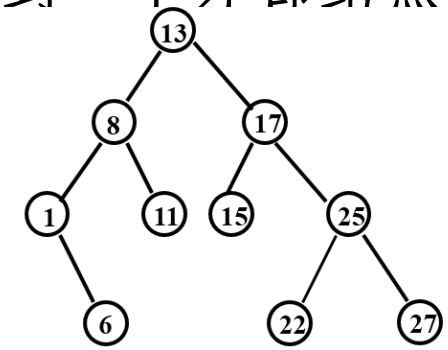
6.4 Red-Black Tree

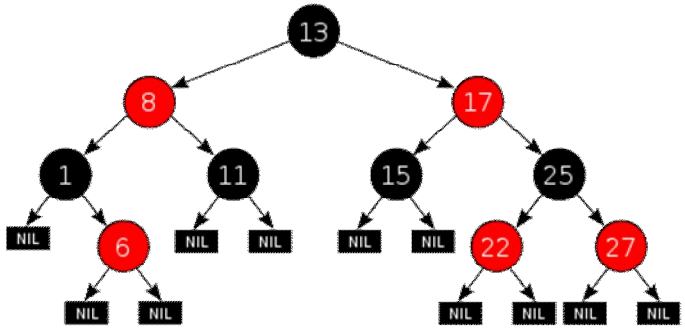
- 红黑树是满足特定结构要求的二叉树：
  n个结点的红黑树的高度不超过2log(n+1)
- 主要用于实现动态查找表
- 主要操作：插入、删除、查找

6.4 Red-Black Tree

- 红黑树是每个结点都带有颜色属性的**二叉排序树**，颜色或红色或黑色：

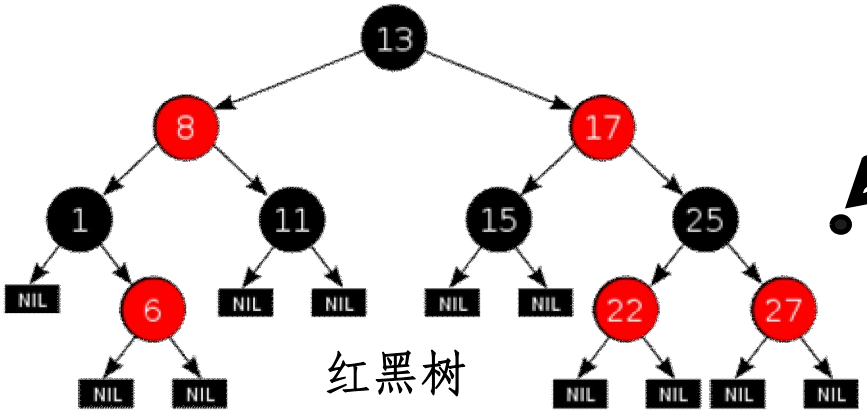
1. 所有的**外部结点**都是黑色： **空树，查找失败**
2. 连向黑结点的边成为**黑边black edge**
3. 一条路径的black length为该路径上black edge的数量
4. 结点的black depth为从根结点到该结点路径的black length
5. 从一个结点到一个外部结点路径称为**外部路径**



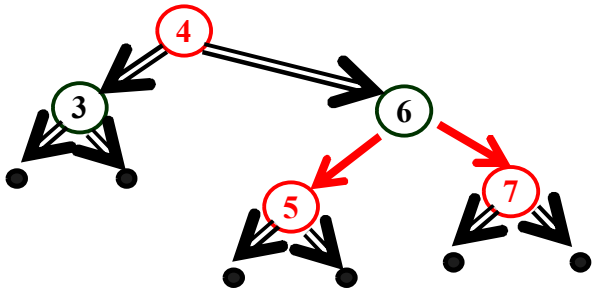


6.4 Red-Black Tree

- 红黑树
  - 根结点是黑色
  - 红结点没有红孩子
  - 树中一个给定结点*u*，其所有外部路径的black length相同，该值称为*u*的black height
- 近似红黑树----根结点是**红色**的，其余的条件同红黑树



红黑树



近似红黑树

6.4 Red-Black Tree

- **红黑树定义：**

1. 二叉排序树
2. 结点是红色或黑色
3. 根结点是黑色
4. 所有**外部结点**都是黑色
5. 每个红色节点的两个子结点都是黑色。不能有两个连续的红色节点
6. 树中每一个结点*u*，其所有外部路径的black length相同

6.4 Red-Black Tree

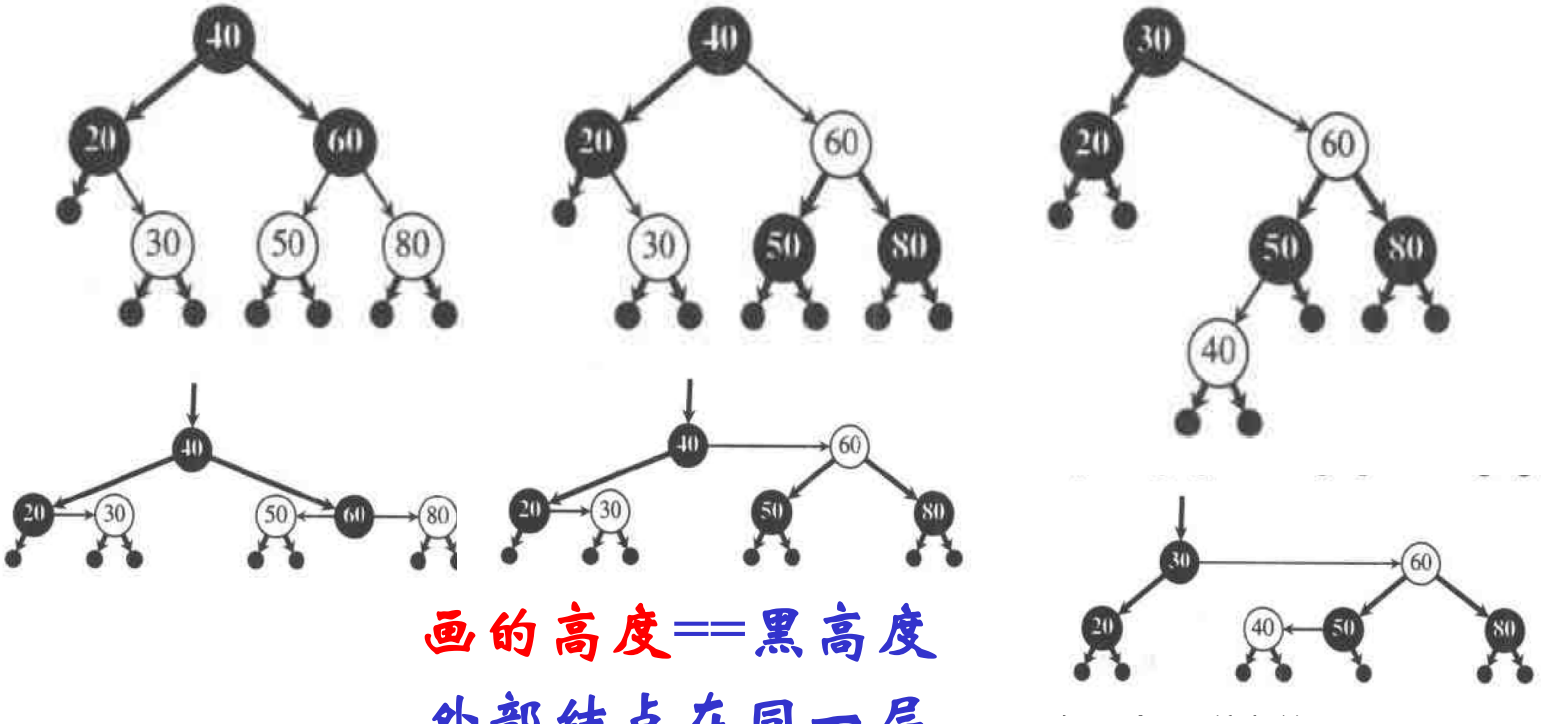
- 红黑树的关键性质：

  从根到叶子的最长的可能路径不多于最短的可能路径的两倍长。

- 这个树大致上是平衡的。
- 因为操作比如插入、删除和查找某个值的最坏情况时间都要求与树的高度成比例，这个在高度上的理论上限允许红黑树在最坏情况下都是高效的，而不同于普通的二叉排序树。

6.4 Red-Black Tree --*drawing*

- 红结点和其父亲在同一层上----***black depth convention***



画的高度==黑高度

外部结点在同一层

6.4 Red-Black Tree

- Definition RB_h和ARB_h
- 下列结点颜色为红或黑、外部结点为黑色的二叉排序树是RB_h和ARB_h：

  (1)一个外部结点是一个RB_0

  (2) h≥1时，一个二叉排序树是ARB_h ----其根为红色，左右子树均为RB_{h-1}

  (3) h≥1时，一个二叉排序树是RB_h ----其根为黑色，左右子树均为RB_{h-1}或ARB_h

6.4 Red-Black Tree

- 引理6.1 RB_h 树和 ARB_h 树的黑色高度为h.
- 引理6.2
  - 设T为RB_h
    - T至少有2^h-1个内部黑结点
    - T至多有4^h-1个内部结点
    - 任一黑结点的深度至多为其黑深度的2倍
  - 设A为ARB_h
    - A至少有2^h-2个内部黑结点
    - A至多有$(4^h-1)/2$个内部结点
    - 任一黑结点的深度至多为其黑深度的2倍

Theorem 6.3 Let T be a red-black tree with n internal nodes. Then no node has depth greater than $2\lg(n+1)$. In other words, the height of T in the usual sense is at most $2\lg(n+1)$.

- 设T为有n个内部结点的红黑树，没有结点深度大于2log(n+1)，即树的高度最多2log(n+1)
- 证明：设T有m个内部黑结点，则m≤n
- 由引理6.2，RB_h至少有2^h-1个内部黑结点，$2^h-1 \le m \le n$
- $2^h \le n+1$
- 黑高度h ≤log(n+1)
- 由引理6.2，高度 ≤2log(n+1)

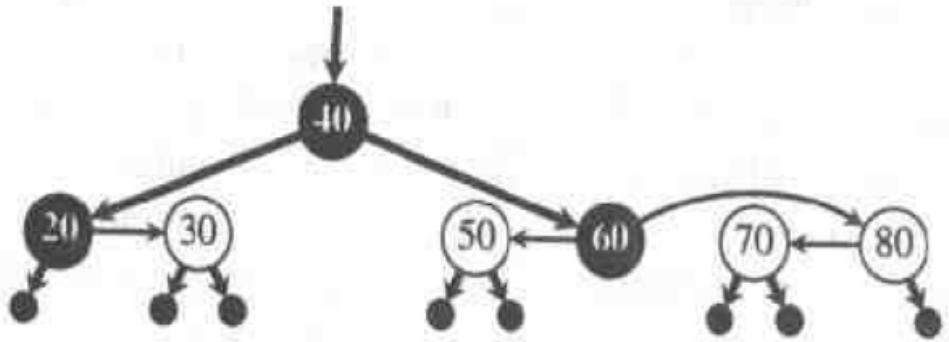
6.4 Red-Black Tree --插入

- 插入位置----根据二叉排序树的定义确定
- 新插结点初始颜色为红色
- 若新插结点为根结点，将其改为黑色，插入操作结束；否则若其父结点为黑色，则插入操作结束；否则进行调整

不妨设新插结点为N

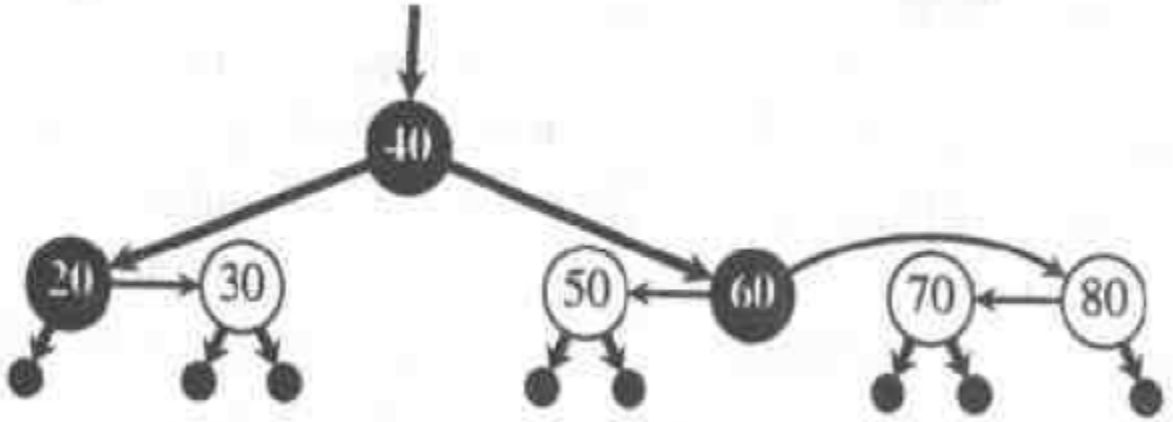
6.4 Red-Black Tree --插入

- 簇cluster：内部结点集，包含一个**黑**结点，所有由该黑结点通过非黑边能够达到的**红结点**
- 临界簇：如果簇中存在一个结点由簇的根到该结点路径长度大于1。



6.4 Red-Black Tree --插入

- 在插入过程中，如果树中不存在**临界簇**，那么符合红黑树的定义，不需要调整



父结点P是红色的----临界簇

临界簇中有4个结点

- 如果叔父节点U也是红色，则将改变P和U的颜色为黑色，改变其祖父节点G为红色

  现在新节点N有了一个黑色的父节点P。因为通过父节点P或叔父节点U的任何路径都必定通过祖父节点G，在这些路径上的黑节点数目没有改变。

  红色的祖父节点G的父结点也有可能是红色的，将G当成是新插入的节点继续进行调整

父结点P是红色的----临界簇

临界簇中有3个结点

- 叔父结点U是黑色或缺少
- 新插结点N的父结点P是P的父结点G的左孩子
  - 新结点N是其父结点P的左孩子----右单旋转----调换P和其父结点的颜色;
  - 新结点N是其父结点P的右孩子----先左后右双旋转----同时修改颜色;
- 父结点P是P的父结点G的右孩子
  - 新结点N是其父结点P的右孩子----左单旋转----调换P和其父结点的颜色;
  - 新结点N是其父结点P的左孩子----先右后左双旋转----同时修改颜色;

新插结点

新插结点N的父结点P是P的父结点G的右孩子

新插结点N的父结点P是P的父结点G的左孩子

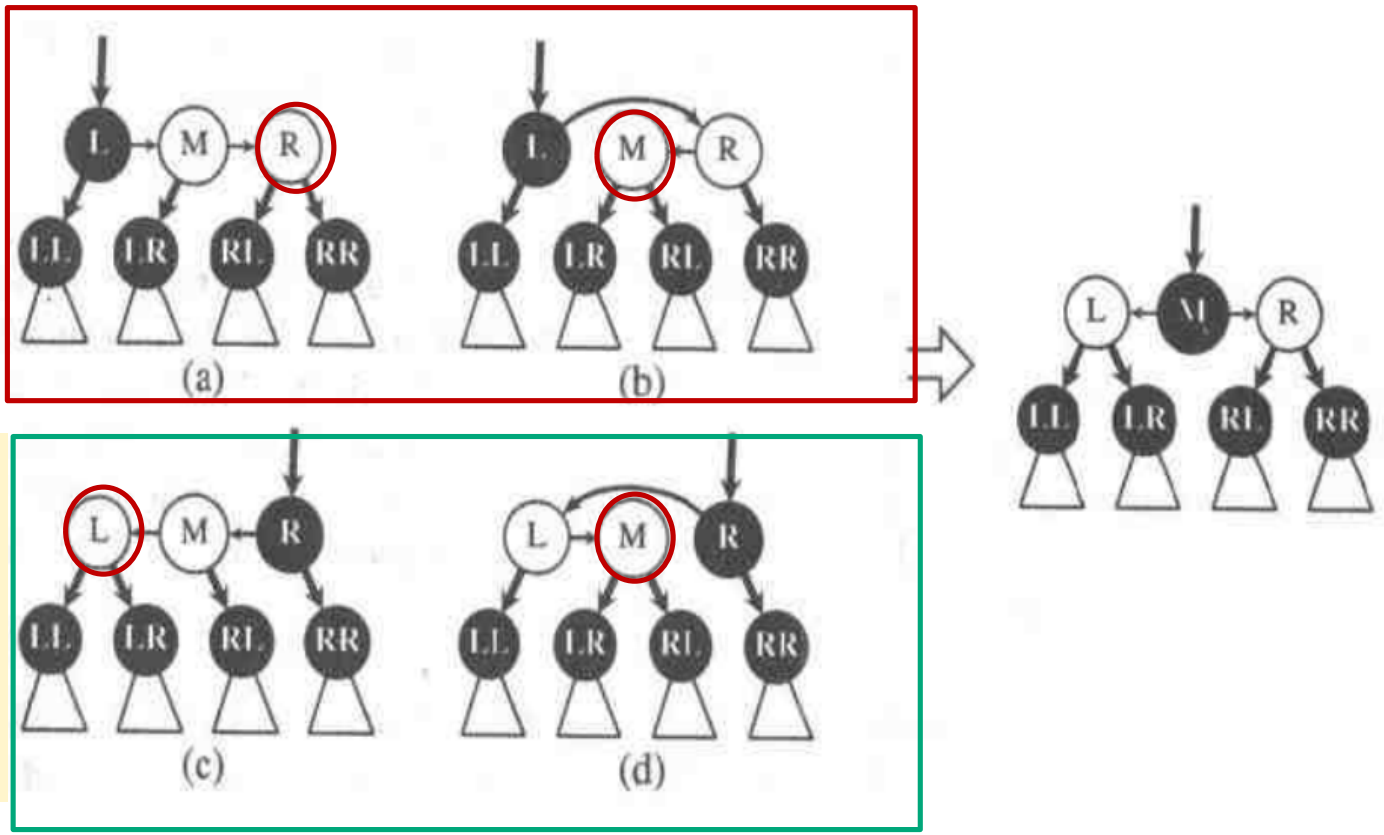
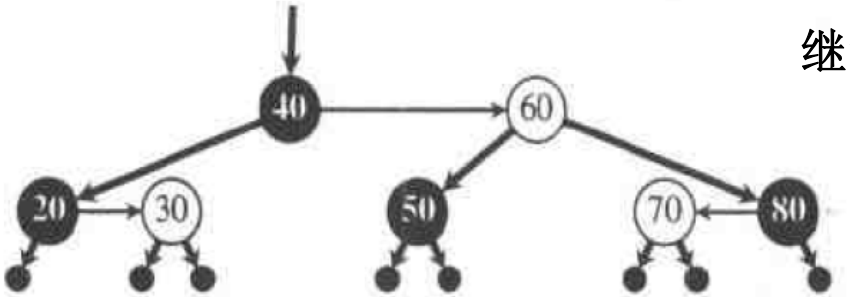
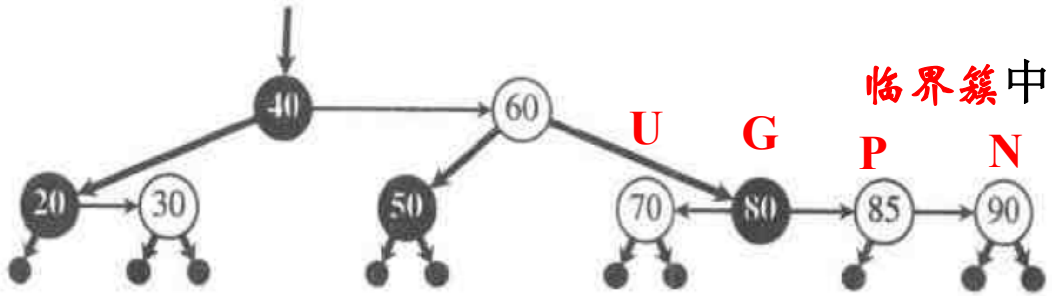


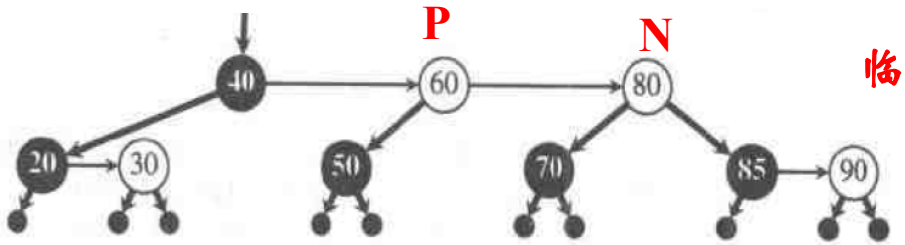
Figure 6.9 Rebalancing repairs any critical cluster of three nodes. Four possible initial arrangements, (a) through (d), become the same final arrangement, right.

继续插入85，90





临界簇中有4个结点



临界簇中有3个结点