



Data Structures

Ch7

查找 Searching

2023 年 11 月 28 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

7-3-2 平衡二叉树

平衡二叉(查找)排序树 (Balanced Binary Sort Tree)

1989奥斯卡最佳短片 Balance

7.3 树表的查找技术

7-3-2 平衡二叉树

1. 平衡二叉树

AVL树（由 Adelson-Velsky 和 Landis 共同发明）

为了实现二叉排序树的平均查找长度和 $\log n$ 等数量级，需要对二叉排序树进行“平衡化”处理，即构造平衡二叉树。

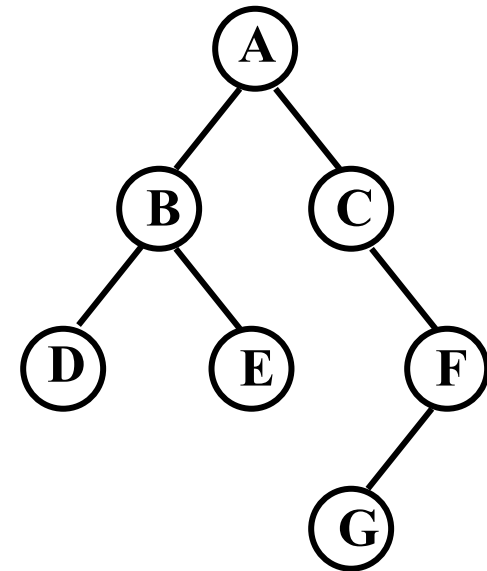
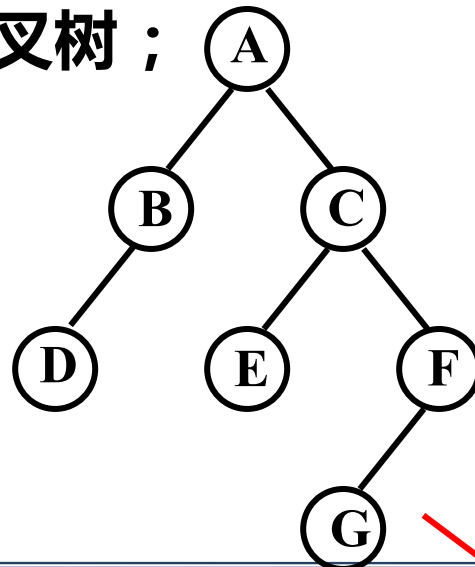
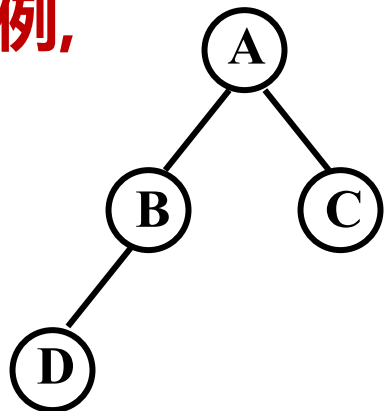
平衡二叉树首先是一棵二叉排序树；

平衡二叉树或者是一棵空树，或者是具有下列性质的二叉树：

其左子树和右子树的深度之差的绝对值不超过 1；

其左子树和右子树都是平衡二叉树；

例，



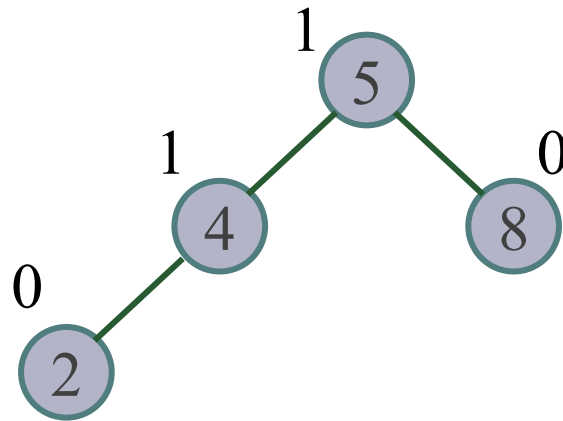
7.3 树表的查找技术

7-3-2 平衡二叉树

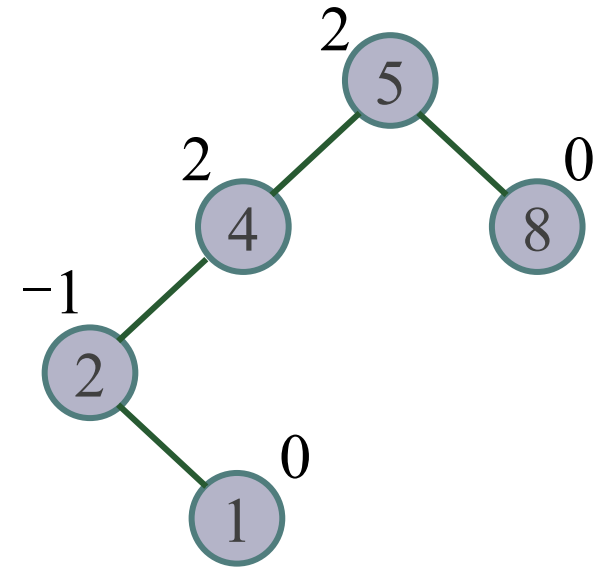
1. 平衡二叉树

 **平衡因子**：该结点的左子树的深度减去右子树的深度

在平衡二叉树中，
结点的平衡因子取值只能为：
1、0 或 -1



平衡二叉树



非平衡二叉树

7.3 树表的查找技术

7-3-2 平衡二叉树

2. 最小不平衡子树

🕒 插入一个结点会影响哪些结点的平衡因子？

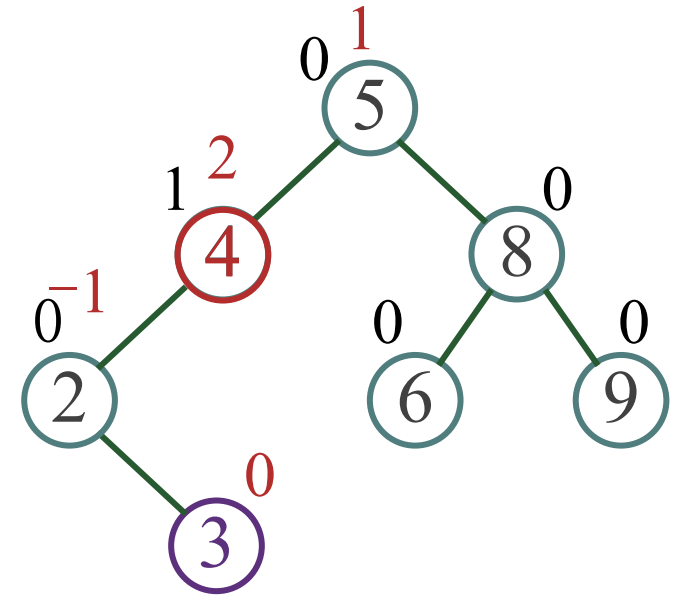
📌 **最小不平衡子树**：以距离插入结点最近的、且平衡因子的绝对值大于 1 的结点为根的子树

且入且判断，一旦失衡立即调整



只调整最小不平衡子树，并且不影响其他结点

🕒 如何调整最小不平衡子树呢？





7.3 树表的查找技术

7-3-2 平衡二叉树

3. 平衡调整算法

如果在一棵AVL树中插入一个新结点，就有可能造成失衡，此时必须**重新调整树的结构**，使之恢复平衡。我们称调整平衡过程为**平衡旋转**。

算法：平衡调整

输入：平衡二叉树，新插入结点A

输出：新的平衡二叉树

1. 找到最小不平衡子树的根结点 D
2. 根据结点A和结点D之间的关系，判断调整类型
3. 根据类型、遵循**扁担原理**和**旋转优先**原则进行相应调整
 - (1) LL型、RR型：调整一次
 - (2) LR型、RL型：调整两次

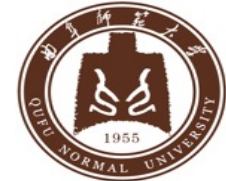
✓ LL平衡旋转

✓ RR平衡旋转

✓ LR平衡旋转

✓ RL平衡旋转

原则：保证二叉排序树的次序不变

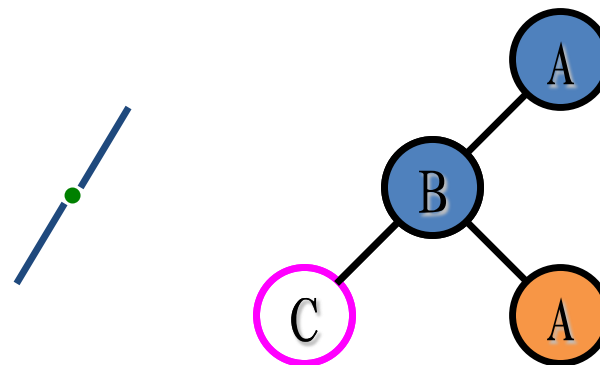


3. 平衡调整算法

1) LL平衡旋转:

若在A的左子树的左子树上插入结点，使A的平衡因子从1增加至2，需要进行一次顺时针旋转。

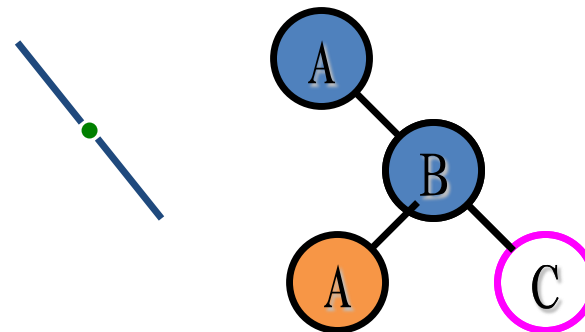
(以B为旋转轴)



2) RR平衡旋转:

若在A的右子树的右子树上插入结点，使A的平衡因子从-1增加至-2，需要进行一次逆时针旋转。

(以B为旋转轴)



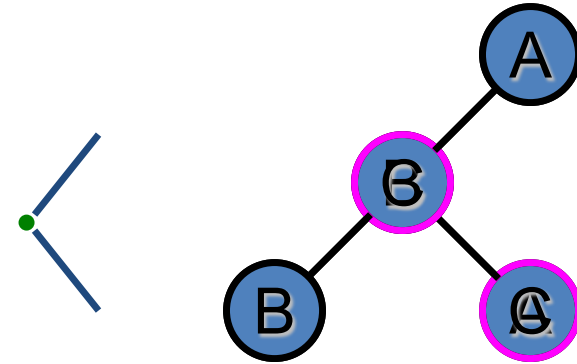
7.3 树表的查找技术

7-3-2 平衡二叉树

3. 平衡调整算法

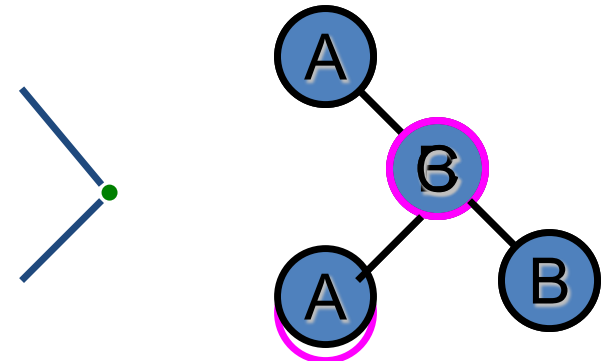
3) LR平衡旋转:

若在A的左子树的右子树上插入结点, 使A的平衡因子从1增加至2, 需要先进行逆时针旋转, 再顺时针旋转。
(以插入的结点C为旋转轴)



4) RL平衡旋转:

若在A的右子树的左子树上插入结点, 使A的平衡因子从-1增加至-2, 需要先进行顺时针旋转, 再逆时针旋转。
(以插入的结点C为旋转轴)

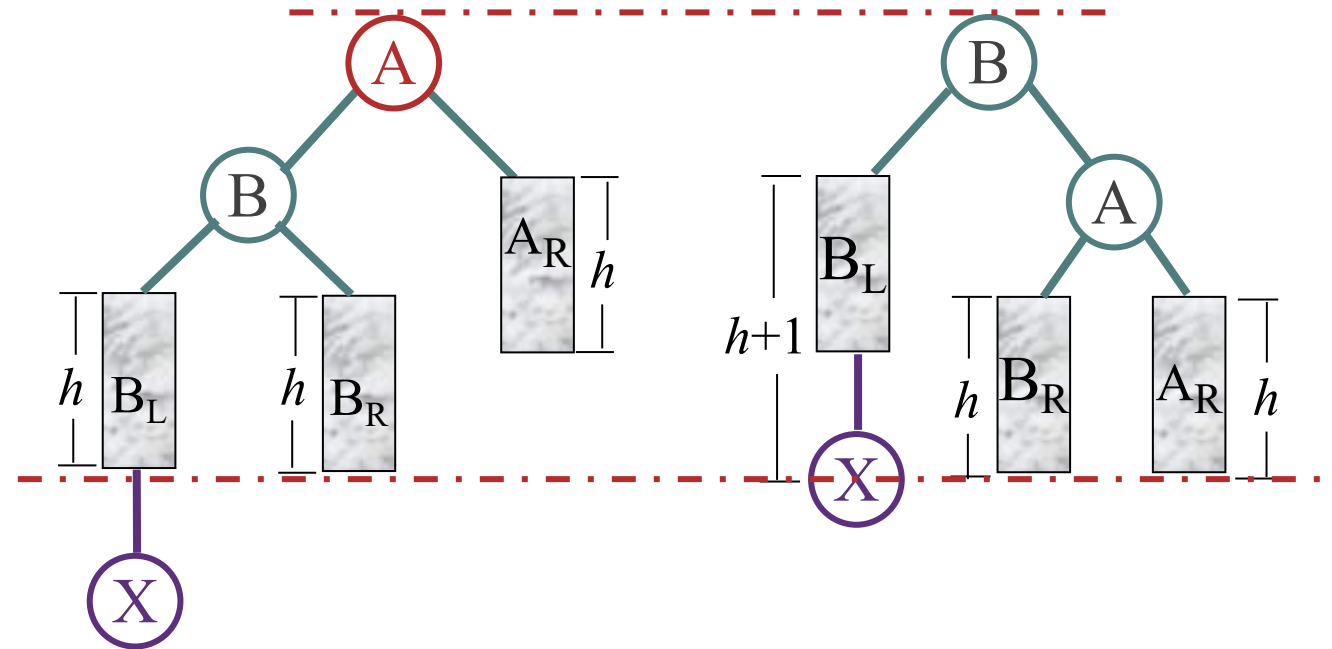
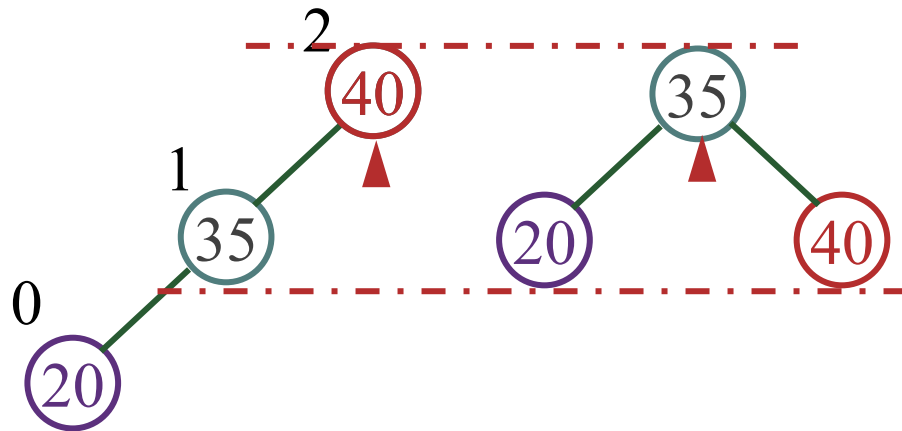


7.3 树表的查找技术

7-3-2 平衡二叉树

4. LL型平衡调整

例 1：设序列{40, 35, 20}，构造平衡二叉树



插入结点X失去平衡

调整后重新平衡



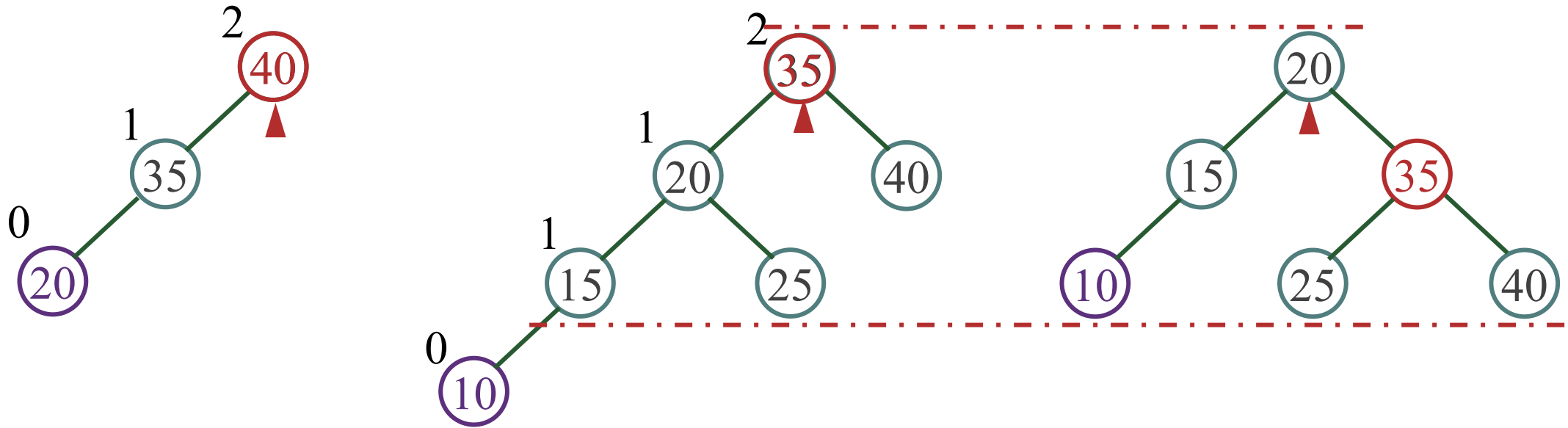
新插入结点20和最小不平衡子树根结点40之间的关系——**LL型**

7.3 树表的查找技术

7-3-2 平衡二叉树

4. LL型平衡调整

例 2：设序列{40, 35, 20, 15, 25, 10}，构造平衡二叉树



✈ 新插入结点10和最小不平衡子树根结点35之间的关系——LL型

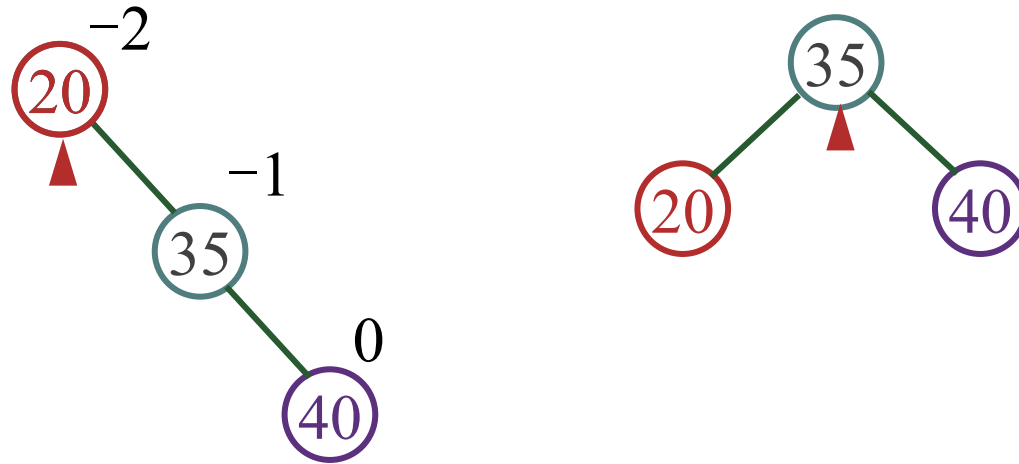
✈ 扁担原理：将根结点看成是扁担中肩膀的位置

7.3 树表的查找技术

7-3-2 平衡二叉树

5. RR型平衡调整

例 3：设序列{20, 35, 40}，构造平衡二叉树



✈ 新插入结点40和最小不平衡子树根结点20之间的关系——RR型

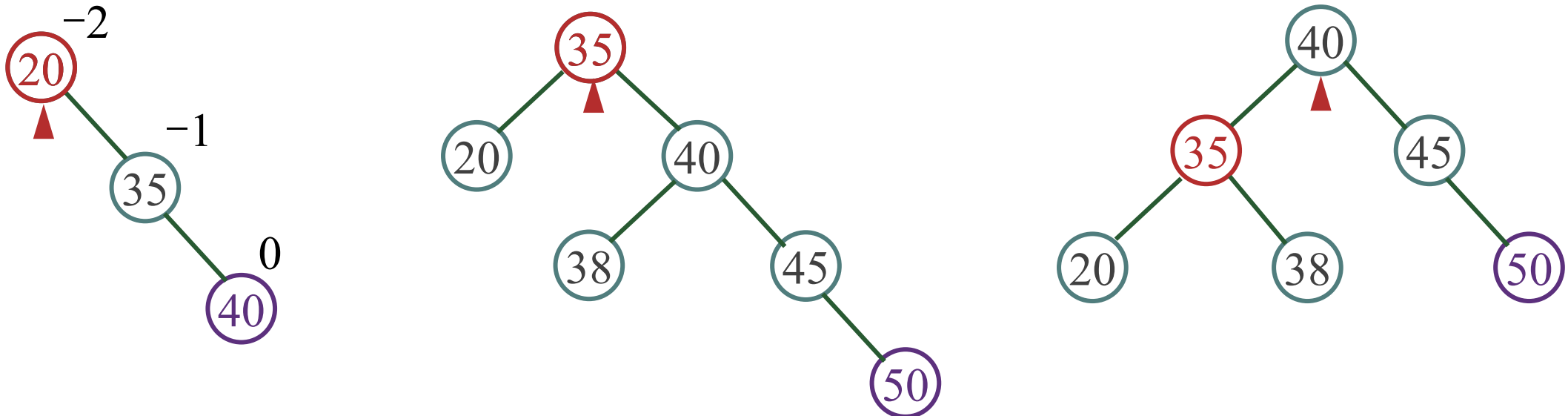
✈ 扁担原理：将根结点看成是扁担中肩膀的位置

7.3 树表的查找技术

7-3-2 平衡二叉树

5. RR型平衡调整

例 4：设序列{20, 35, 40, 38, 45, 50}，构造平衡二叉树



➤ 新插入结点50和最小不平衡子树根结点35之间的关系——RR型

➤ 扁担原理：将根结点看成是扁担中肩膀的位置

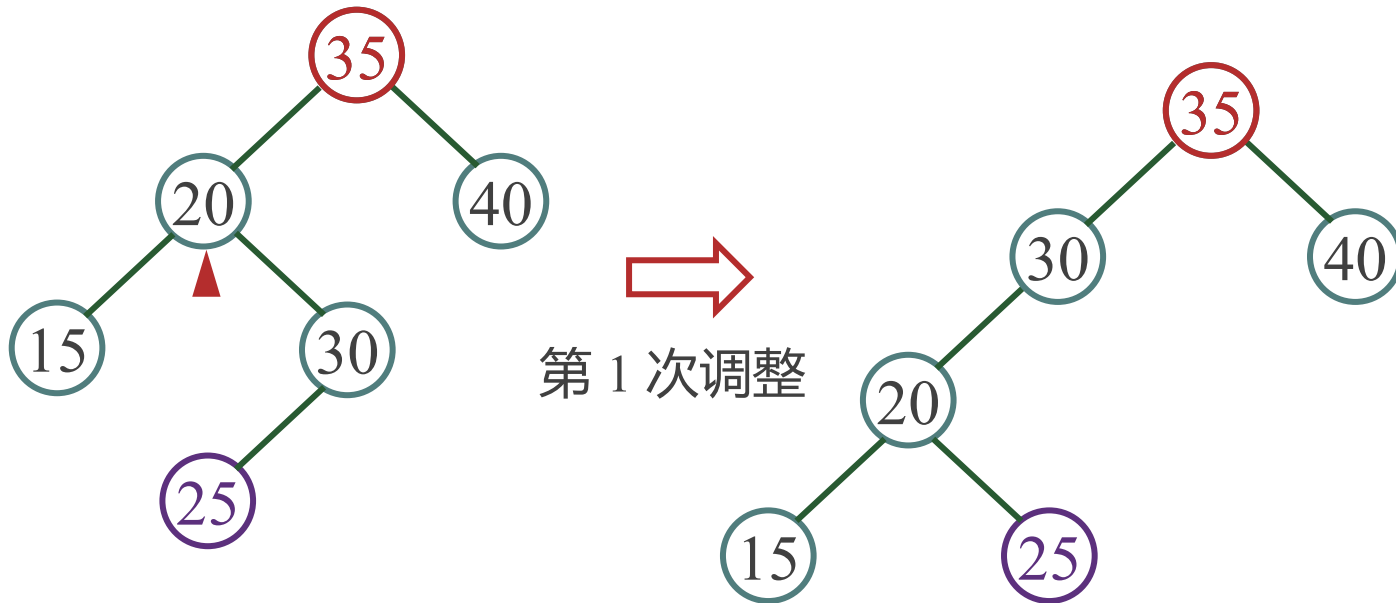
➤ 旋转优先：旋转下来的结点作为新根结点的孩子

7.3 树表的查找技术

7-3-2 平衡二叉树

6. LR型平衡调整

例 5：设序列{35, 40, 20, 15, 30, 25}，构造平衡二叉树



✦ 新插入结点25和最小不平衡子树根结点35之间的关系——LR型

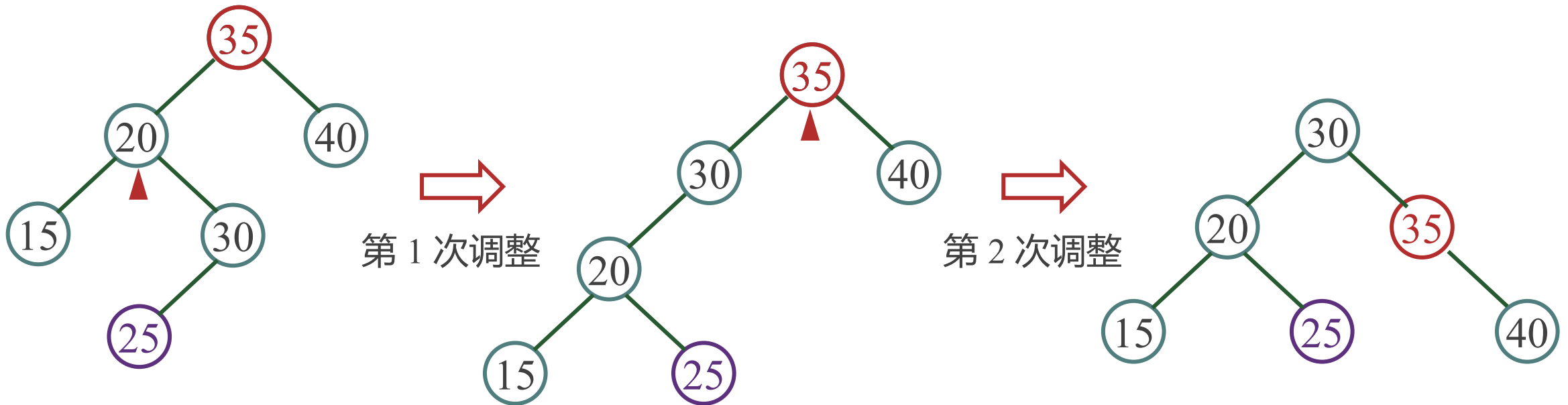
✦ 扁担原理：将根结点看成是扁担中肩膀的位置

7.3 树表的查找技术

7-3-2 平衡二叉树

6. LR型平衡调整

例 5：设序列{35, 40, 20, 15, 30, 25}，构造平衡二叉树



✦ 新插入结点25和最小不平衡子树根结点35之间的关系——LR型

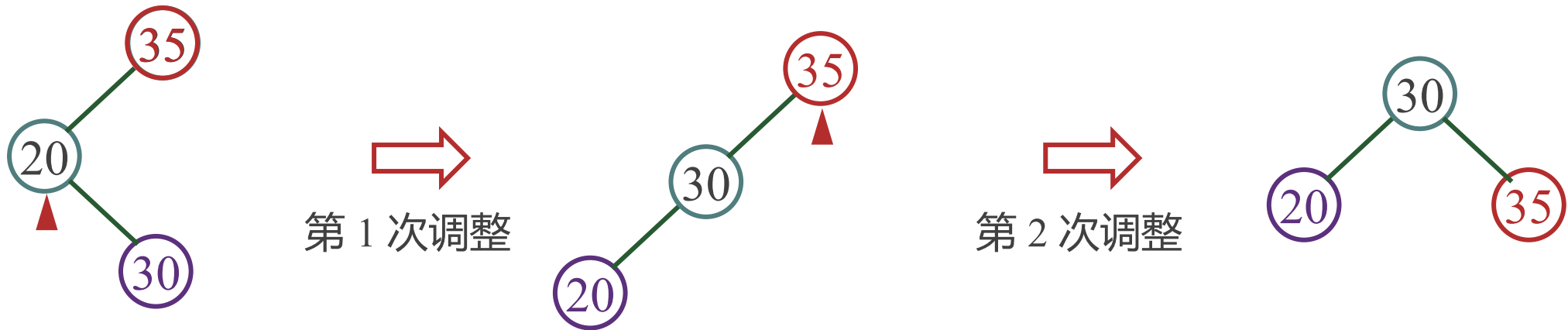
✦ 扁担原理：将根结点看成是扁担中肩膀的位置

7.3 树表的查找技术

7-3-2 平衡二叉树

6. LR型平衡调整

例 6：设序列{35, 20, 30}，构造平衡二叉树



新插入结点30和最小不平衡子树根结点35之间的关系——LR型



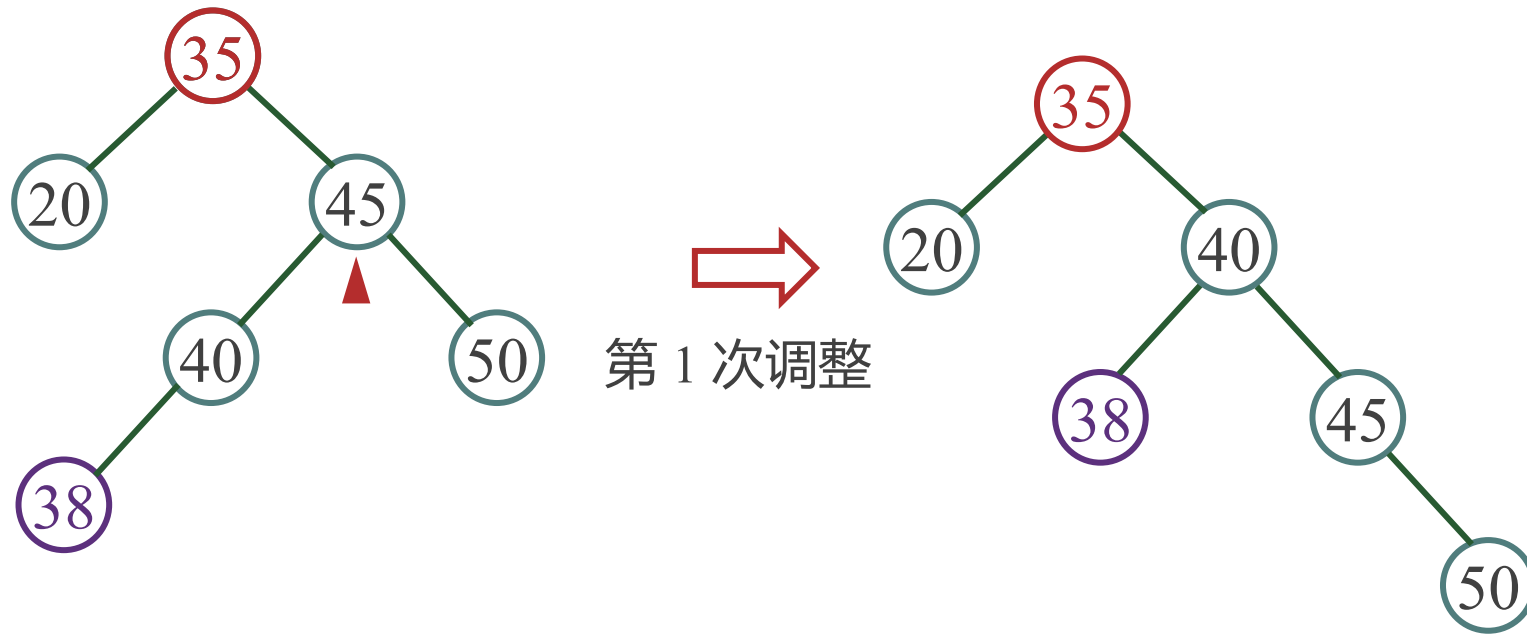
扁担原理：将根结点看成是扁担中肩膀的位置

7.3 树表的查找技术

7-3-2 平衡二叉树

7. RL型平衡调整

例 7：设序列{35, 45, 20, 50, 40, 38}，构造平衡二叉树



✈ 新插入结点38和最小不平衡子树根结点35之间的关系——RL型

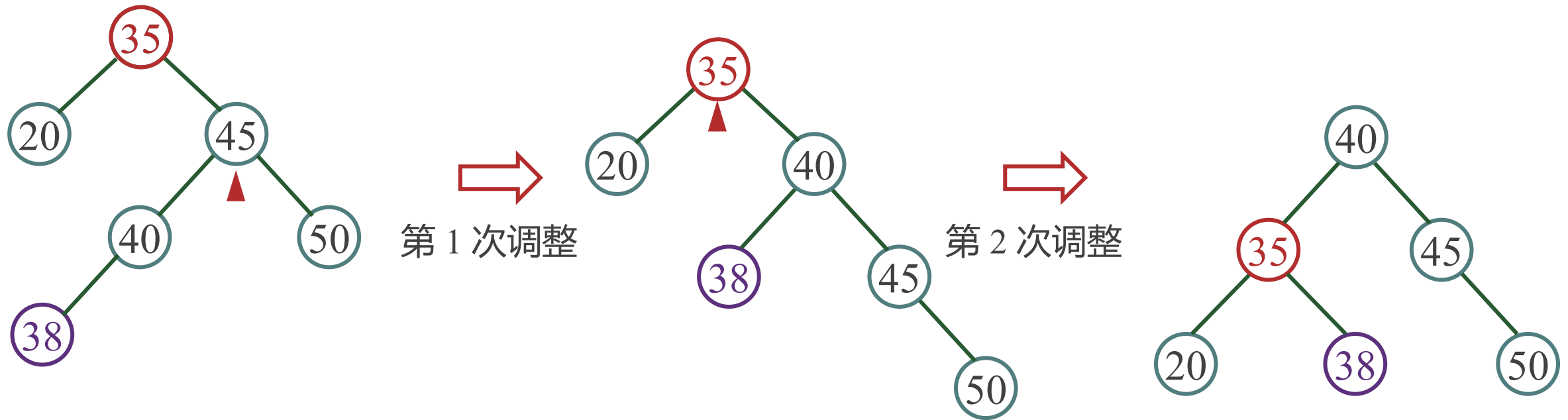
✈ 扁担原理：将根结点看成是扁担中肩膀的位置

7.3 树表的查找技术

7-3-2 平衡二叉树

7. RL型平衡调整

例 7：设序列{35, 45, 20, 50, 40, 38}，构造平衡二叉树



✈ 新插入结点38和最小不平衡子树根结点35之间的关系——RL型

✈ 扁担原理：将根结点看成是扁担中肩膀的位置

7.3 树表的查找技术

7-3-2 平衡二叉树

8. 平衡二叉树的性能

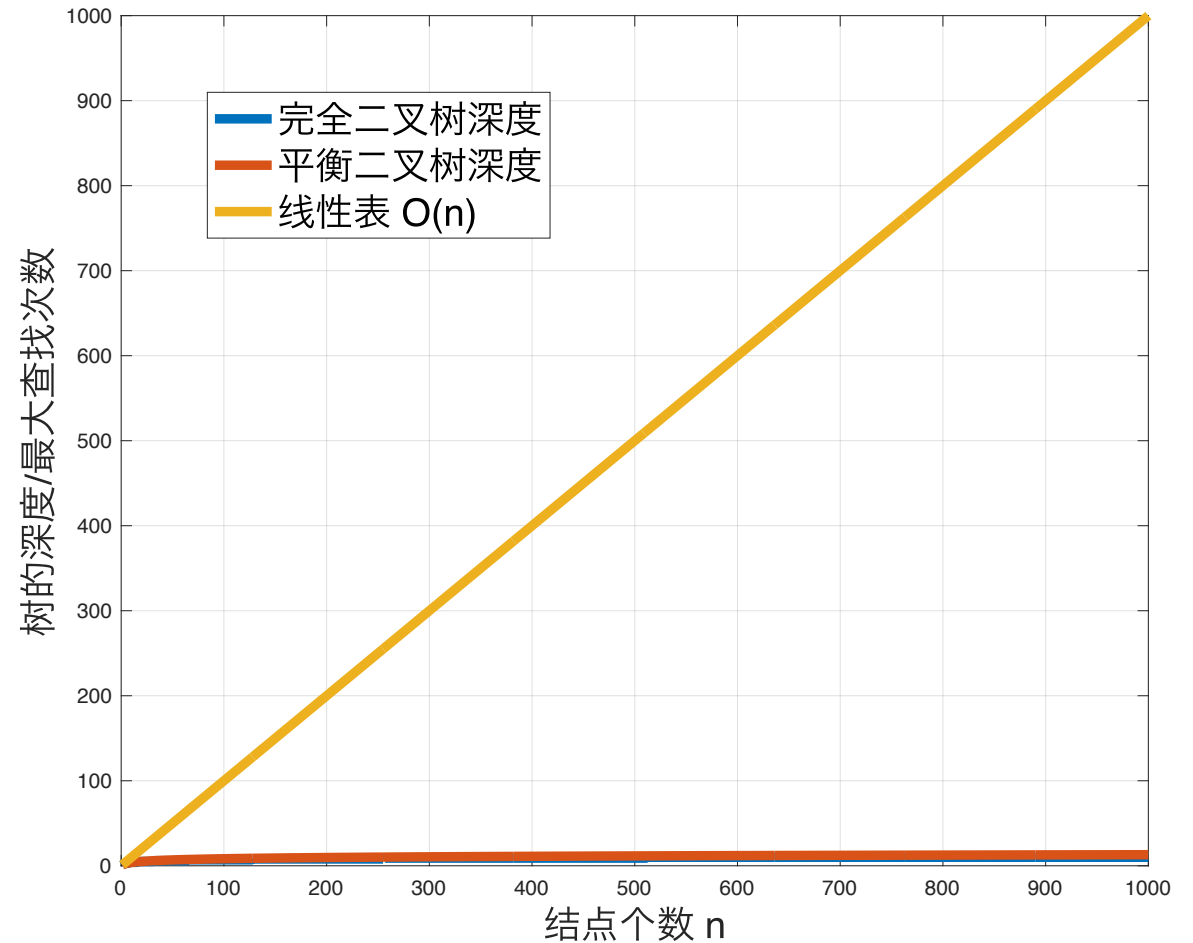
平衡二叉树的深度: $1.44 \log_2(n+2) - 1.328$

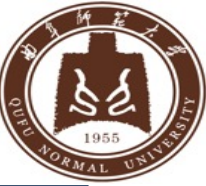
左右子树的深度相差 1

左右子树的深度相同

完全二叉树的深度: $\lfloor \log_2 n \rfloor + 1$

你知道 $O(n)$ 在哪里吗?





7.3 树表的查找技术

7-3-2 平衡二叉树

8. 平衡二叉树的性能

树高的下界:

假设树高为 h , 最大点数为: $n=2^{h+1}-1$

导出: $h=\log_2(n+1)-1$ (满二叉树)

树高的上界:

假设树高为 h , 高度一定时, 左右子树高低都差1时树的结点数最少。

总点数为: $n(h) = n(h-1) + n(h-2) + 1$

$$n(0) = 1, \quad n(1) = 2$$

$$n(h) + 1 = n(h-1) + 1 + n(h-2) + 1$$

$$n(0) = 1, \quad n(1) = 2$$

斐波那契数列通项公式:

$$n(h) = f(h+3) - 1$$

$$\begin{cases} f(n) = f(n-1) + f(n-2), & n \geq 3 \\ f(1) = f(2) = 1 \end{cases}$$

斐波那契数列通项公式

$$f(n) = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

$$n(h) = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{h+3} - \left(\frac{1-\sqrt{5}}{2} \right)^{h+3} \right] - 1 > \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{h+3} - 2$$
$$\frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^{h+3} < 1$$

$$h < \left[\log_{\frac{1+\sqrt{5}}{2}} \sqrt{5}(n(h) + 2) \right] - 3$$

$$h < \frac{\log_2(n(h) + 2)}{\log_2\left(\frac{1+\sqrt{5}}{2}\right)} + \frac{\log_2 \sqrt{5}}{\log_2\left(\frac{1+\sqrt{5}}{2}\right)} - 3$$

平衡二叉树的深度上界: $1.44 \log_2(n+2) - 1.328$

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

**1. 设查找的关键字序列为(35,26,6,96,75,12,46,58,32),
请构造出对应的二叉排序树和平衡二叉树。**

实验八 查找算法的实现与应用

一、实验目的

1. 掌握二叉排序树的逻辑结构和存储结构
2. 掌握二叉排序树**构建**原理及实现方法
3. 掌握二叉排序树**查找并插入结点**的原理及实现方法
4. 掌握二叉排序树**查找并删除结点**的原理及实现方法
5. 用C++语言实现相关算法，并上机调试。

二、实验内容

1. 建立二叉排序树类。
2. 实现二叉排序树的建立、查找。
3. 实现二叉排序树的插入、删除。
4. 给出测试过程和测试结果。

实验时间： 第16周周四晚

22网安： 18:30-20:10 **22物联网：** 20:10-21:50

实验地点： 软件基础实验室301（老干部处）

实验报告要求： 测试数据不低于10个，每插入一个结点，绘制树的形状。

12月16日前提交预习报告，12月25日前提交正式报告。



Thank You !

Q & A

7.3 树表的查找技术

7-3-2 平衡二叉树

8. 平衡二叉树的性能

平衡二叉树的深度: $1.44 \log_2(n+2) - 1.328$

左右子树的深度相差 1

左右子树的深度相同

完全二叉树的深度: $\lfloor \log_2 n \rfloor + 1$

你知道 $O(n)$ 在哪里吗?

