本节内容

平衡二叉树

删除操作

平衡二叉树的插入&删除

平衡二叉树的**插入**操作：

- 插入新结点后，要保持二叉排序树的特性不变（左<中<右）
- 若插入新结点导致不平衡，则需要调整平衡

平衡二叉树的**删除**操作：

- 删除结点后，要保持二叉排序树的特性不变（左<中<右）
- 若删除结点导致不平衡，则需要调整平衡

平衡二叉树的删除

平衡二叉树的**删除**操作：

- 删除结点后，要保持二叉排序树的特性不变（左<中<右）
- 若删除结点导致不平衡，则需要调整平衡

平衡二叉树的**删除**操作具体步骤：

①删除结点（方法同“二叉排序树”）

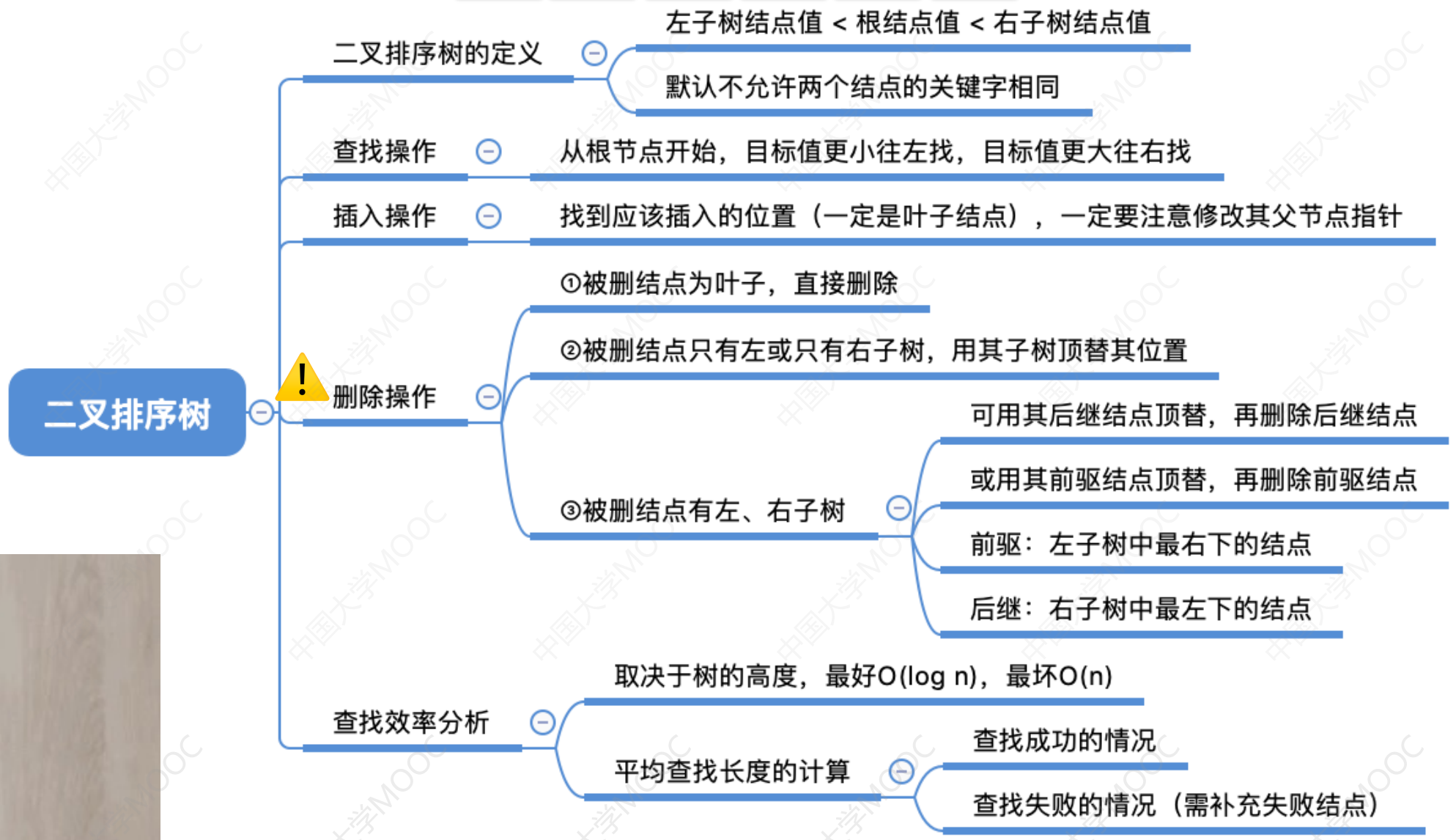
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

暂停偷看：二叉排序树的删除操作



AVL树删除操作——例1

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

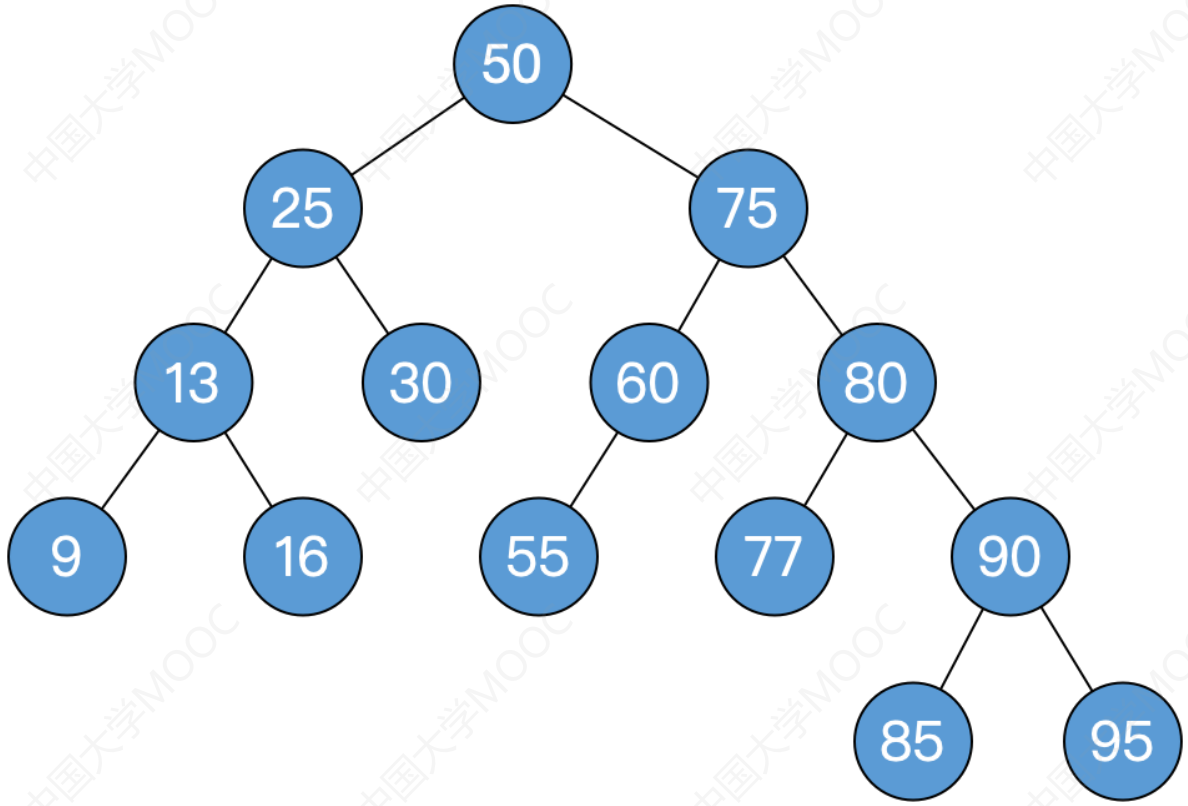
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例1

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

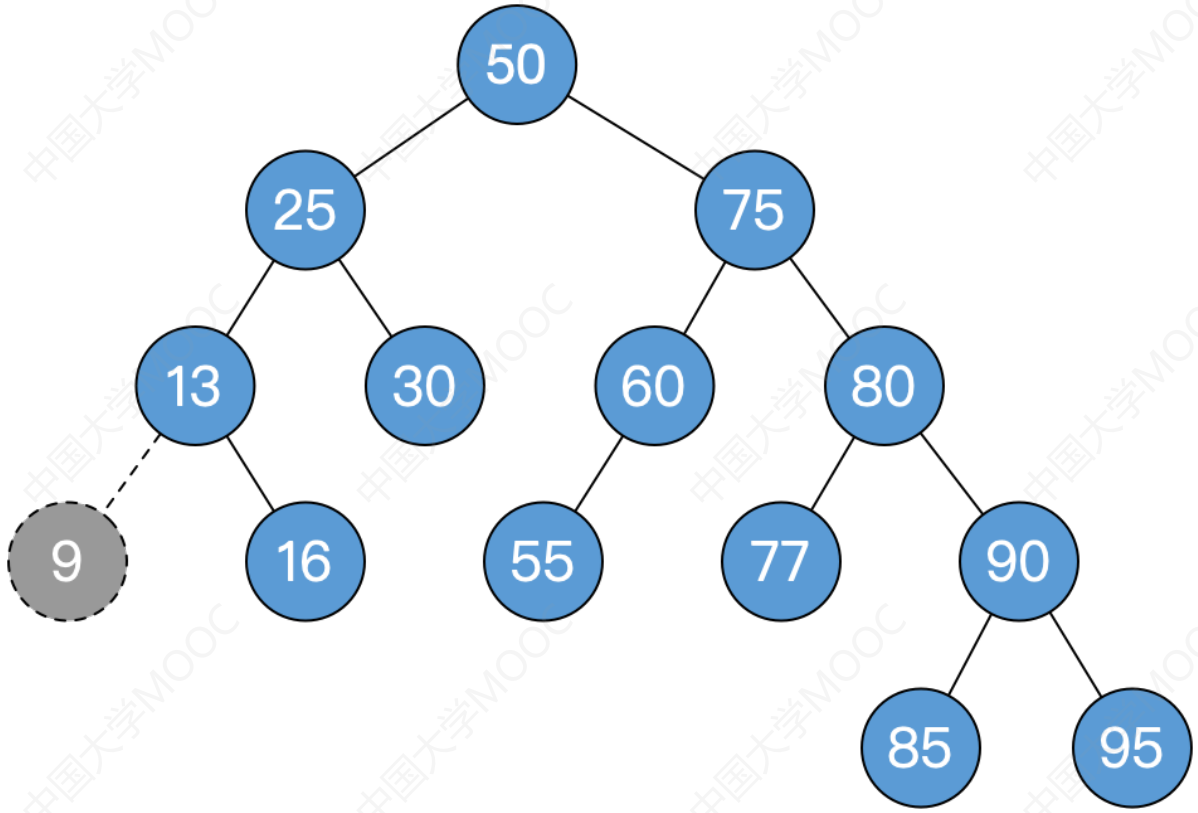
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例1

平衡二叉树的删除操作具体步骤：

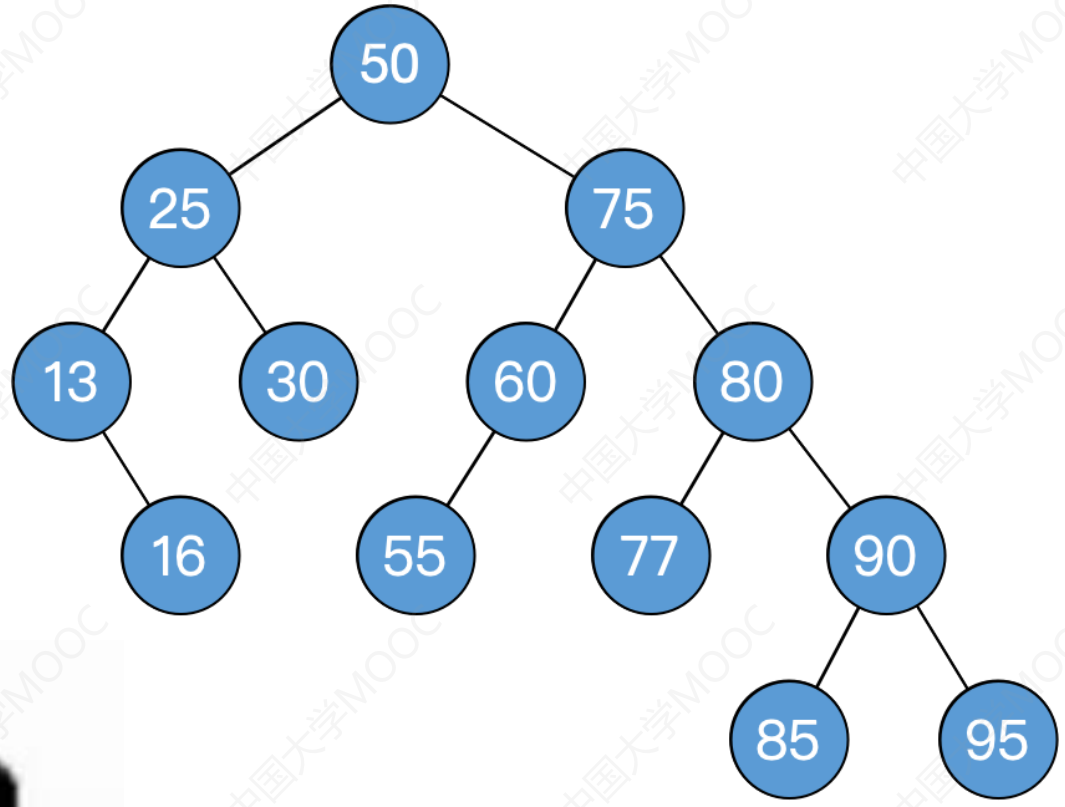
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例2

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

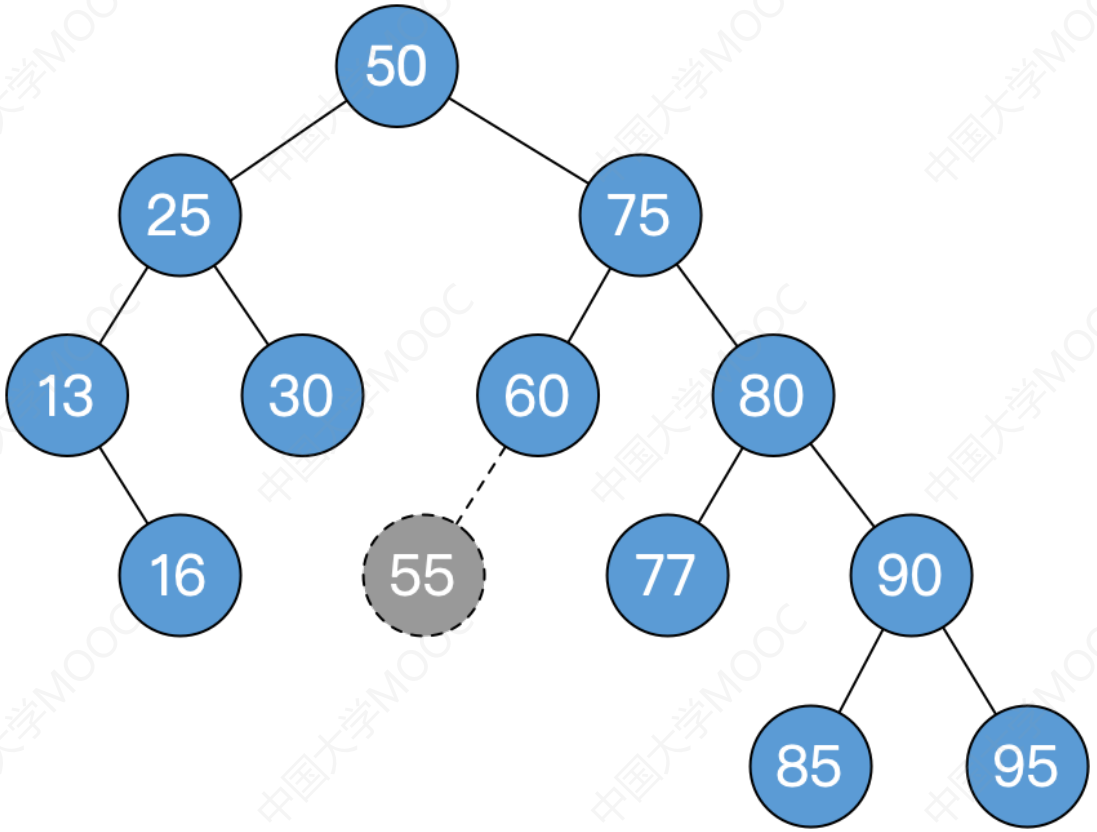
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例2

平衡二叉树的删除操作具体步骤：

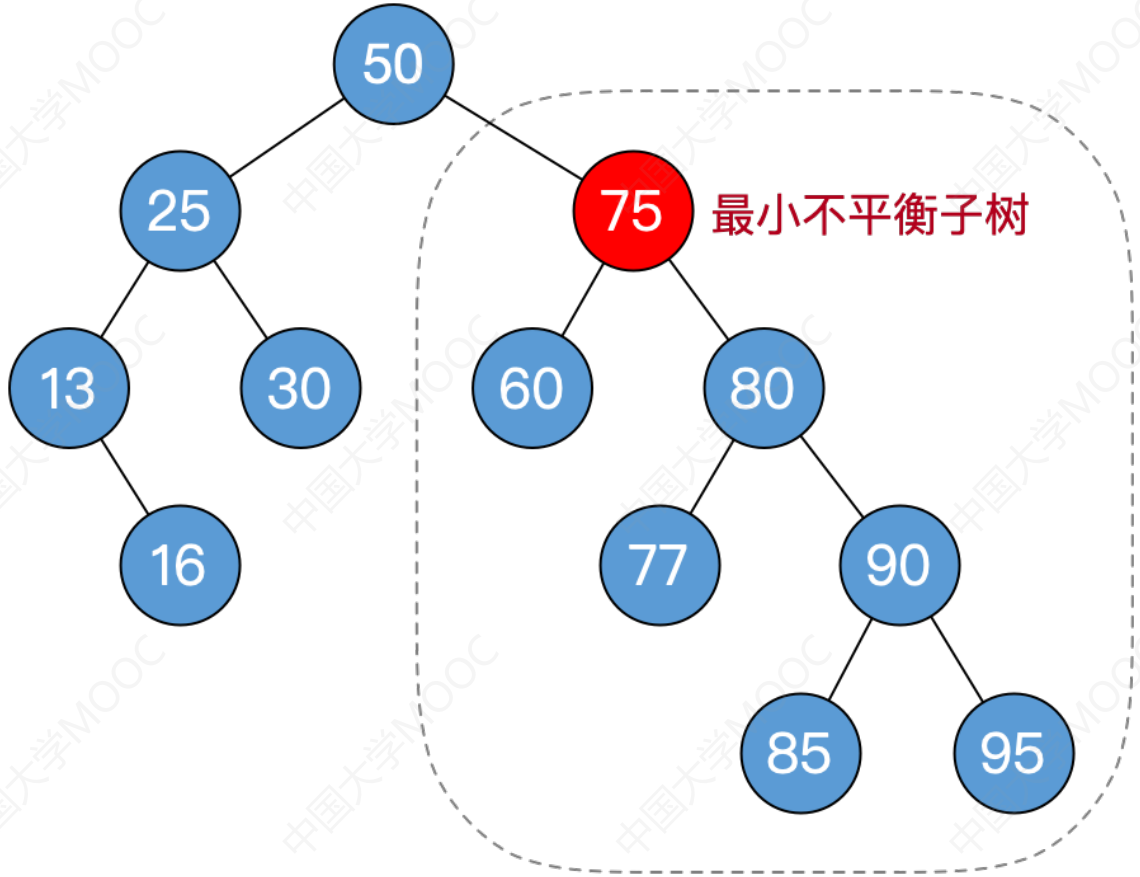
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例2

平衡二叉树的删除操作具体步骤：

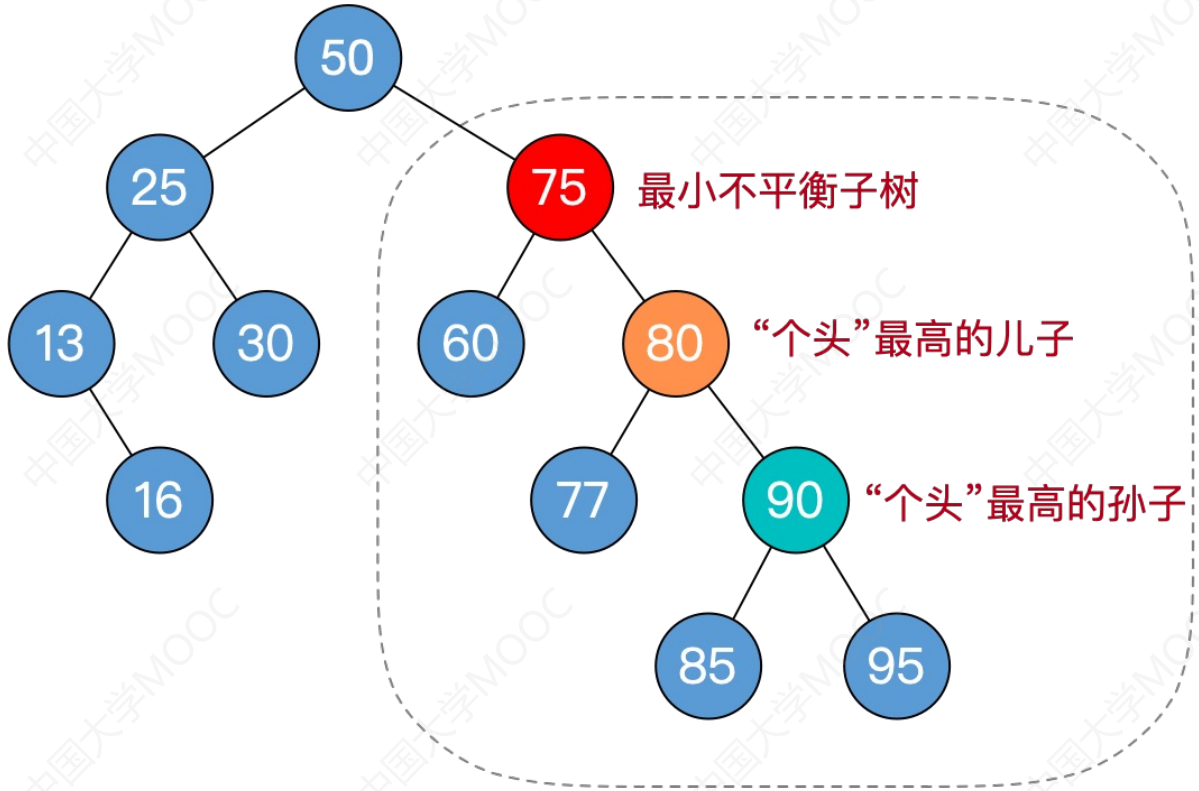
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例2

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

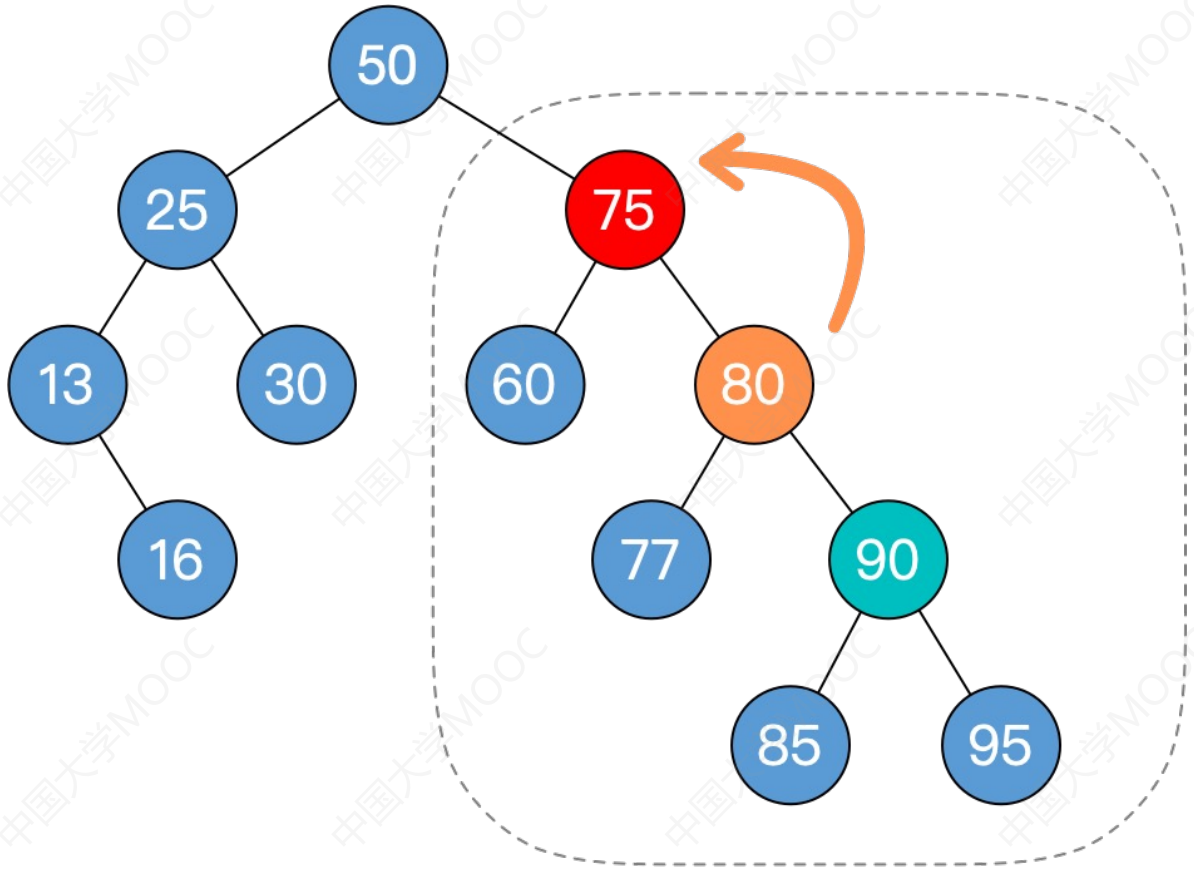
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



孙子在RR，儿子左单旋

AVL树删除操作——例2

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

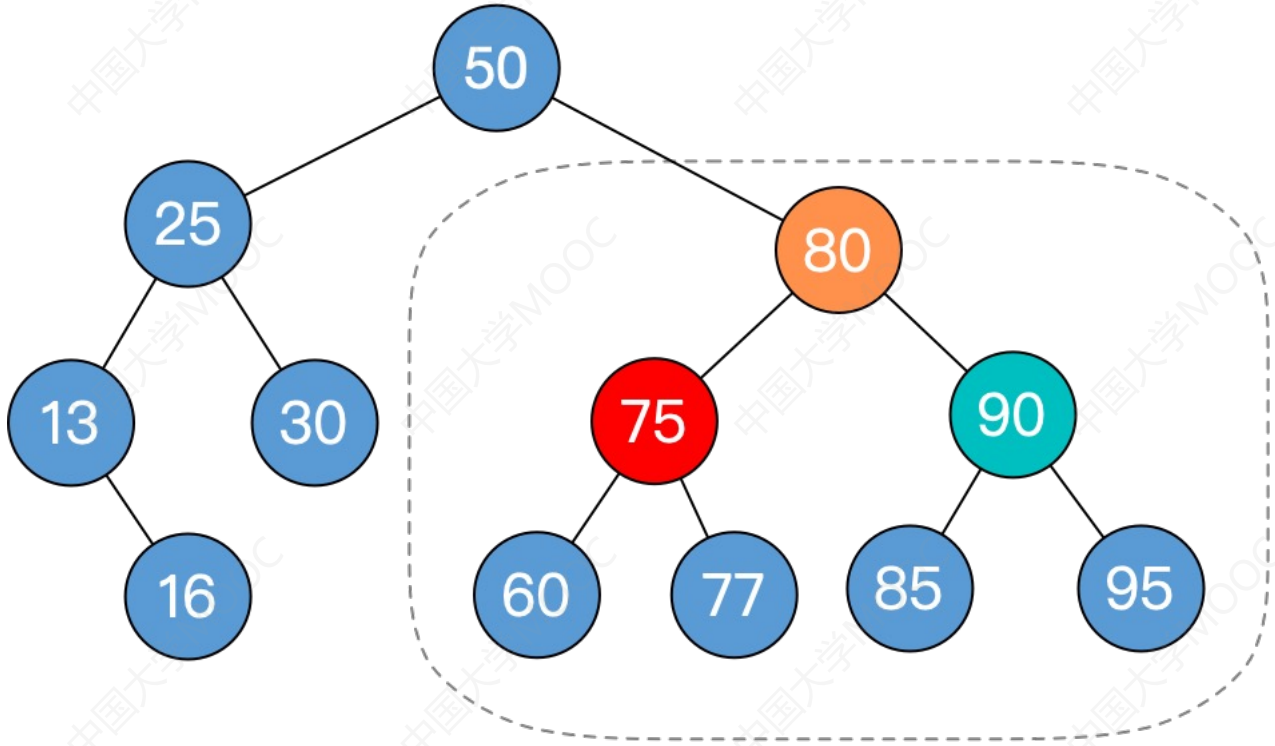
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



最小不平衡子树恢复平衡

AVL树删除操作——例2

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

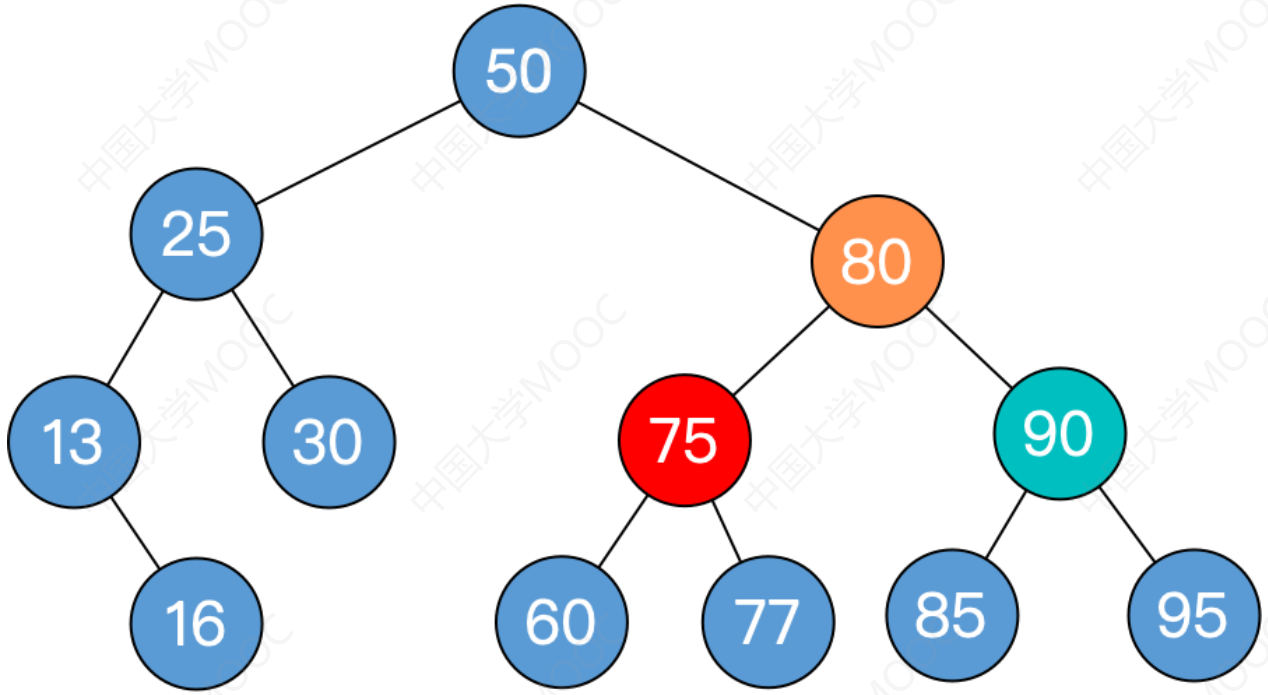
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）



AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

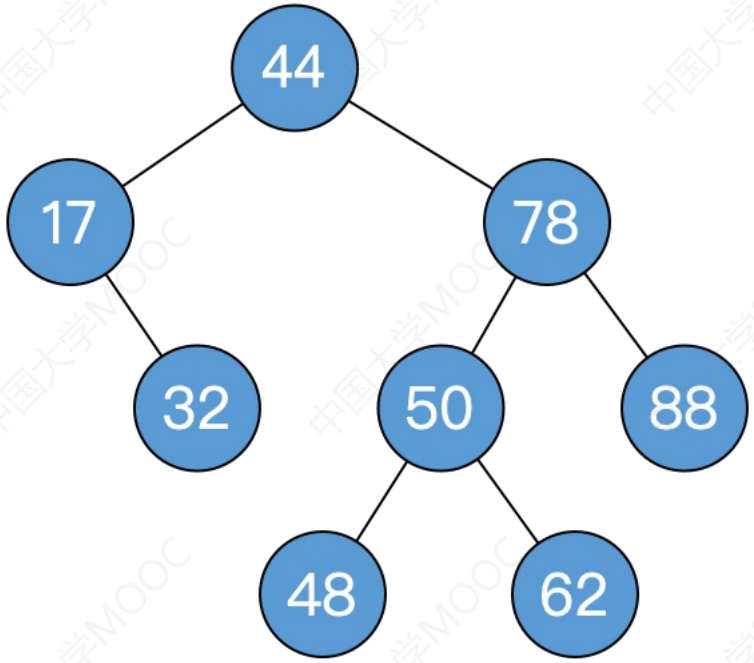
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

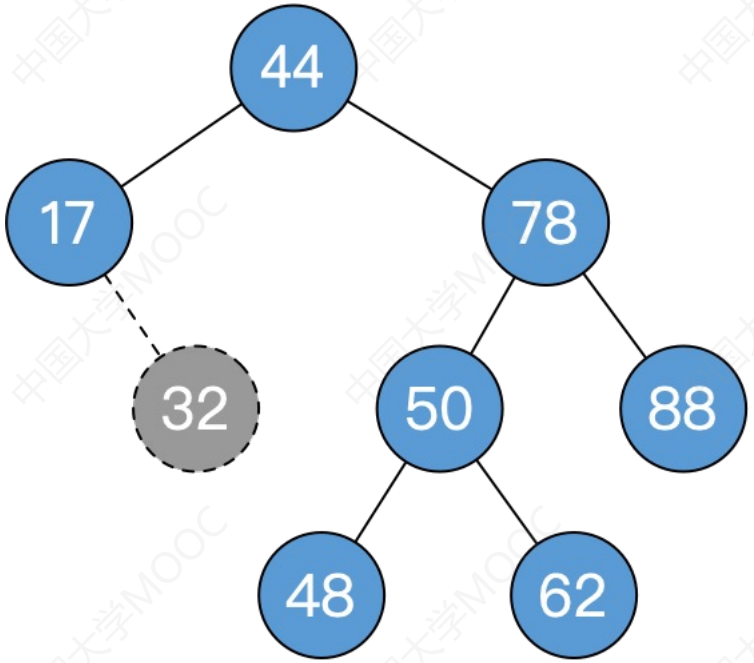
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

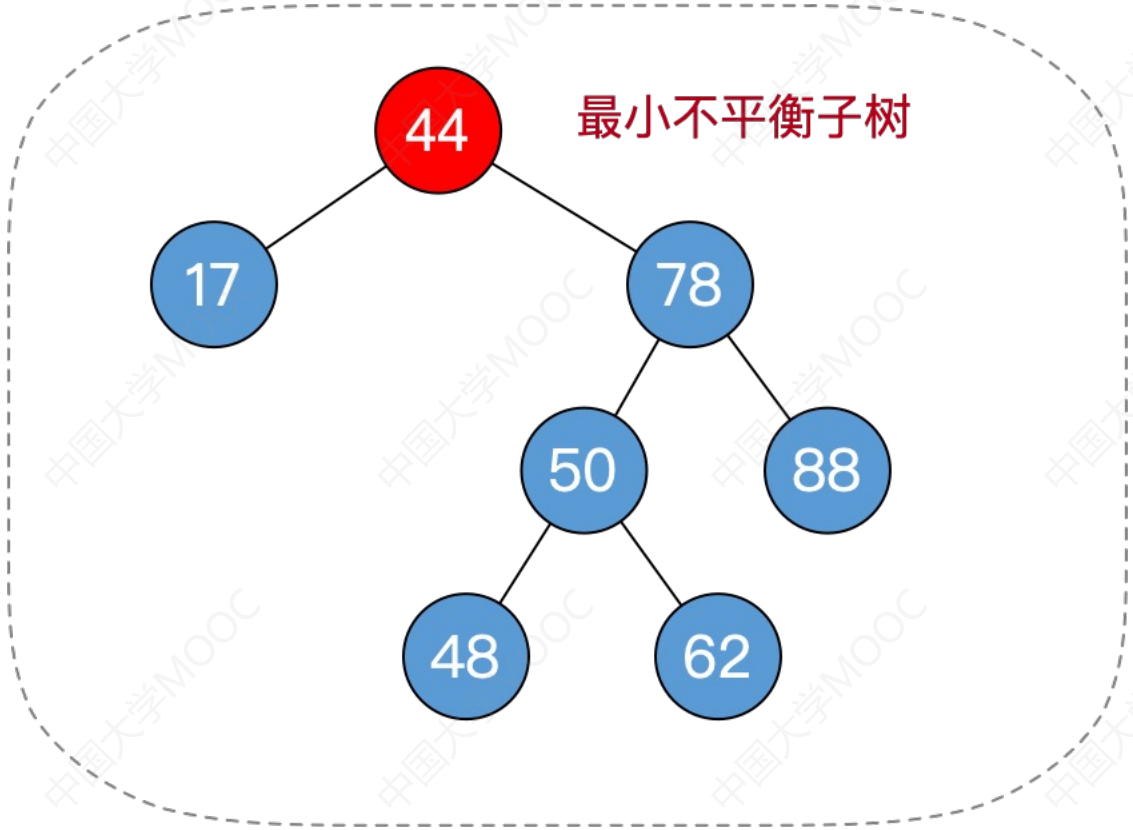
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

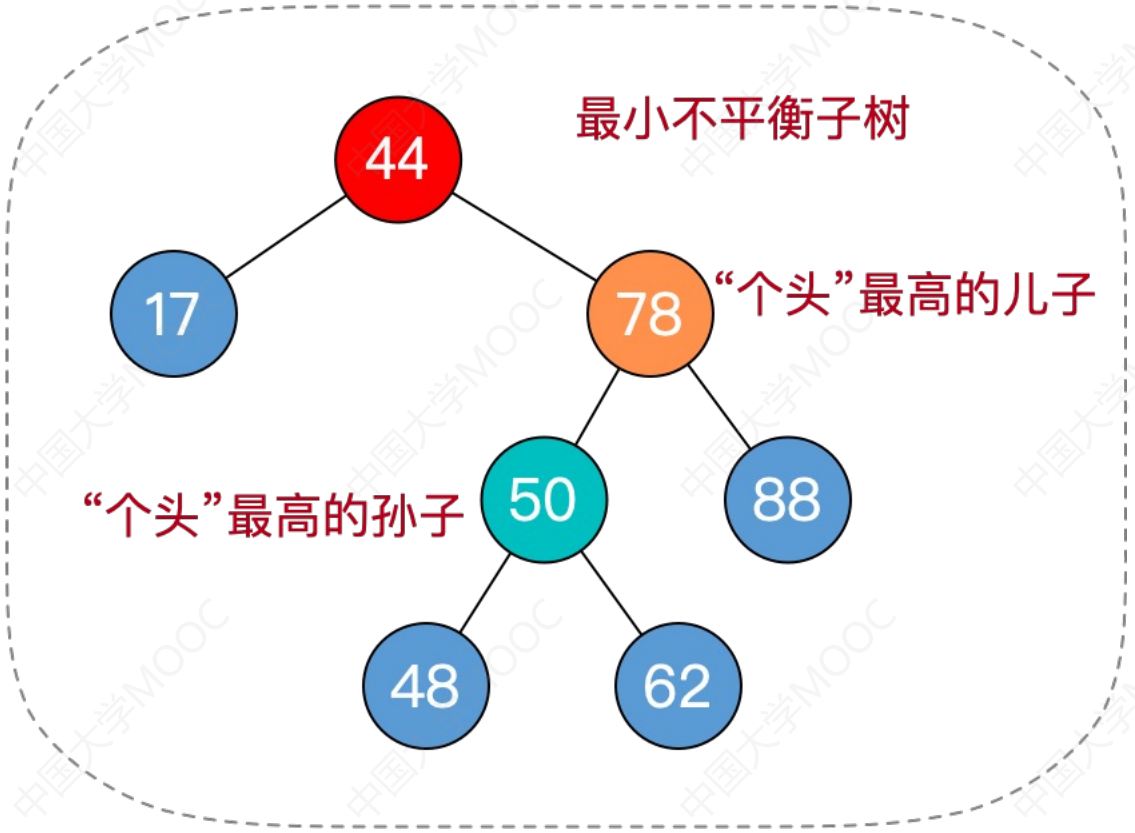
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

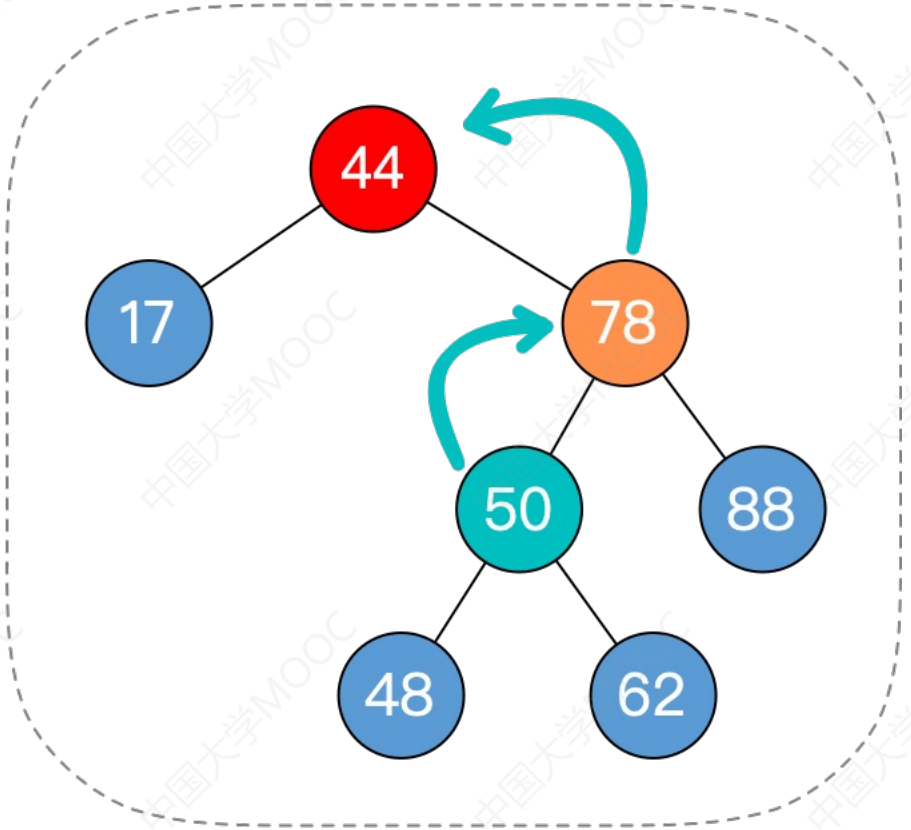
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



孙子在RL，孙子先右旋，再左旋

AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

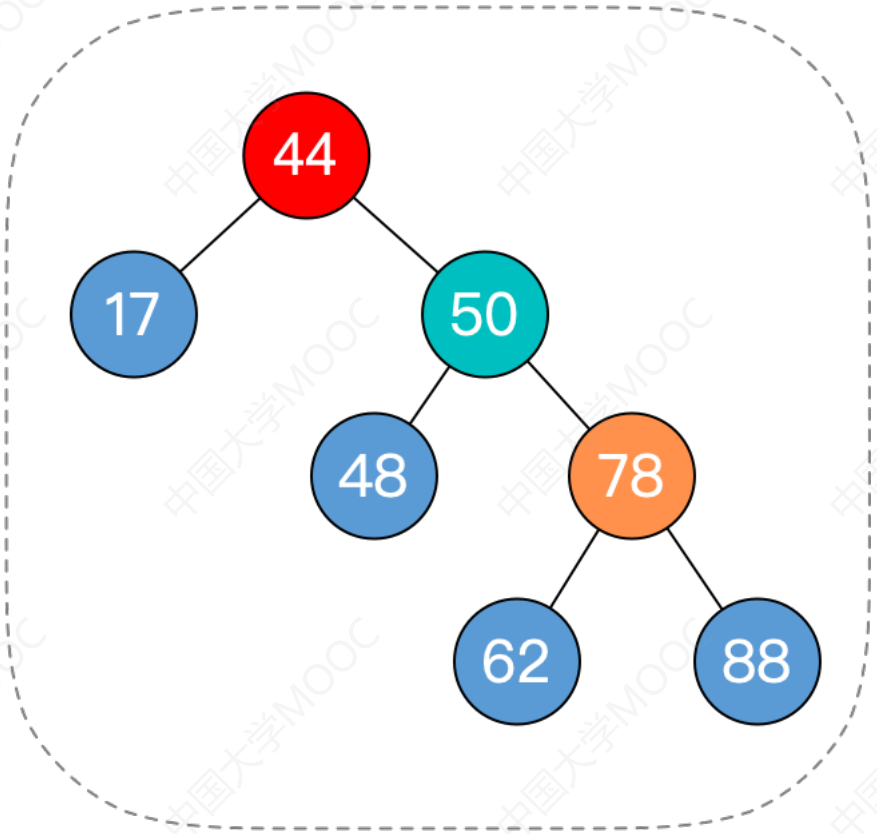
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



先右旋

AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

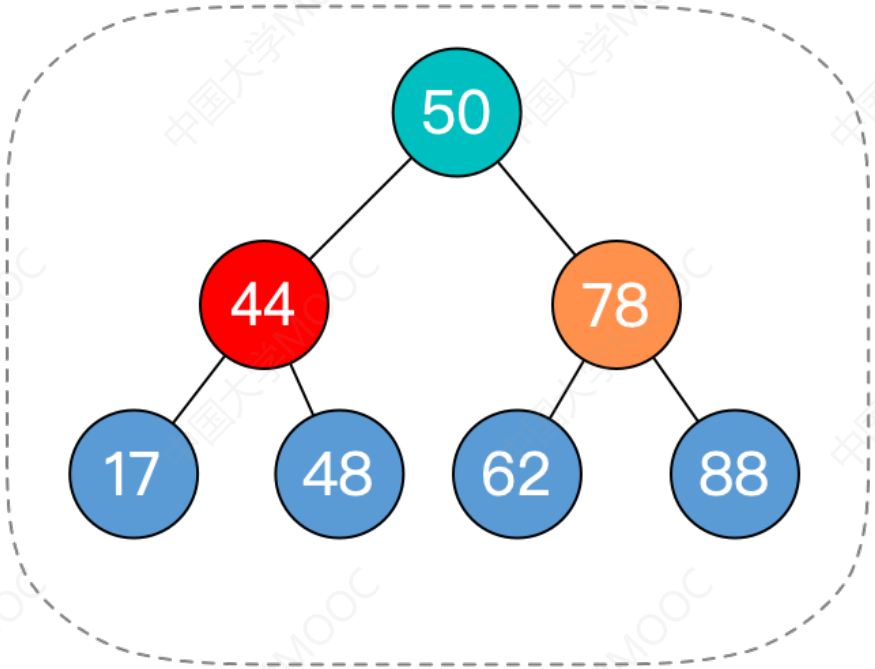
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



再左旋

最小不平衡子树恢复平衡

AVL树删除操作——例3

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

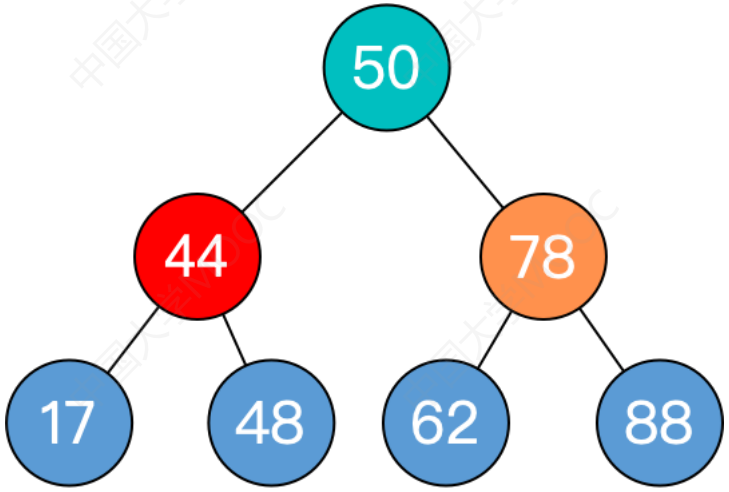
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）



AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

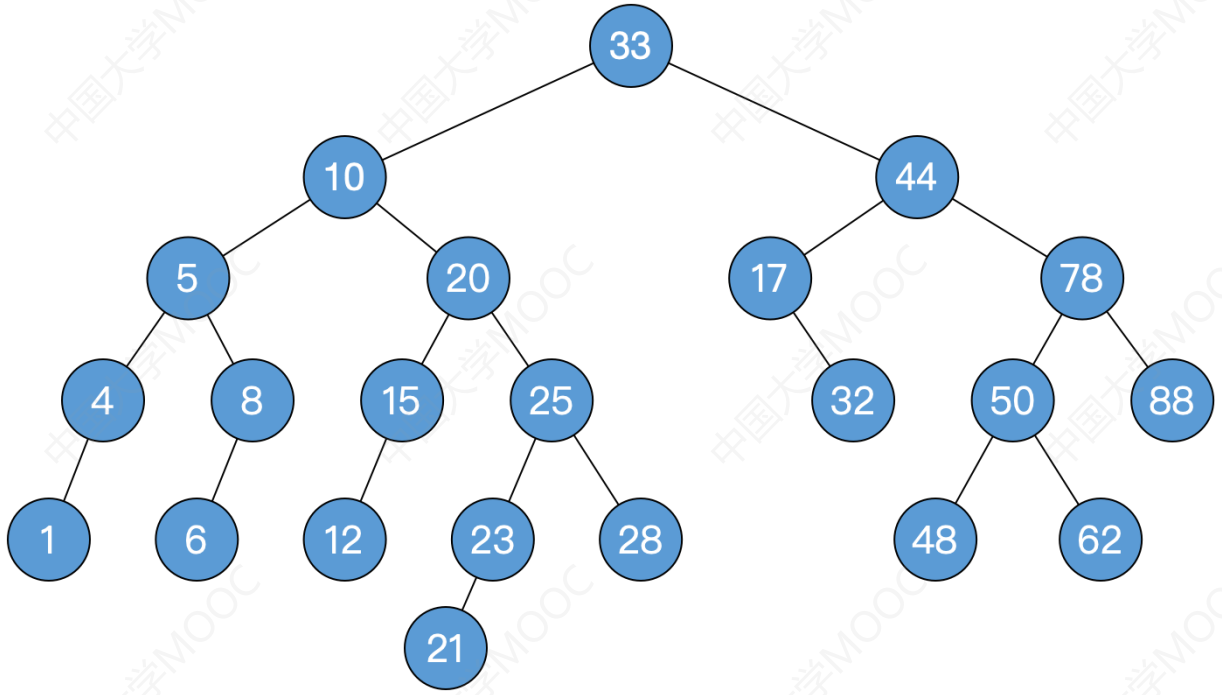
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

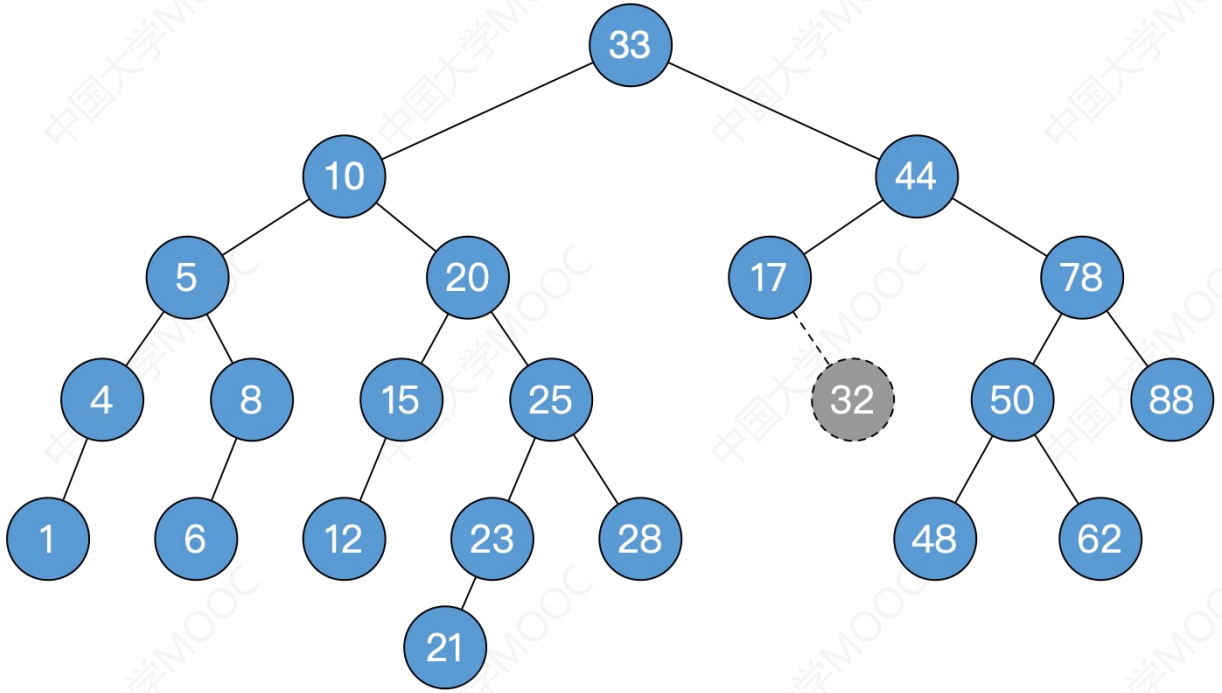
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

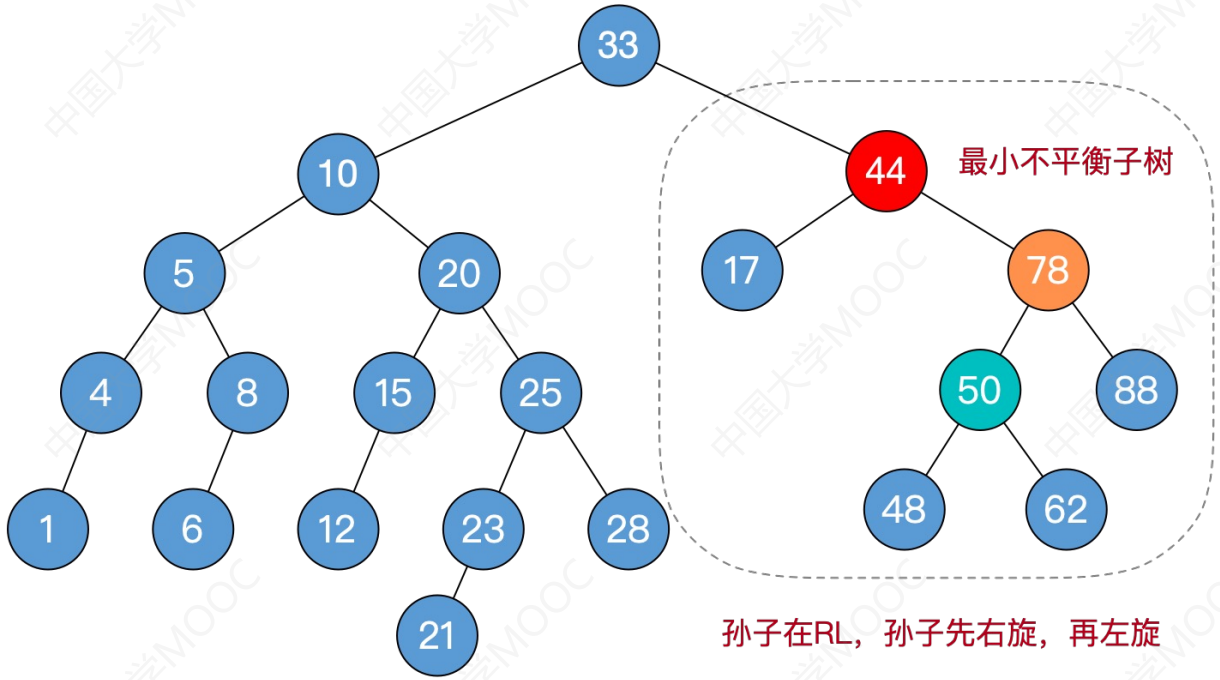
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

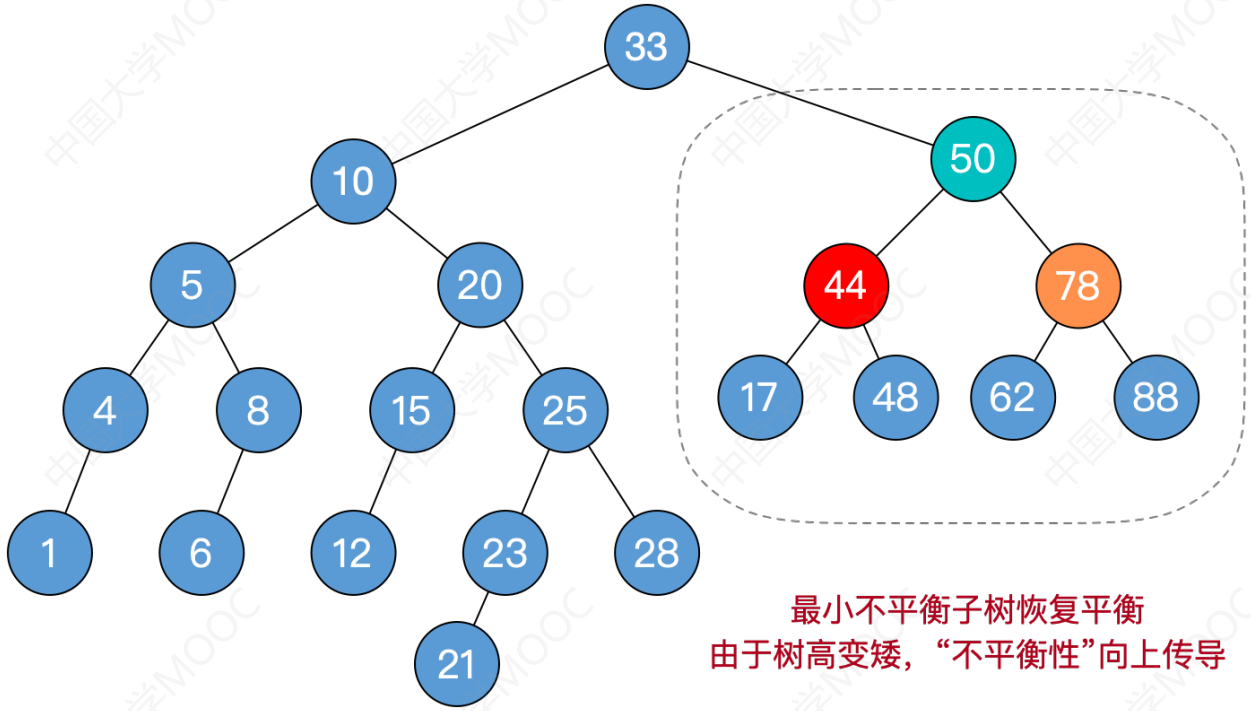
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）



最小不平衡子树恢复平衡
由于树高变矮，“不平衡性”向上传导

AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

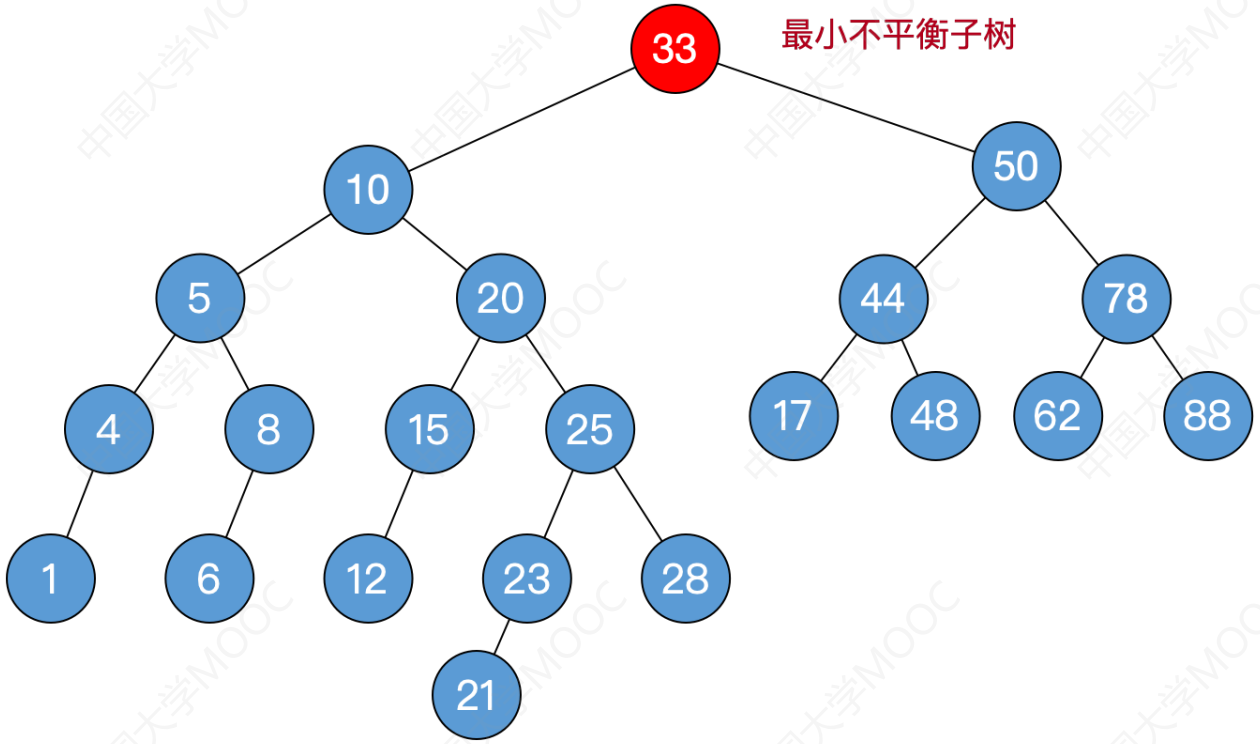
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

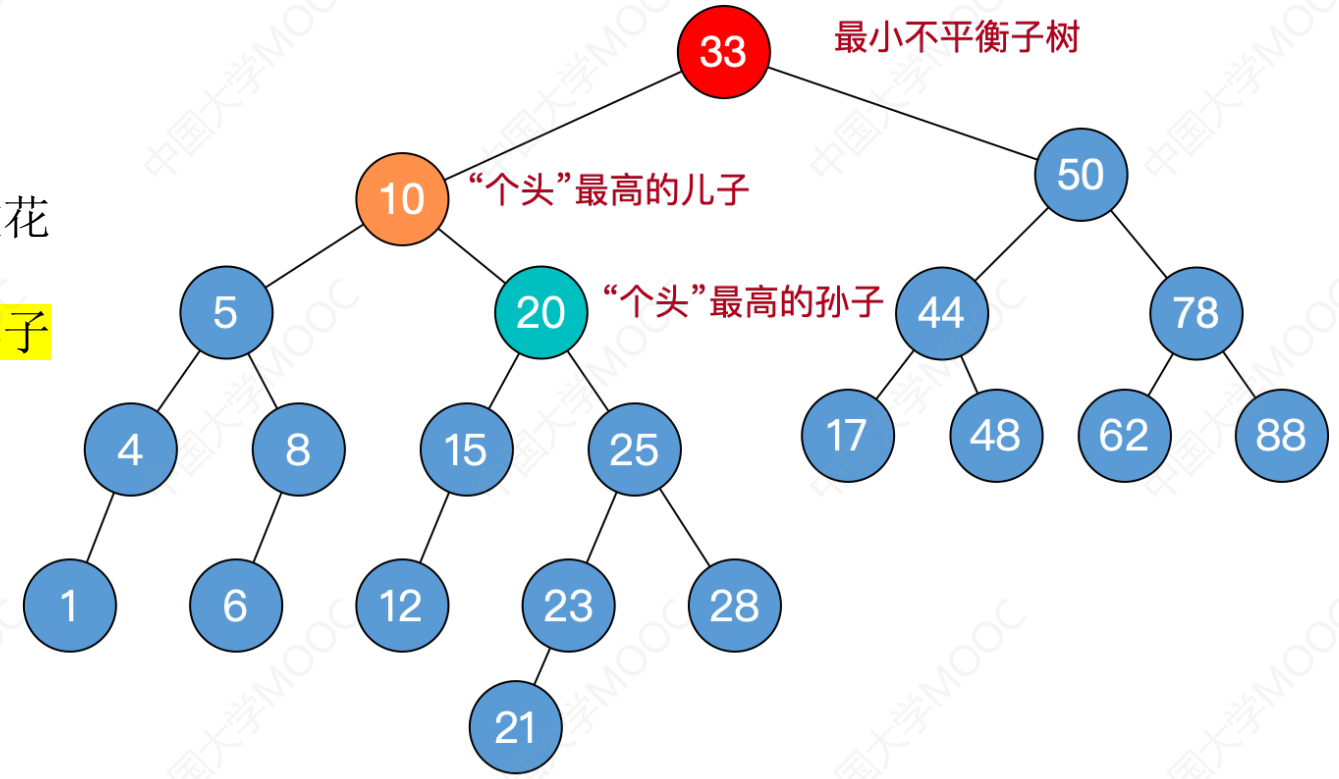
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



孙子在LR，孙子先左，再右旋

AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

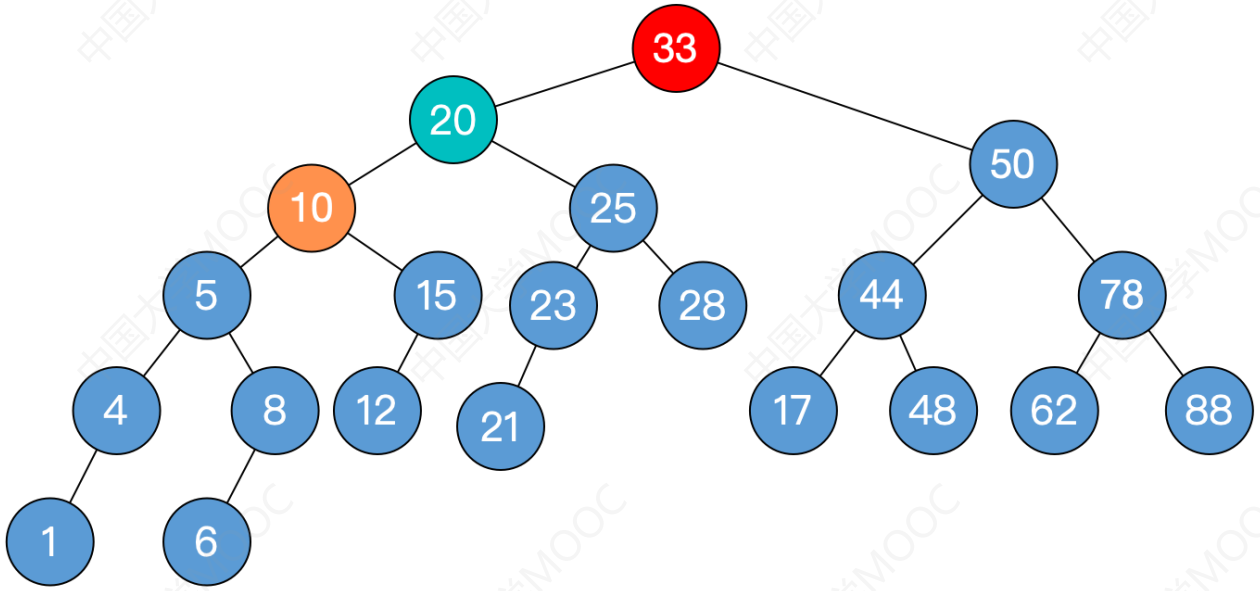
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



先左旋

AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

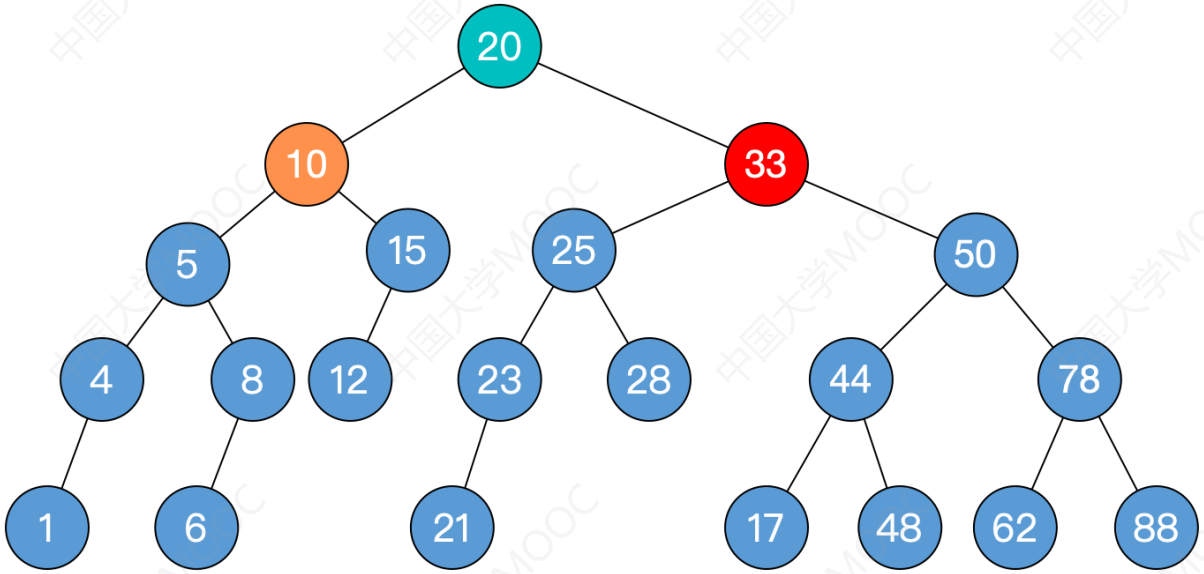
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



AVL树删除操作——例4

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

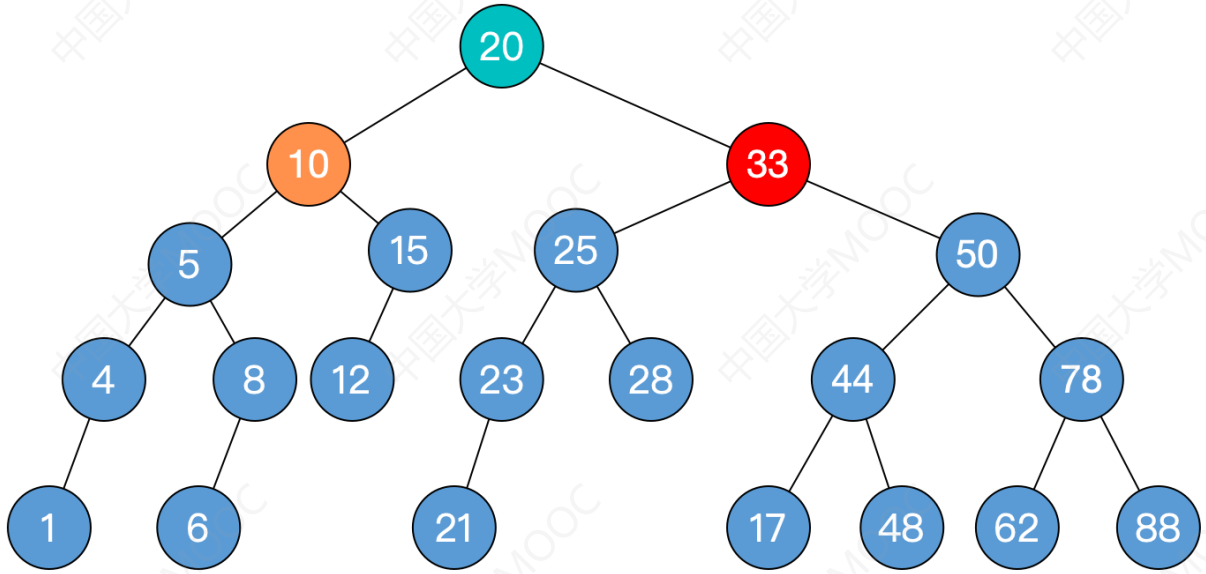
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）



AVL树删除操作——例5

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

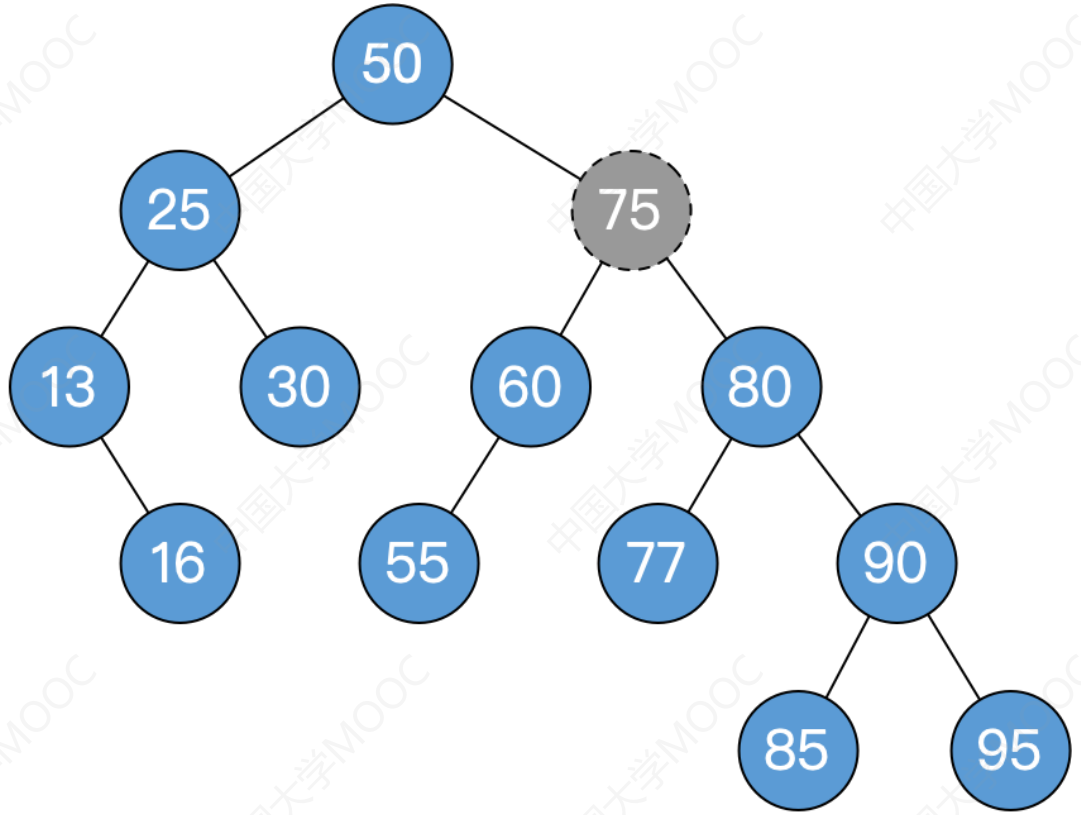
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



被删除结点有左右子树，用前驱结点顶替（复制数据即可）
并转化为对前驱结点的删除

AVL树删除操作——例5

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

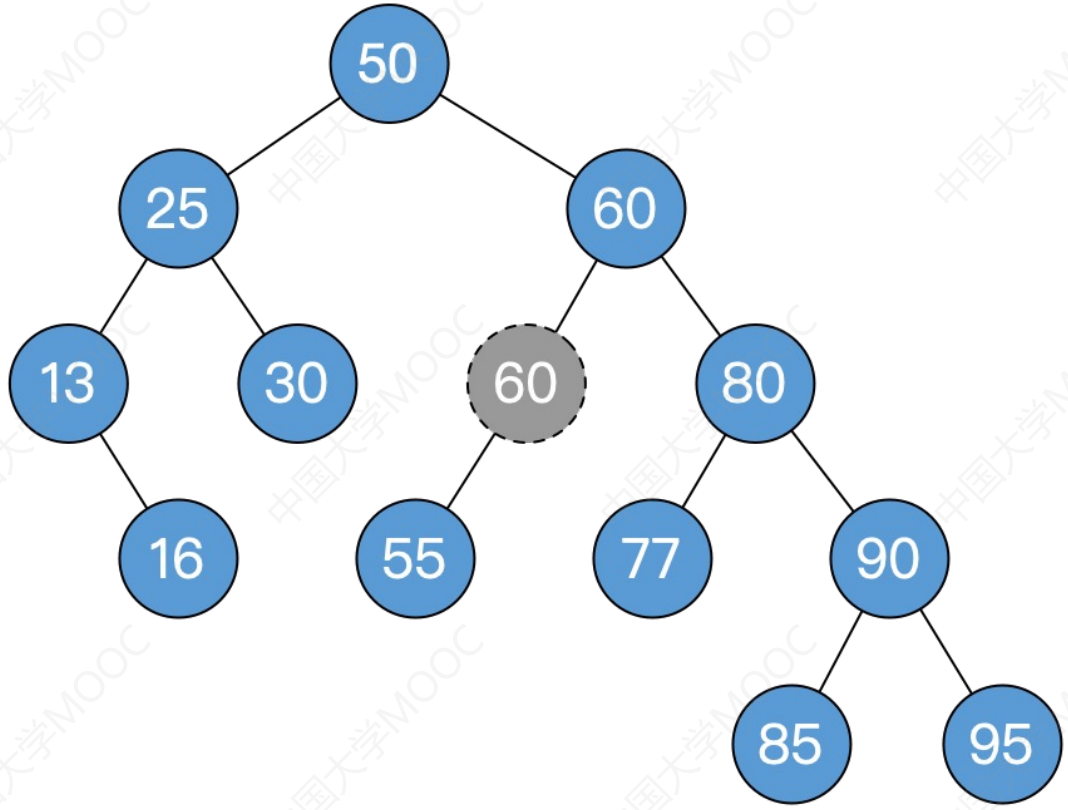
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



被删除结点只有左子树，用子树顶替删除位置（用结点实体顶替）

AVL树删除操作——例5

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

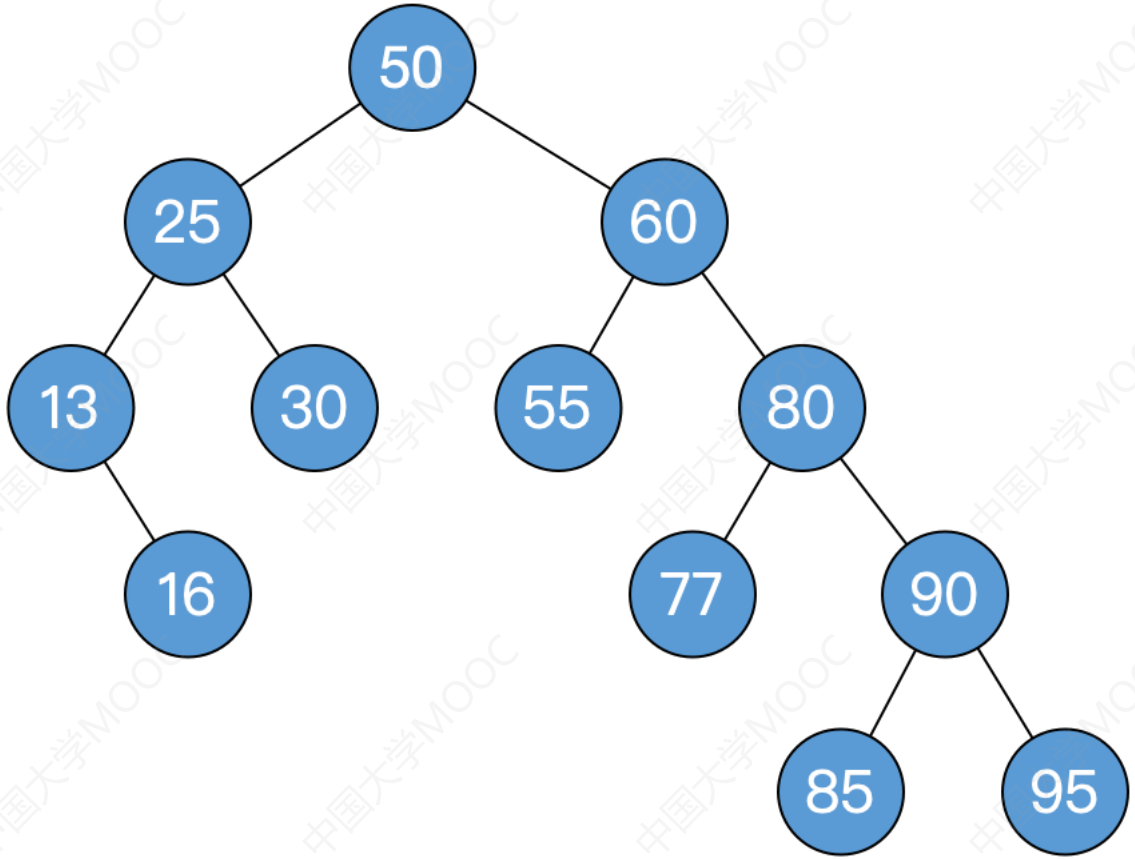
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例5

平衡二叉树的删除操作具体步骤：

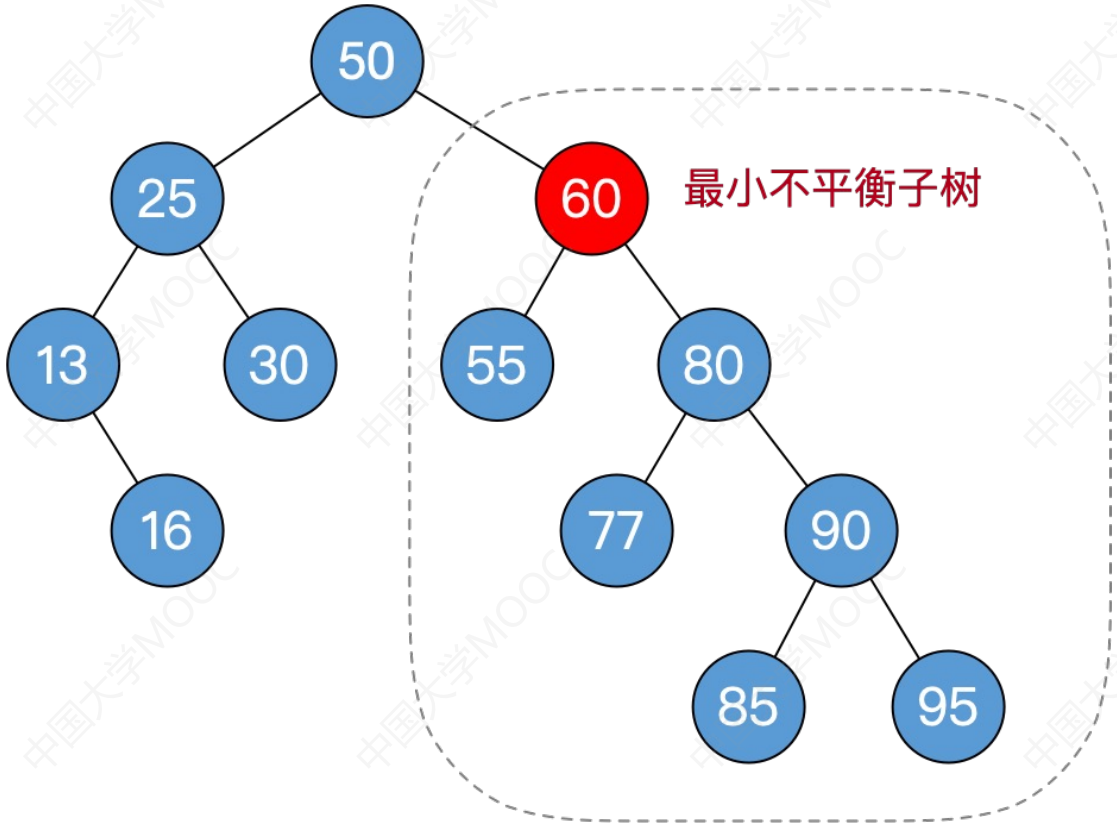
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例5

平衡二叉树的删除操作具体步骤：

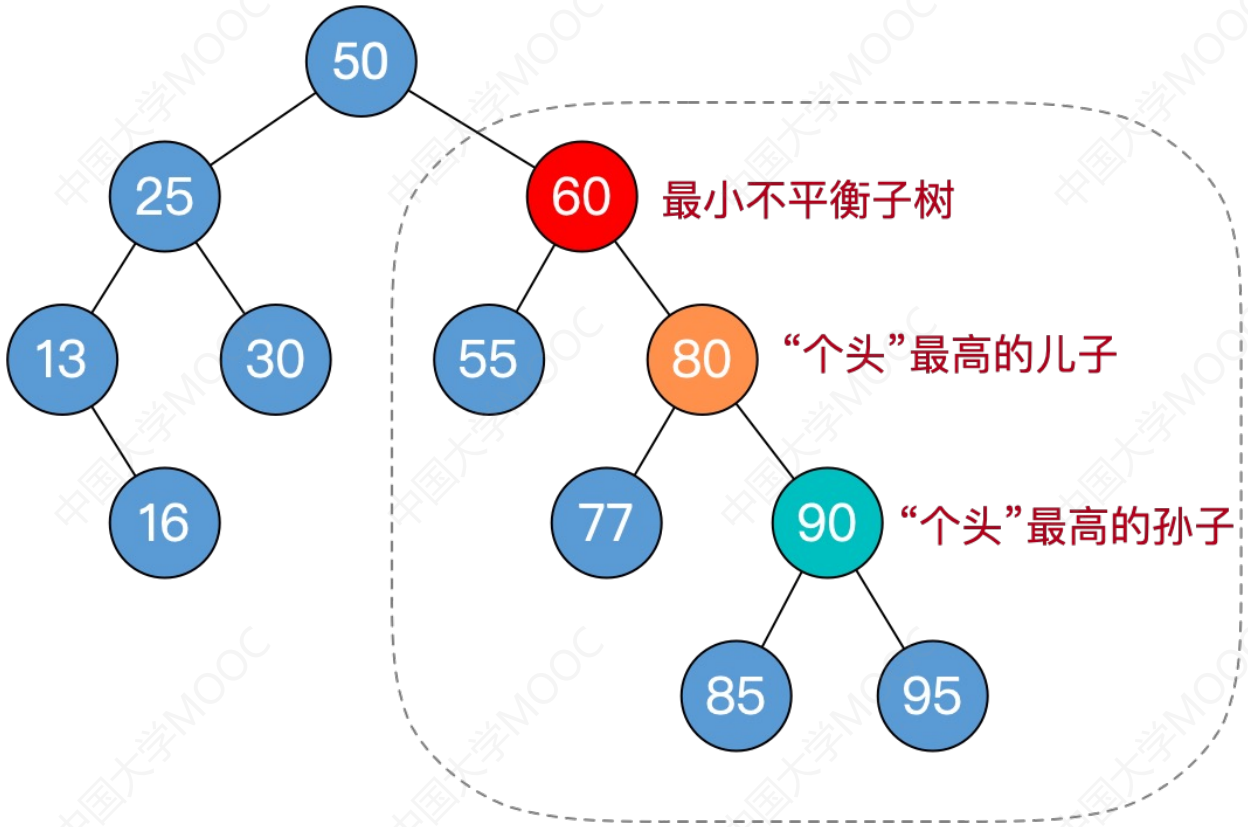
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例5

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

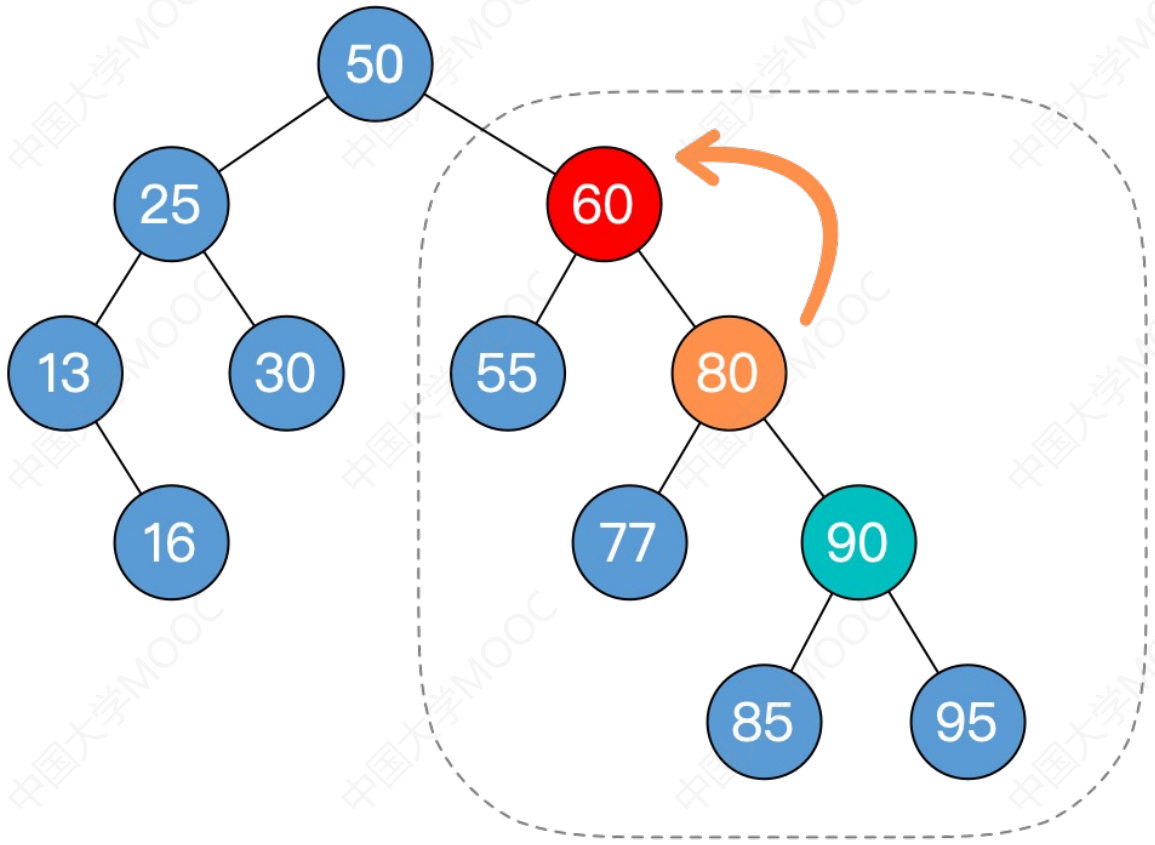
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



孙子在RR，儿子左单旋

AVL树删除操作——例5

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

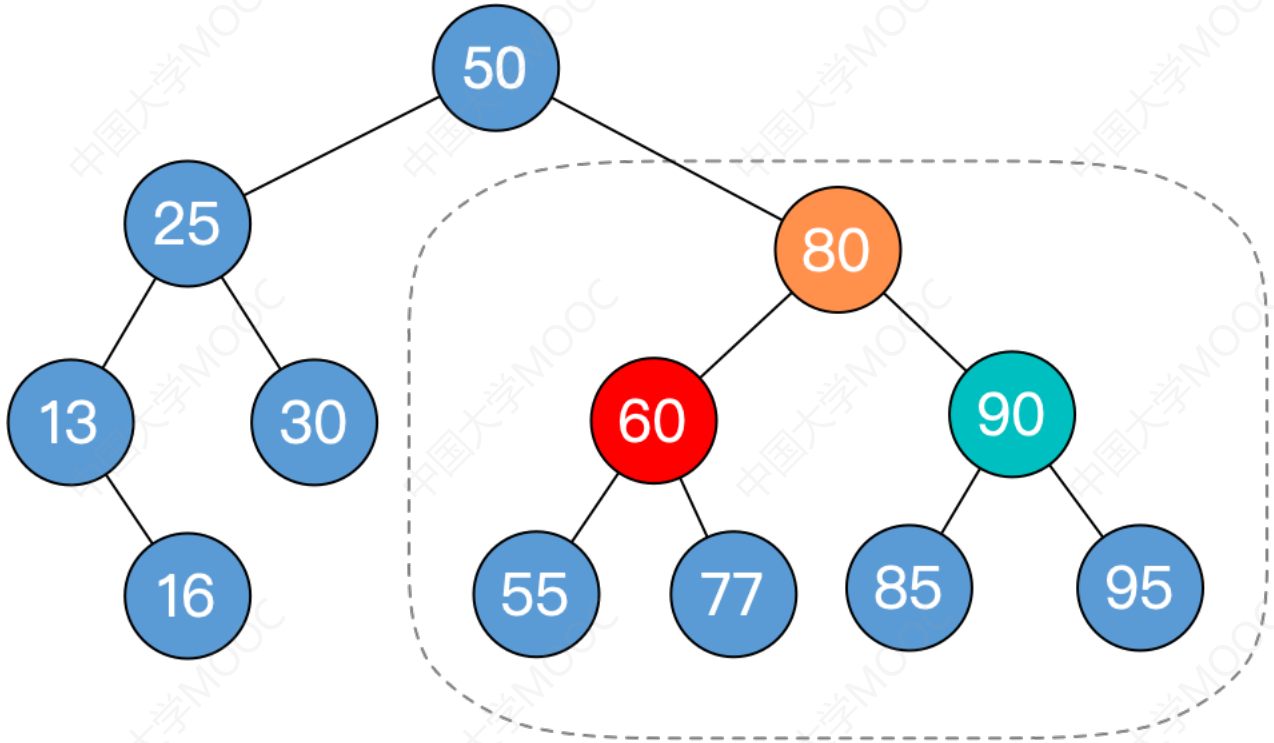
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）



AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

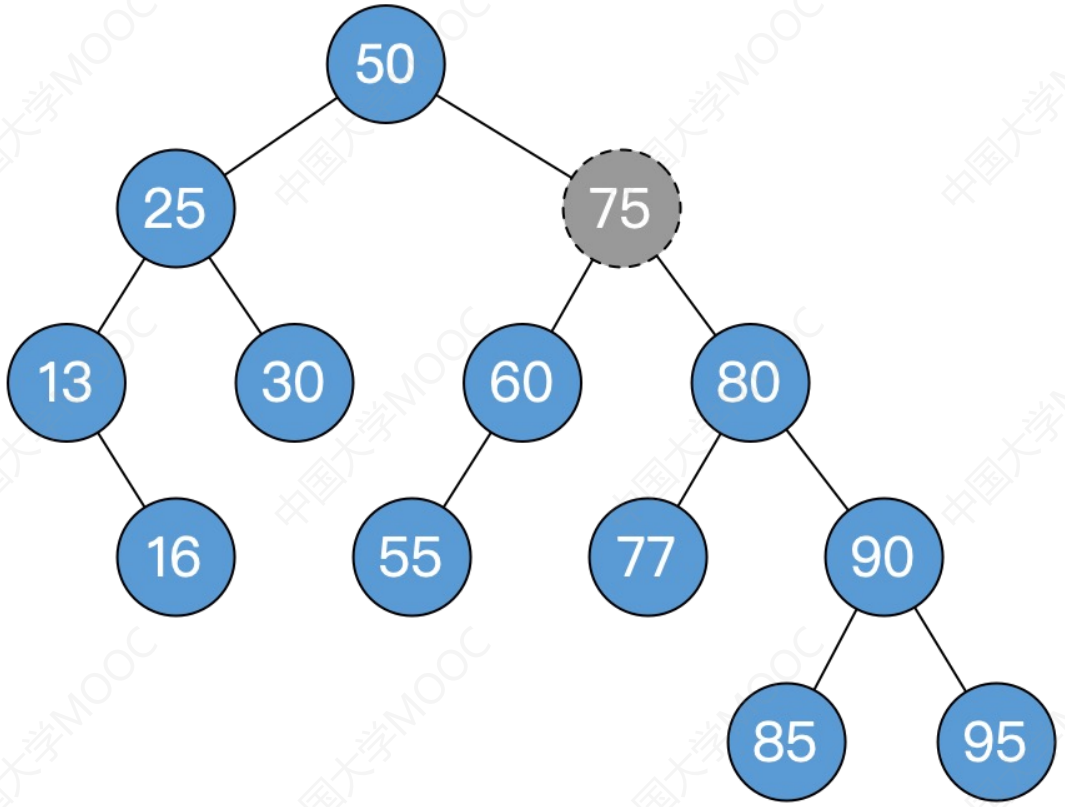
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



被删除结点有左右子树，用后继结点顶替（复制数据即可）
并转化为对后继结点的删除

AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

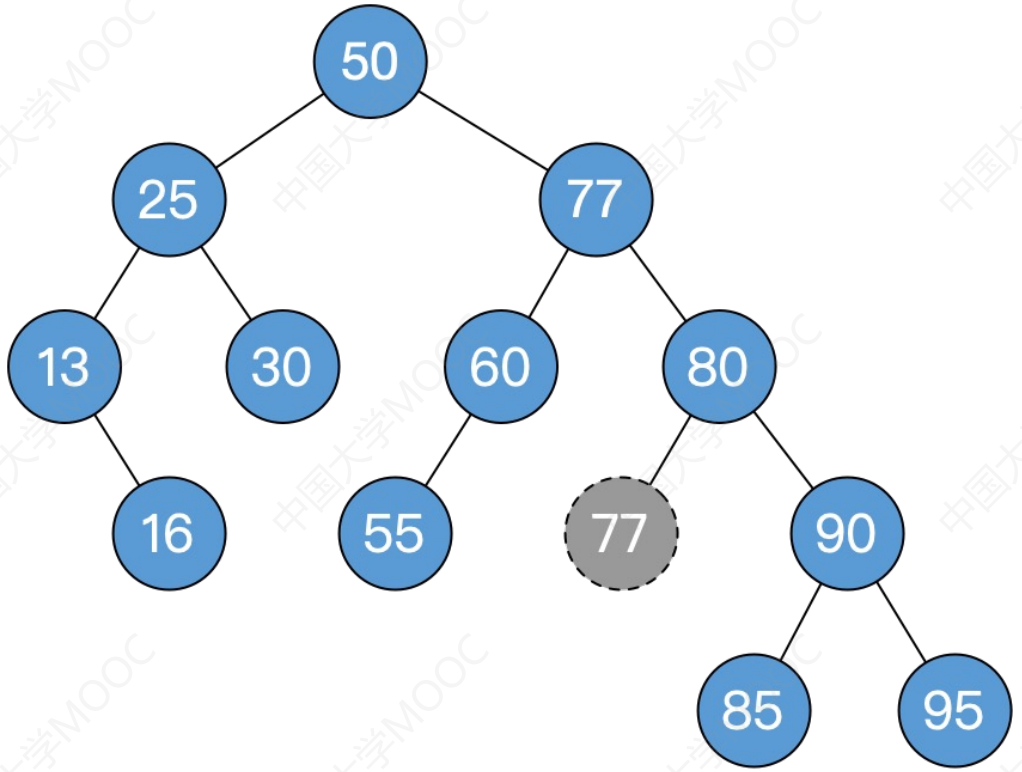
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



被删除结点为叶子，直接删即可

AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

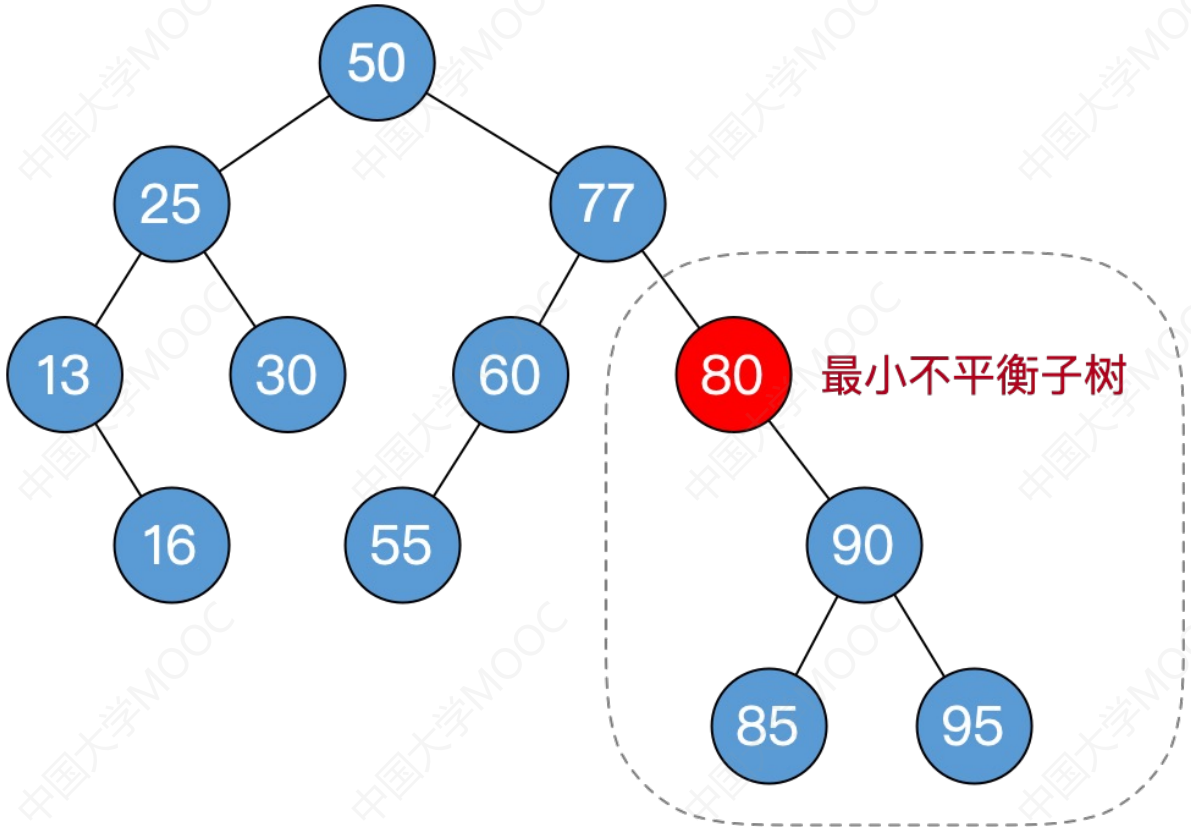
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

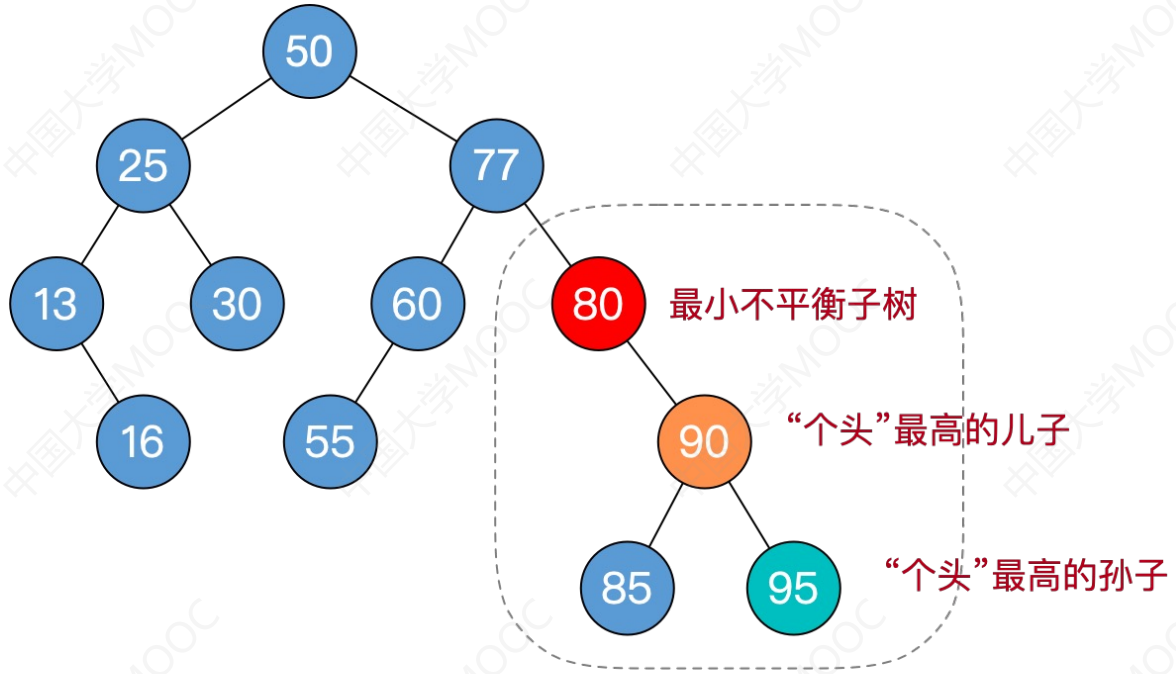
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



孙子在RR，儿子左单旋

AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

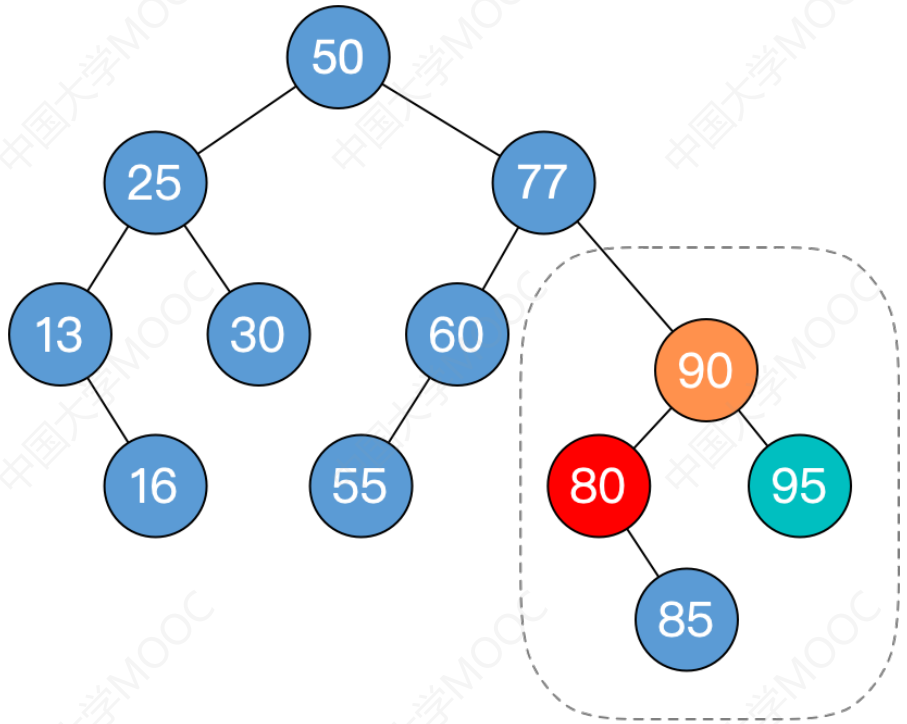
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



儿子左单旋

AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

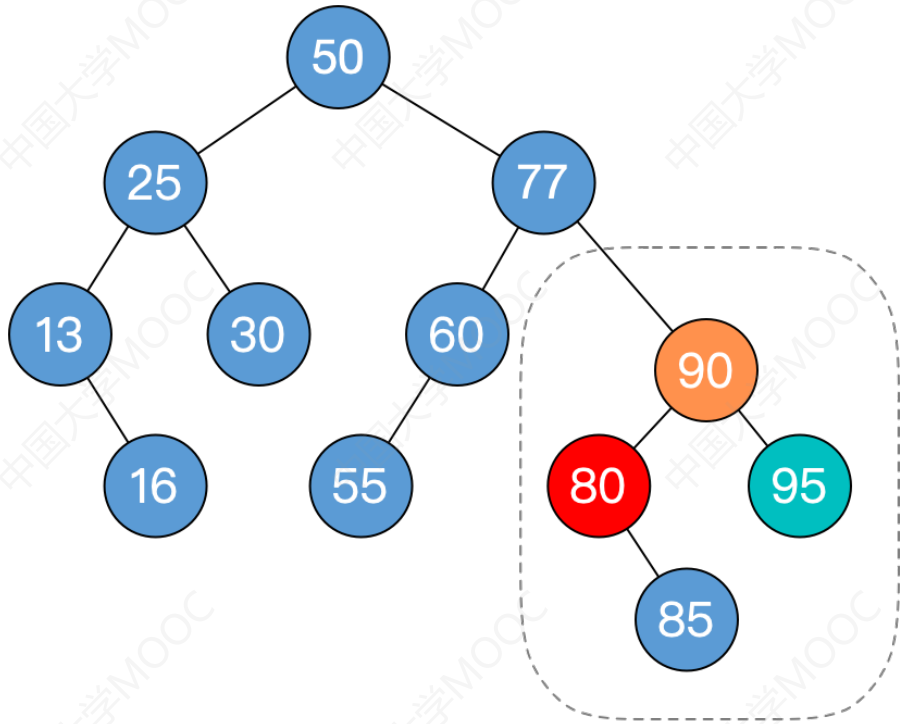
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）



AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

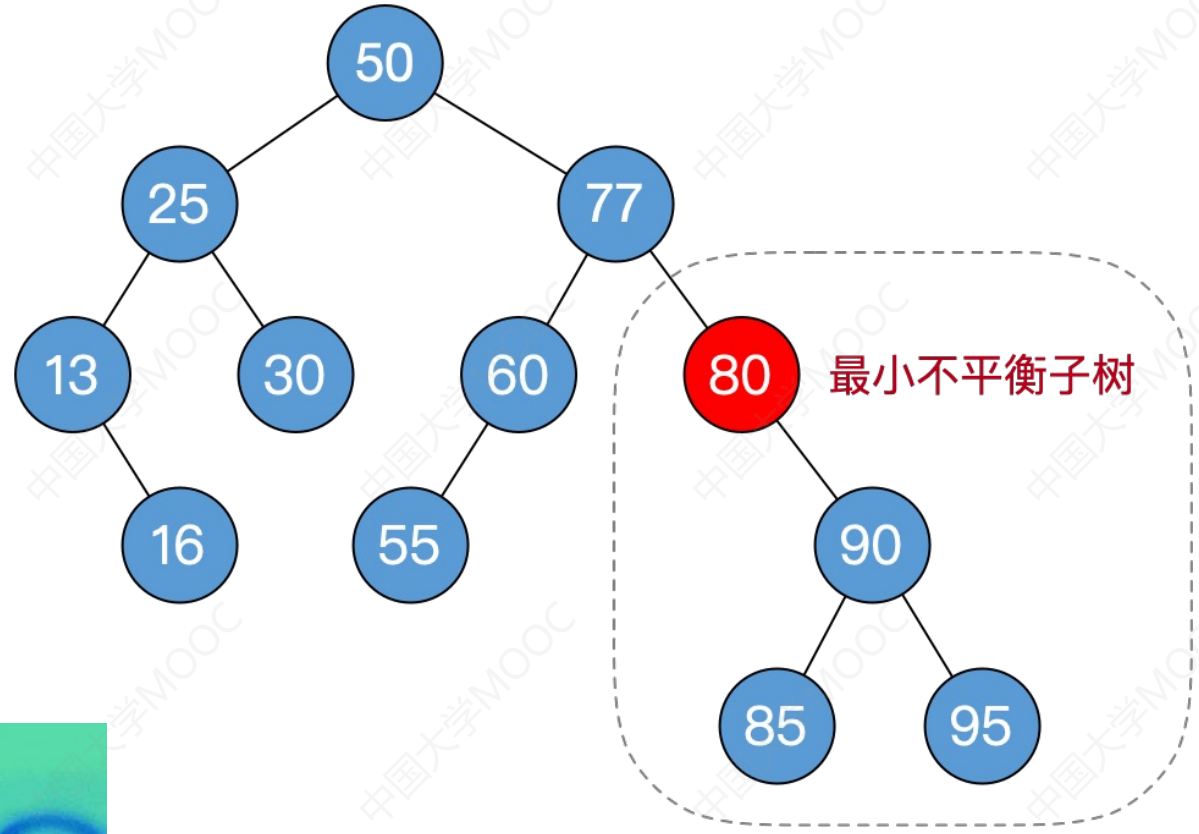
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

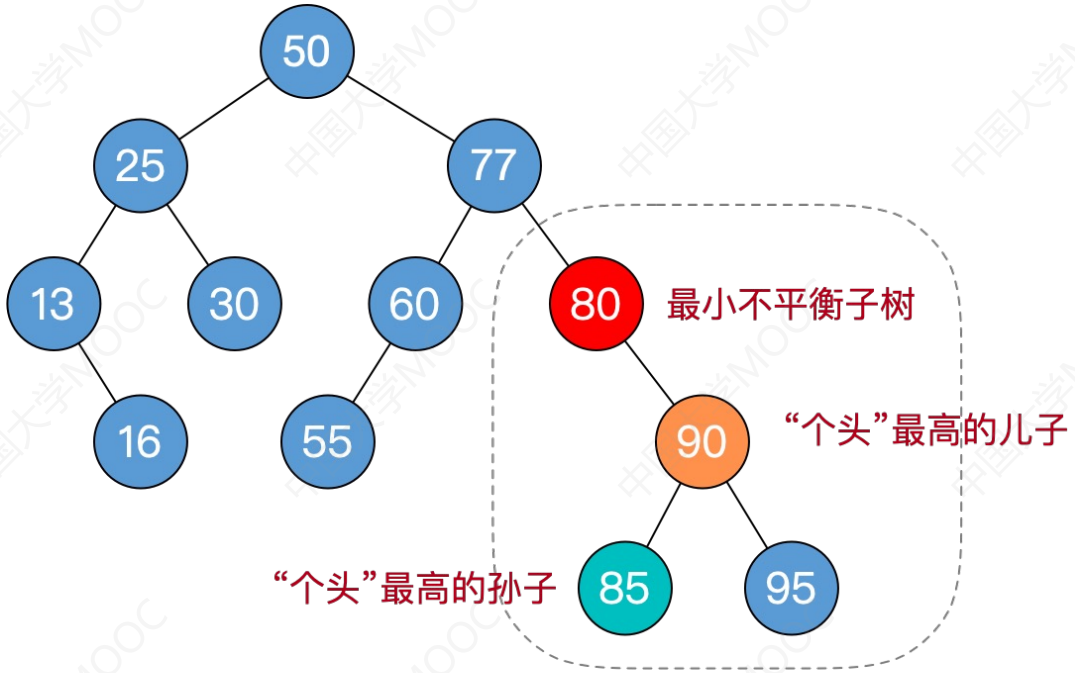
①删除结点（方法同“二叉排序树”）

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②



孙子在RL，孙子先右旋，再左旋

AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

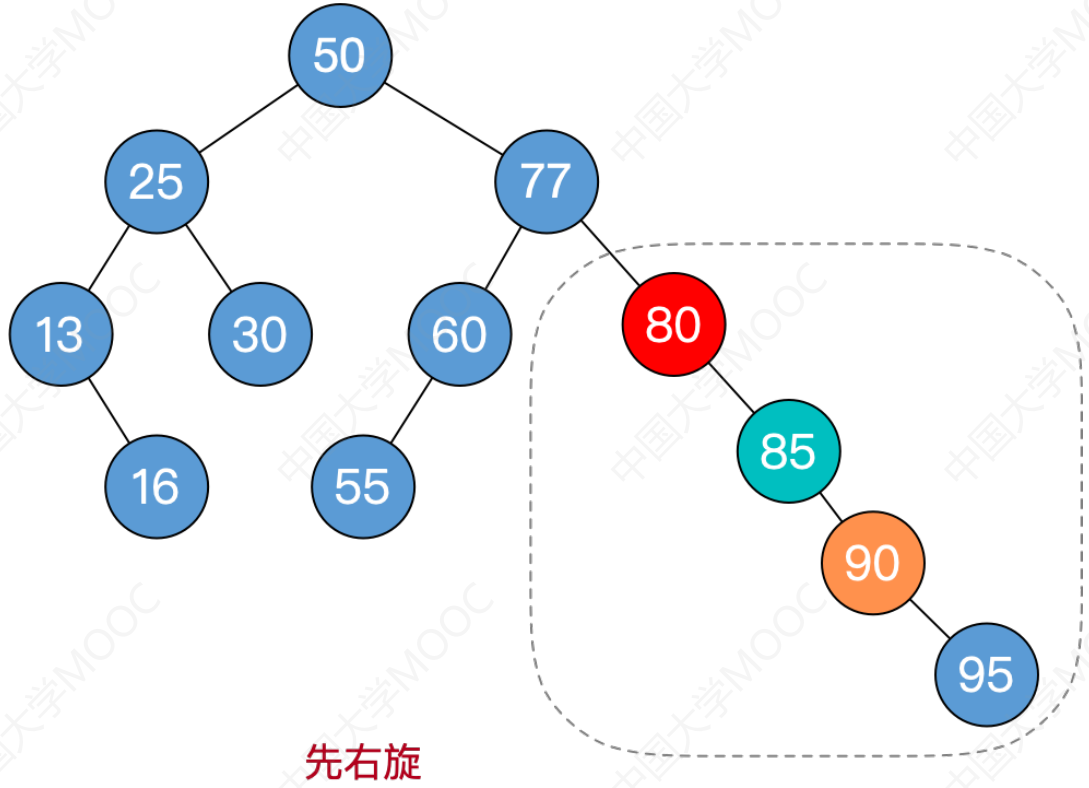
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

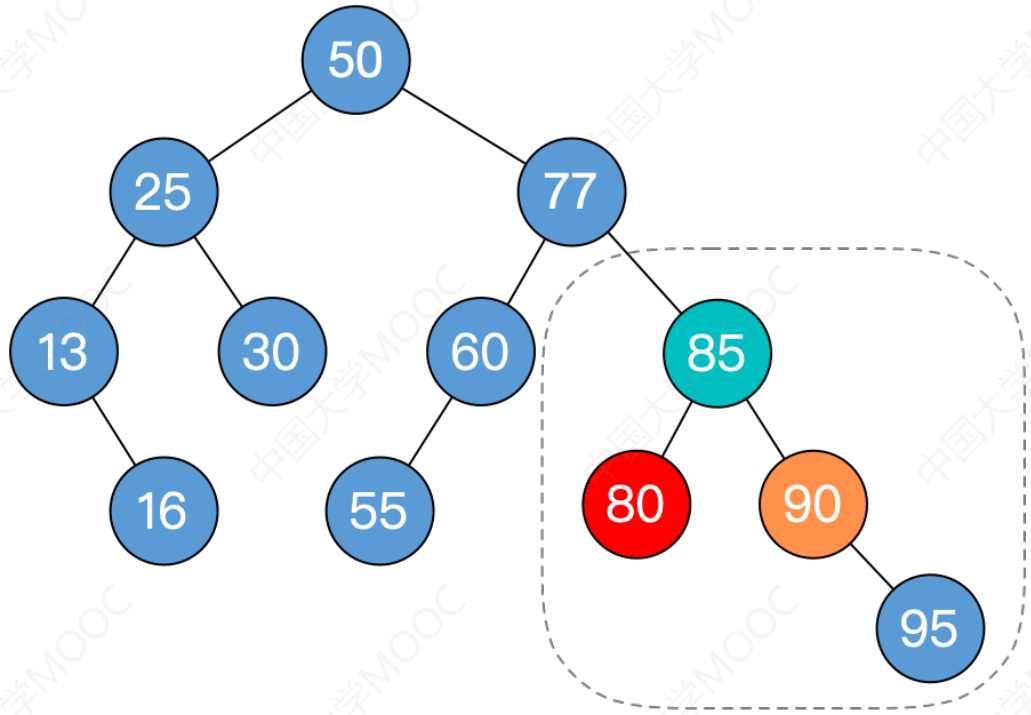
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

⑤如果不平衡向上传导，继续②



再左旋

AVL树删除操作——例6

平衡二叉树的删除操作具体步骤：

①删除结点（方法同“二叉排序树”）

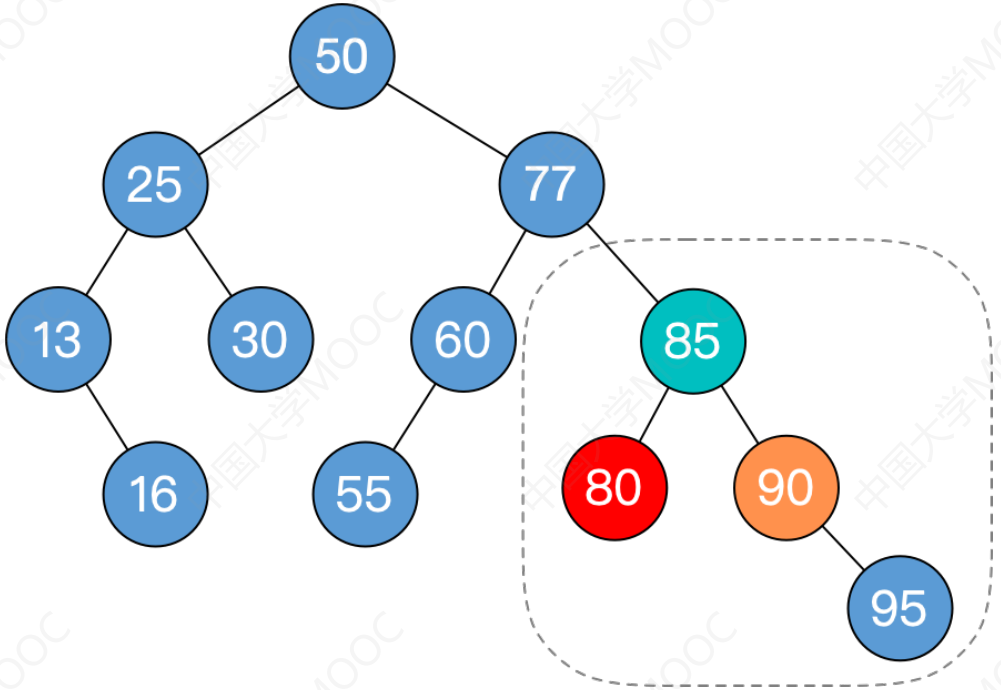
②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）

⑤如果不平衡向上传导，继续②

- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）



不可能考这种有多种处理方式的题目！

知识回顾与重要考点

平衡二叉树的**删除**操作具体步骤：

①删除结点（方法同“二叉排序树”）
- 若删除的结点是叶子，直接删。
- 若删除的结点只有一个子树，用子树顶替删除位置
- 若删除的结点有两棵子树，用前驱（或后继）结点顶替，并转换为对前驱（或后继）结点的删除。

②一路向北找到最小不平衡子树，找不到就完结撒花

③找最小不平衡子树下，“个头”最高的儿子、孙子

④根据孙子的位置，调整平衡（LL/RR/LR/RL）
- 孙子在LL：儿子右单旋
- 孙子在RR：儿子左单旋
- 孙子在LR：孙子先左旋，再右旋
- 孙子在RL：孙子先右旋，再左旋

平衡二叉树删除操作时间复杂度=$O(\log_2 n)$

⑤如果不平衡向上传导，继续②
- 对最小不平衡子树的旋转可能导致树变矮，从而导致上层祖先不平衡（不平衡向上传递）

欢迎大家对本节视频进行评价~

学员评分：7.3.2_2 平...

扫一扫二维码打开或分享给好友

– 腾讯文档 –

可多人实时在线编辑，权限安全可控

公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研