1：XYb

- 祖父的另一个孩子是黑色—3节点变为4节点
- 旋转变色：根黑、子红



## 连续红边情况2: XYr

- 祖父的另一个孩子是红色——溢出
- 不转变色: 根红、子黑



# 例题

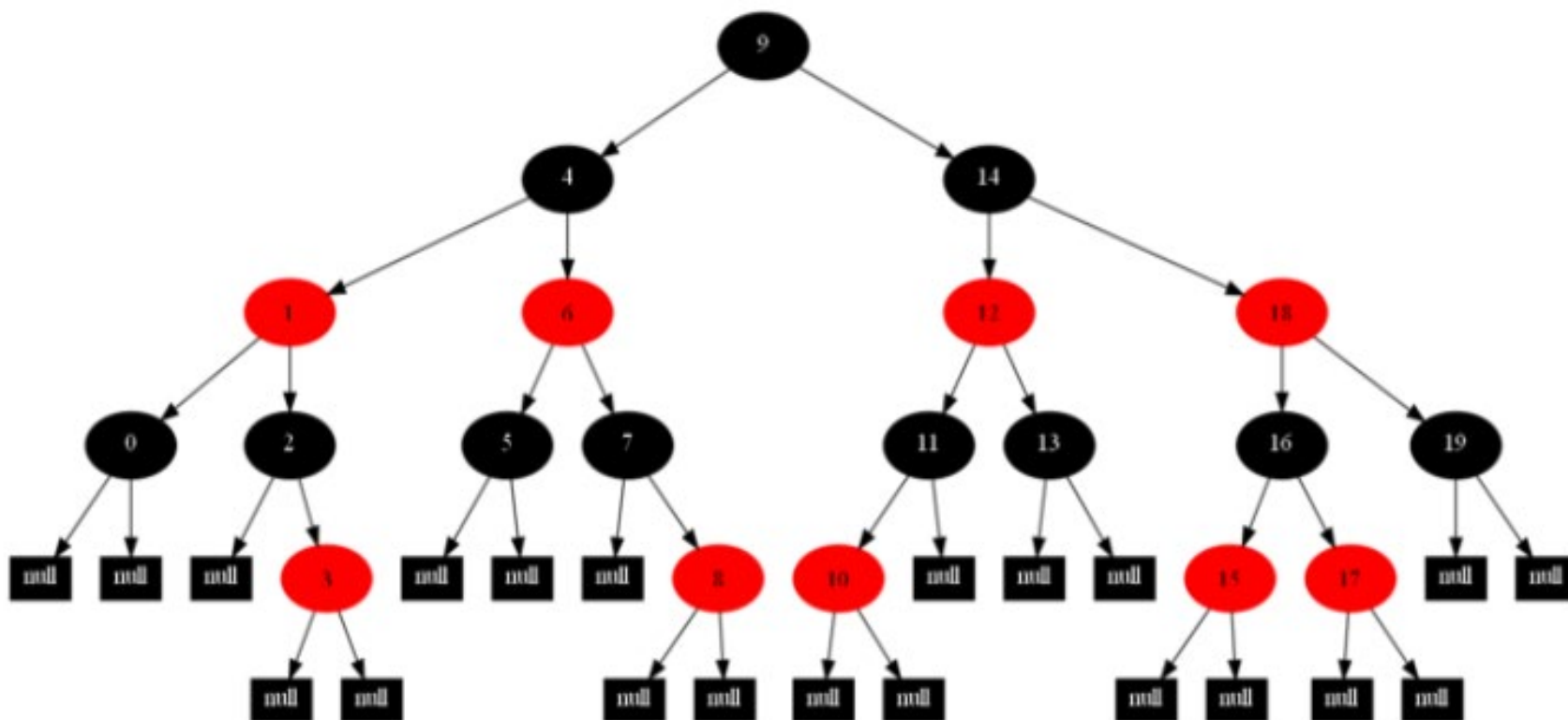
---

- 教材P370，将70, 60, 65, 62依次插入到15-13a)所示的红黑树中



# H2结束

- 练习：依次插入这些12 1 9 2 0 11 7 19 4 15 18 5 14 13 10 16 6 3 8 17，生成红黑树



# 红—黑树的删除

---

- 首先还是类似二叉搜索树的方法——  
转化为叶节点删除——  
不是叶节点的，与中序遍历的后继节点交换
- 可对应2-3-4树自底向上删除算法设计红—黑树自底向上的删除算法
- 若删除红节点，结束——3节点或4节点的删除
- 若删除黑节点，调整，可能回溯——2节点的删除



# 例题

---

- 有以下关键字：28， 72， 97， 23， 11， 63， 4， 53， 84， 32， 61， 52， 按序插入到初始为空的4阶B树中。依次绘制以下的结构：
  - (1) 4阶B树
  - (2) 红黑树
  - (3) 还原的森林





---

# 本章结束

