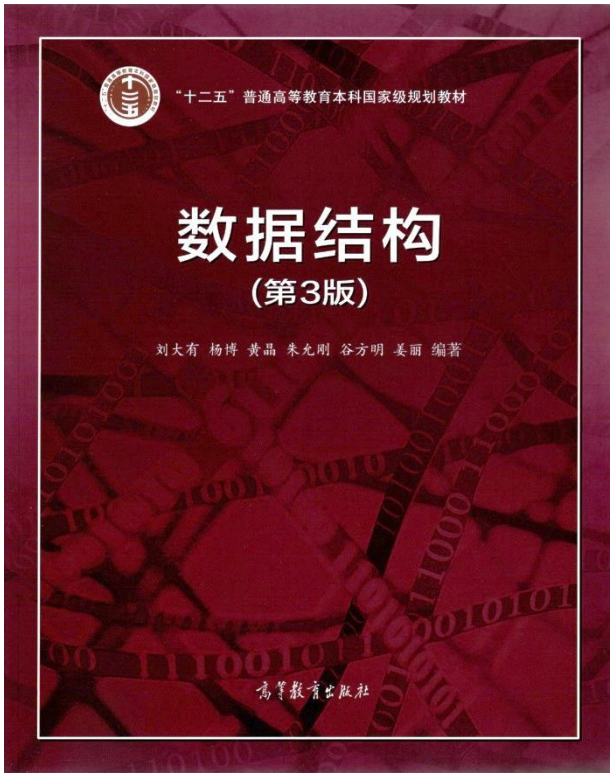


红黑树

- 概述
- 定义与性质
- 插入算法
- 删除算法
- 总结

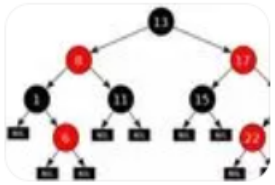
Last updated on 2024.2.26

朱允刚
zhuyungang@jlu.edu.cn

红黑树

- 2022年考研大纲新增“红黑树”。
- 腾讯、阿里、华为、字节跳动、京东、百度、美团等各大公司面试常考题目。

面试阿里,字节跳动,美团必被问到的红黑树



2020年5月21日 因为要满足红黑
节点,那就违反了性质五,需要进行
点,那就只有在要插入节点的父节
博客园 百度快照

京东内部面试题:Mysql索引优化,索引数据结构红黑树,

2021年10月10日 京东内部面试题:Mysql索引优化
ash,B+树详解腾讯内容开放平台
知乎

美团面试被问“红黑树”,我一脸懵逼

7月22日 美团面
的二叉

为C++工程师

硬核图解--字节面试必问的红黑
2020年11月5日 红黑树是面试中一个
个问题。很多人会觉得这个知识点太难,

美团面试:熟悉红黑树?能不能手写一下?

知乎 首发于 帅地玩编程

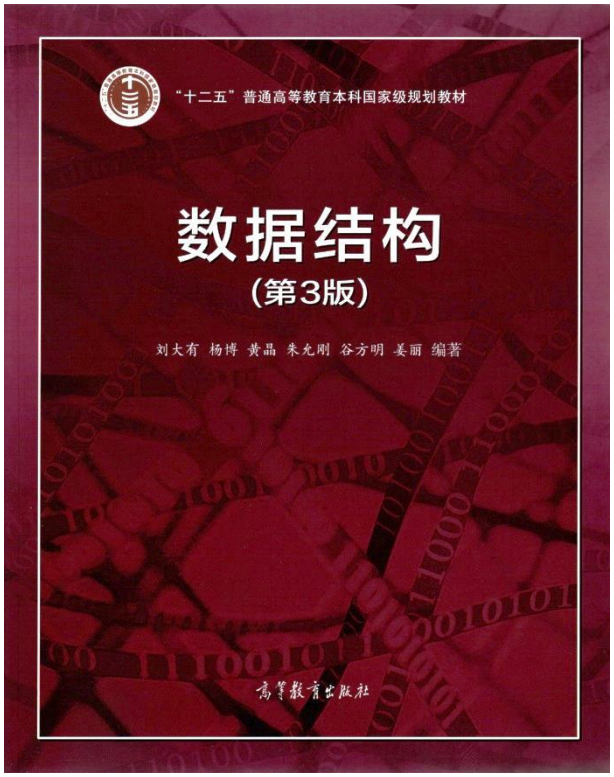
记一次腾讯面试：有了二叉查找树、平衡树（AVL）为啥还需要红黑树?

阿里员工吐槽：学懂AVL和红黑树，字节跳动面试真的太无聊!

讲授红黑树的高校（不完全统计）

国外：麻省理工学院、斯坦福大学、加州大学伯克利分校、普林斯顿大学、卡内基梅隆大学、牛津大学、剑桥大学……

国内：清华大学、北京大学、上海交通大学、浙江大学……



红黑树

- 概述
- 定义与性质
- 插入算法
- 删除算法
- 总结

朱允刚
zhuyungang@jlu.edu.cn

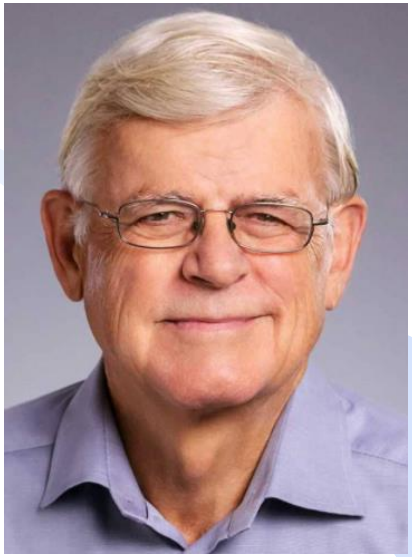
起源

Rudolf Bayer
慕尼黑工业大学教授

Rudolf Bayer, *Symmetric Binary B-Trees: Data Structures and Maintenance Algorithms*, Acta Informatica, 1:290-306, 1972.

起源

Leonidas J. Guibas
斯坦福大学教授
美国工程院院士



Robert Sedgewick
普林斯顿大学教授

Leo J. Guibas, **Robert Sedgewick**, *A Dichromatic Framework for Balanced Trees*, Proceedings of the 19th Annual Symposium on Foundations of Computer Science, pages 8-21. IEEE Computer Society, 1978.

动机

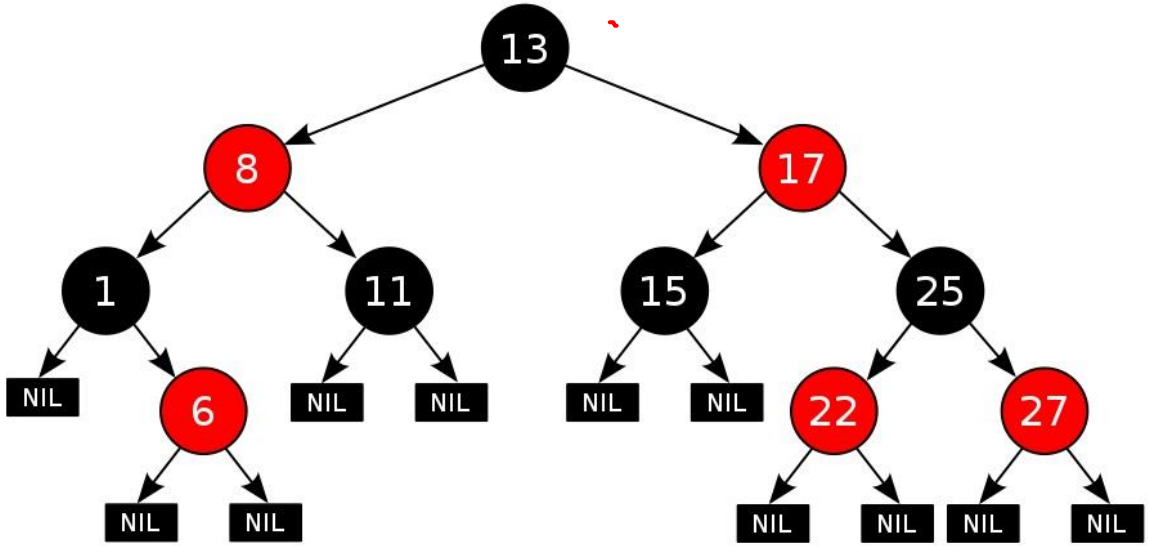
- 二叉查找树查找、插入、删除最坏情况时间复杂度可能退化为O(*n*)。

- AVL树很好的限制了树的高度为O(log*n*)，插入、删除、查找的最坏时间复杂度均为O(log*n*)；但删除操作最多需要做O(log*n*)次旋转。

红黑树的定义

红黑树是具有如下特点的二叉查找树：

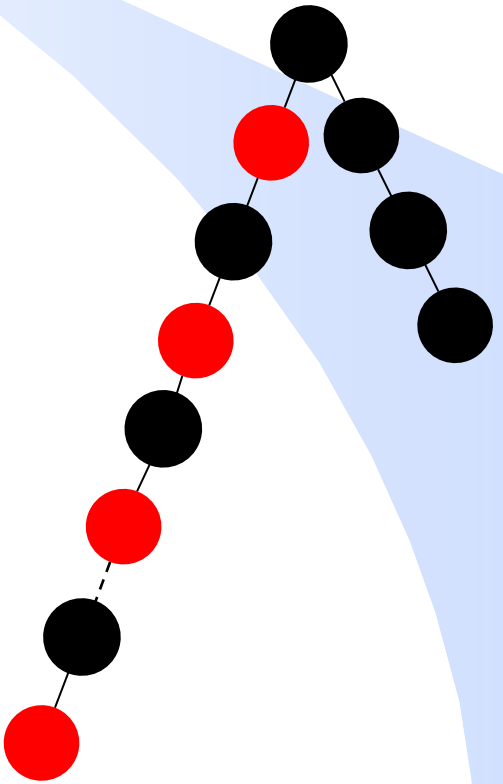
① 每个结点是红色或黑色的；
② 根结点为黑色；
③ 外结点为黑色；
④ 如果一个结点是红色，那么它的孩子必须是黑色。
⑤ 任一结点到外结点的路径上，包含相同数目的黑结点。



结点的黑高度：该结点到外结点的路径上包含的黑结点数目
红黑树的黑高度：根结点的黑高度

红黑树的性质

- 若忽略红结点而只考虑黑结点，则这棵树是平衡的。
- 任何一条路径上不能有两个连续的红结点。从任意结点出发最长的路径（红黑结点间隔组成）是最短路径（仅由黑结点组成）的2倍。
- 任何一个结点的左右子树的高度最多相差2倍。
- 红黑树的平衡性比AVL树更弱。
- 平均和最坏高度： O(log*n*)。
- 查找、插入、删除操作的平均和最坏时间复杂度是O(log*n*)，且仅涉及O(1)次旋转。



红黑树的定义

红黑树是具有如下特点的二叉查找树：

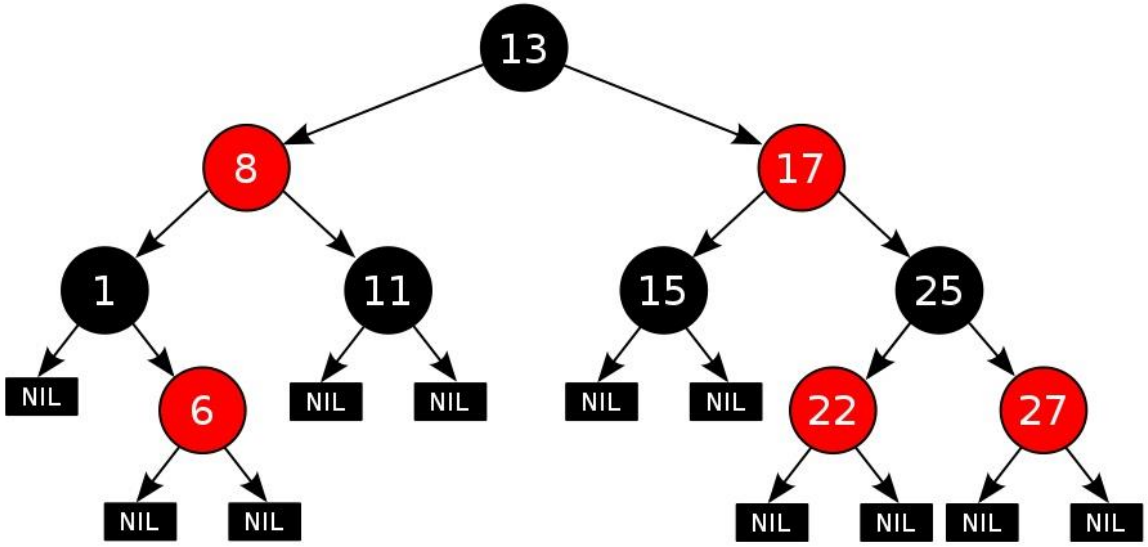
① 非红即黑 是红色或黑色的；

② 根黑 黑色；

③ 外黑 黑色；

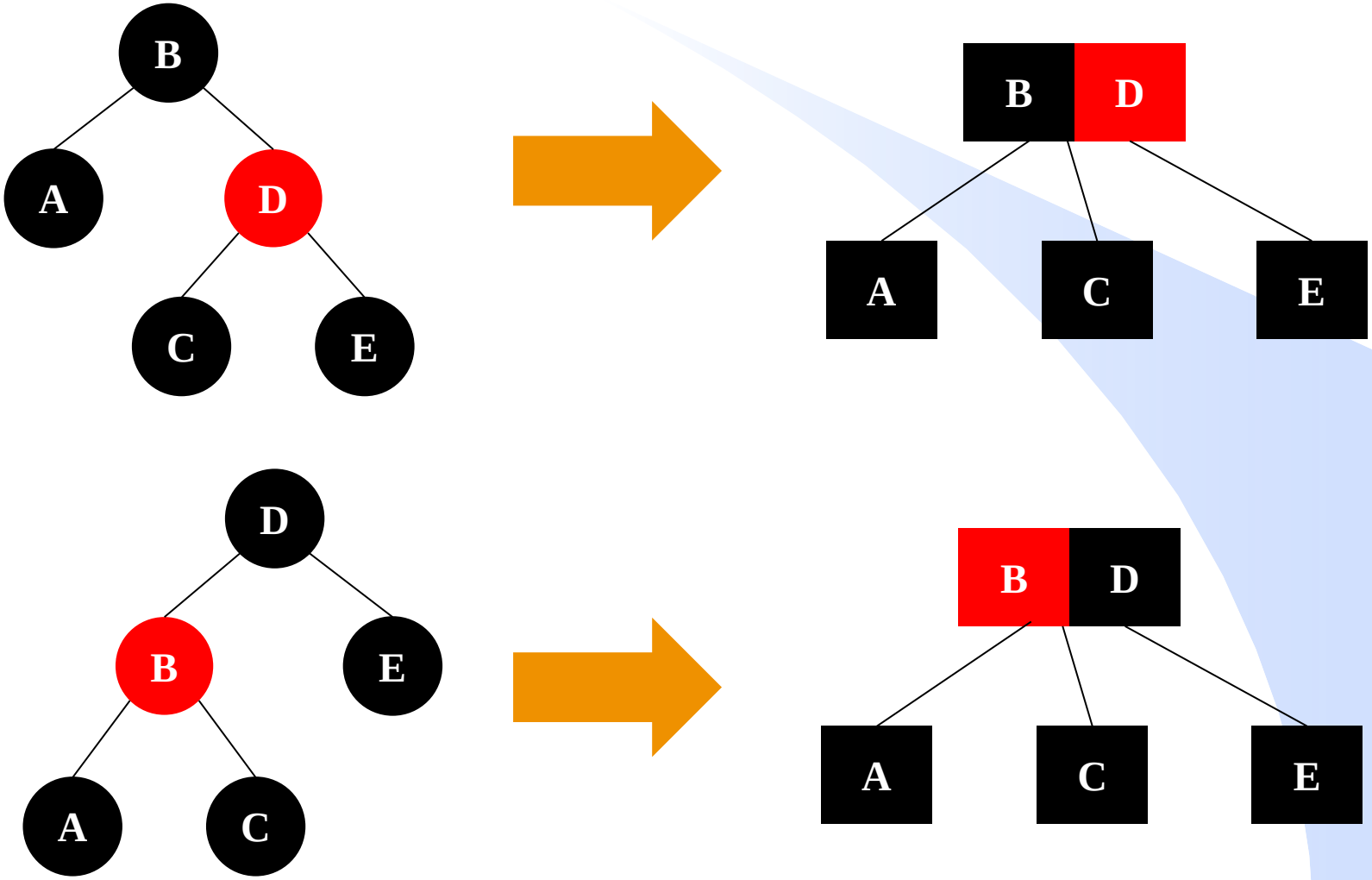
④ 红父黑子 结点是红色，那么它的孩子必须是黑色。

⑤ 黑高相等 到外结点的路径上，包含相同数目的黑结点。



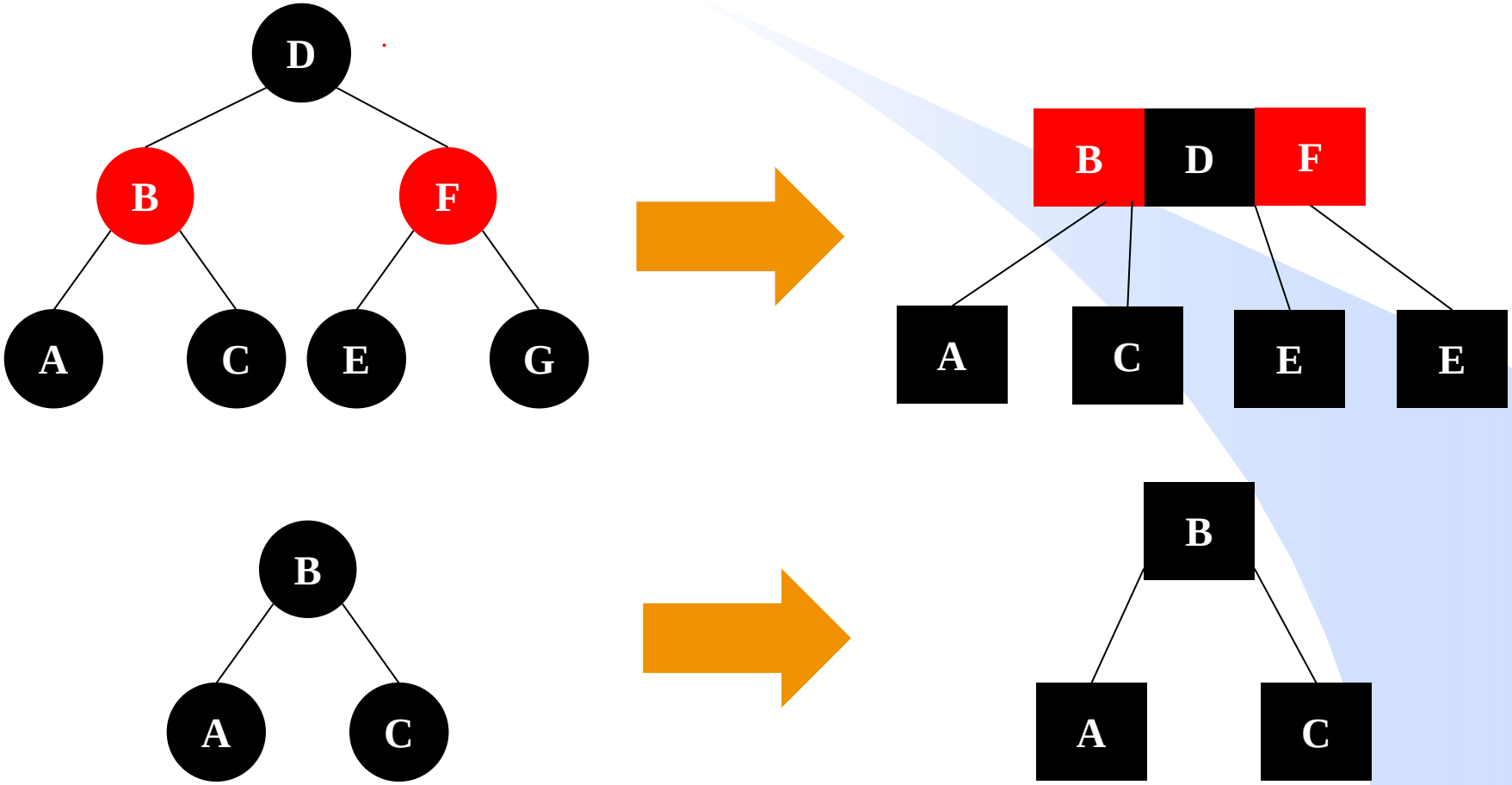
红黑树与4阶B树的等价性

提升变换：将每个红结点向上提升至与其父结点平齐（等高）。



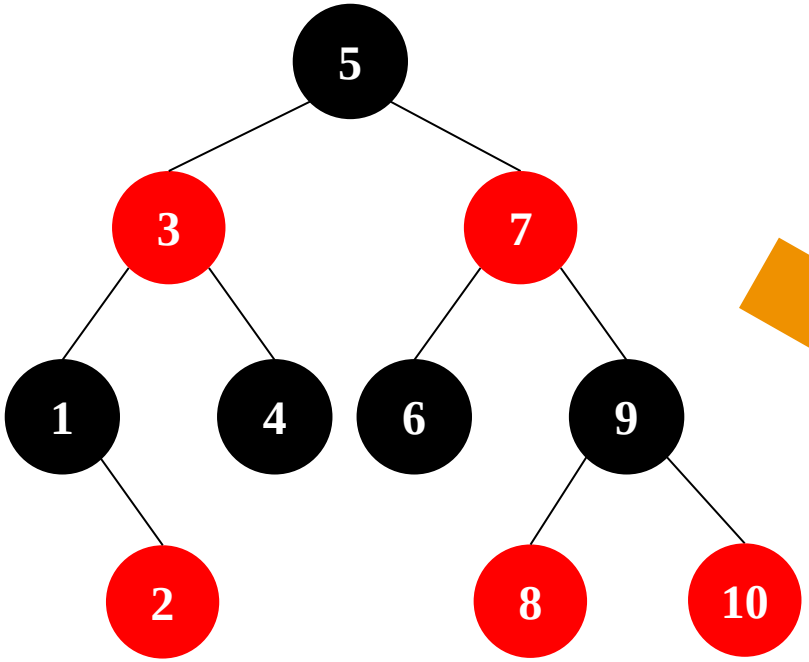
红黑树与4阶B树的等价性

提升变换：将每个红结点向上提升至与其父结点平齐（等高）。

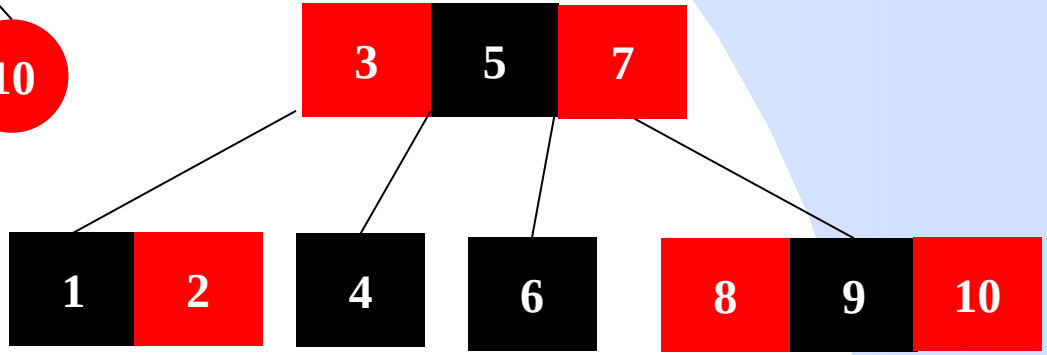


红黑树与4阶B树的等价性

经提升变换后，任何一棵红黑树都自然对应一棵4阶B树。



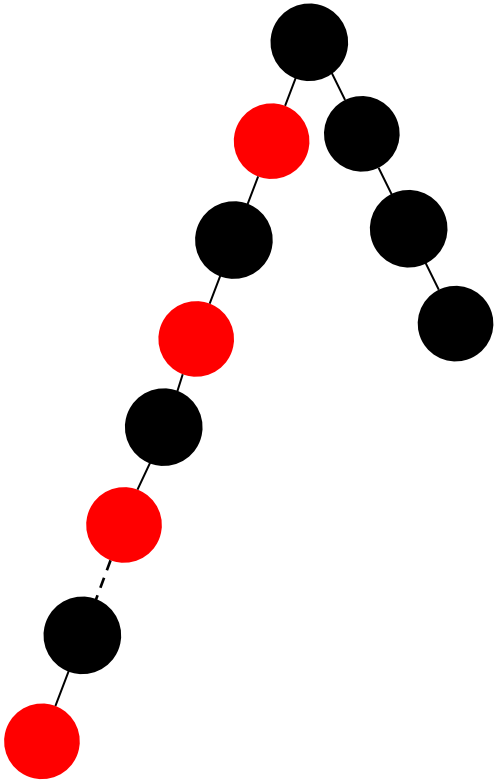
红黑树中红结点父亲必然是黑的，故对应的B树中不会出现两个红关键词相邻，每个结点里必有黑关键词



红黑树的高度

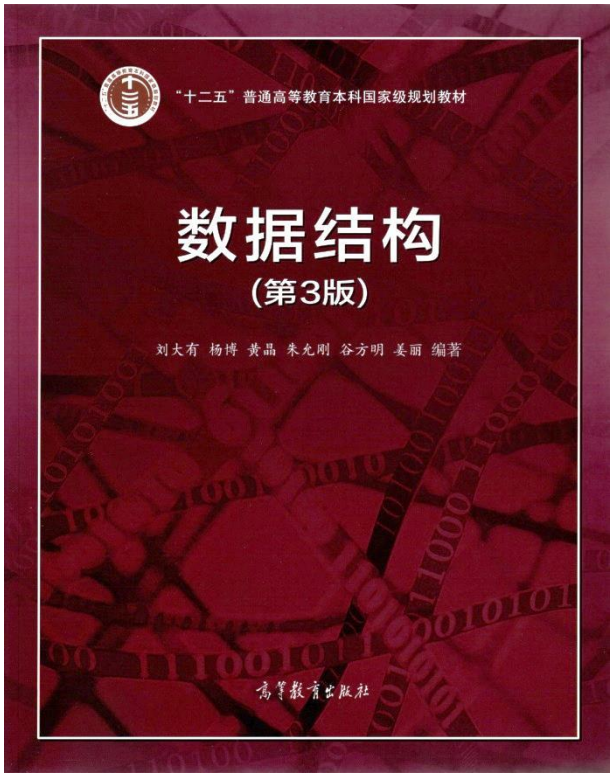
***n*个内结点的红黑树，高度*h*=O(log*n*)**

log*n* ≤ *h* ≤ 2log*n*



$h \leq 2H_B$

$\leq 2\log n$



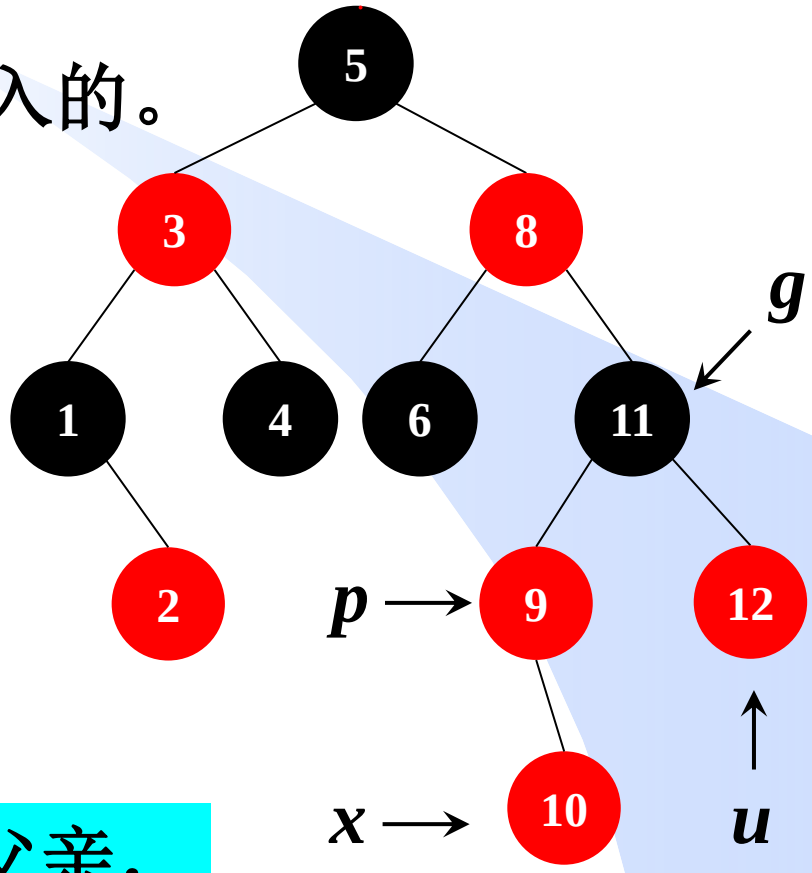
红黑树

- 概述
- 定义与性质
- **插入算法**
- 删除算法
- 总结

朱允刚
zhuyungang@jlu.edu.cn

红黑树的插入

- 查找，若查找成功则不插入，若查找失败，在查找失败的位置插入新结点x。
- 新结点x总是作为叶结点插入的。
- 新结点x必为红色。

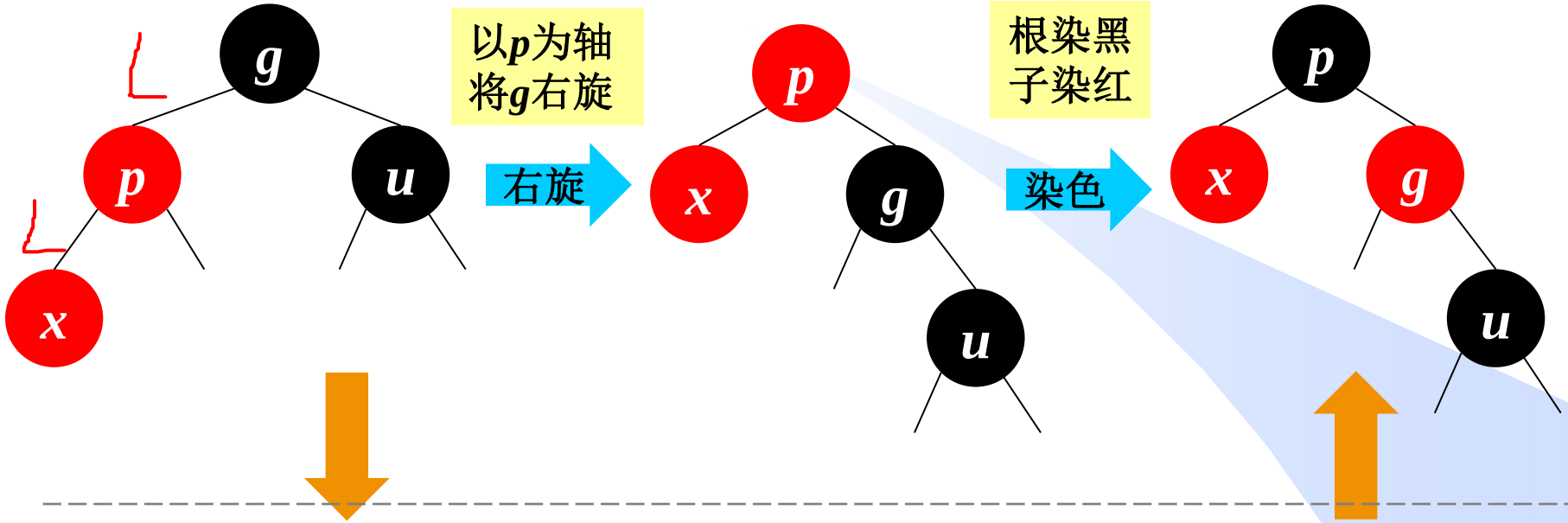


- ✓ 若x的父亲是黑色，插入过程结束。
- ✓ 若x的父亲是红色：双红缺陷。

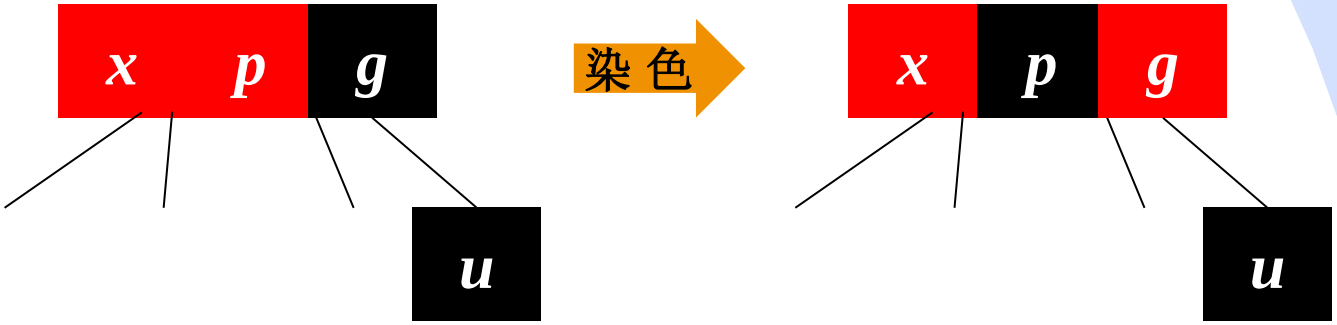
x为新插入节点，p为x的父亲，g为x的爷爷，u为x的叔叔。

双红修正—情况1：x的叔叔是黑色

1.1 g到x的路径为LL型。

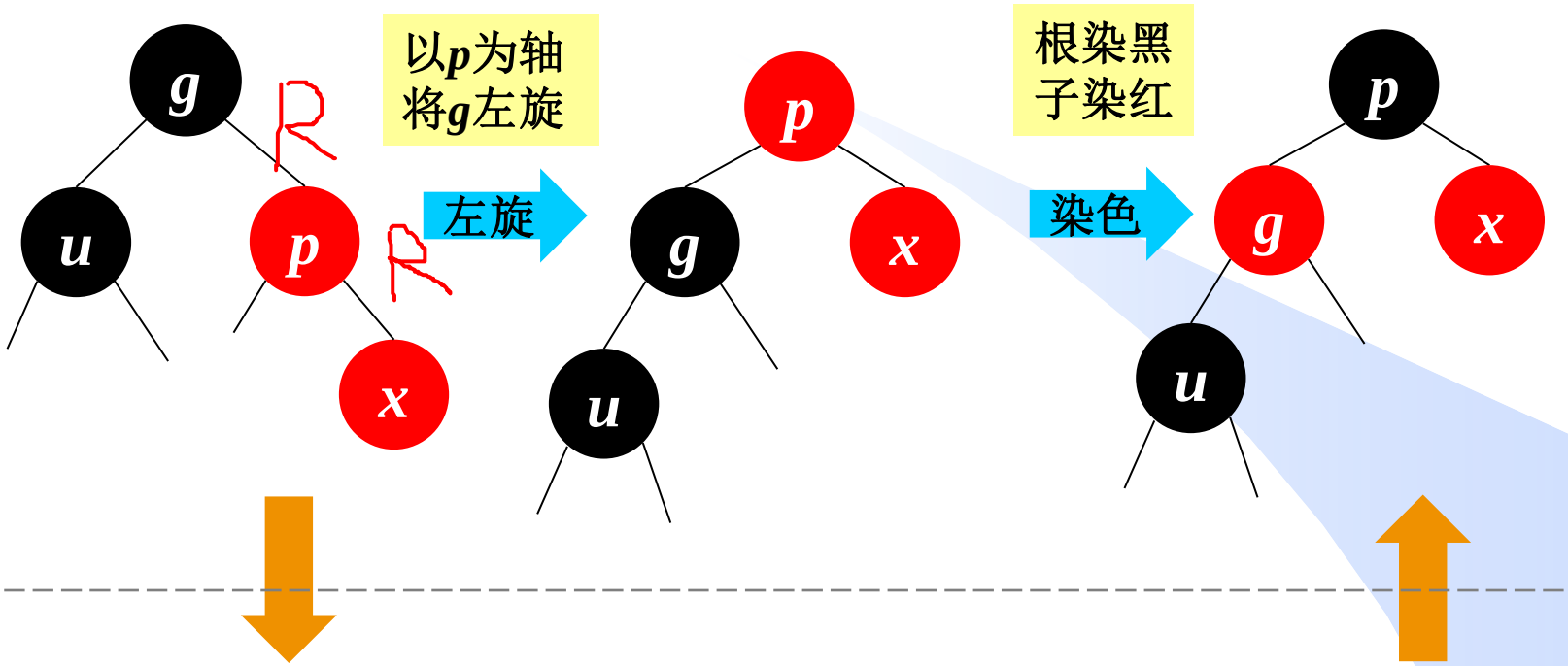


B树

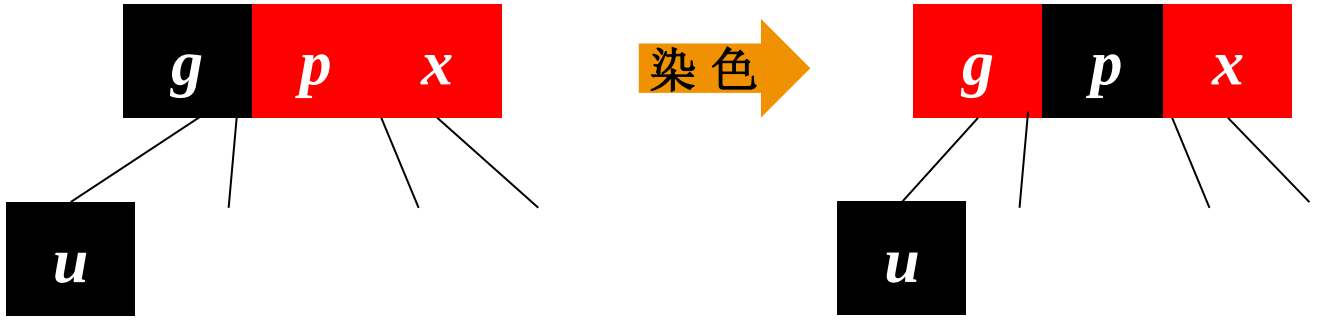


双红修正—情况1：x的叔叔是黑色

1.2 g到x的路径为RR型。

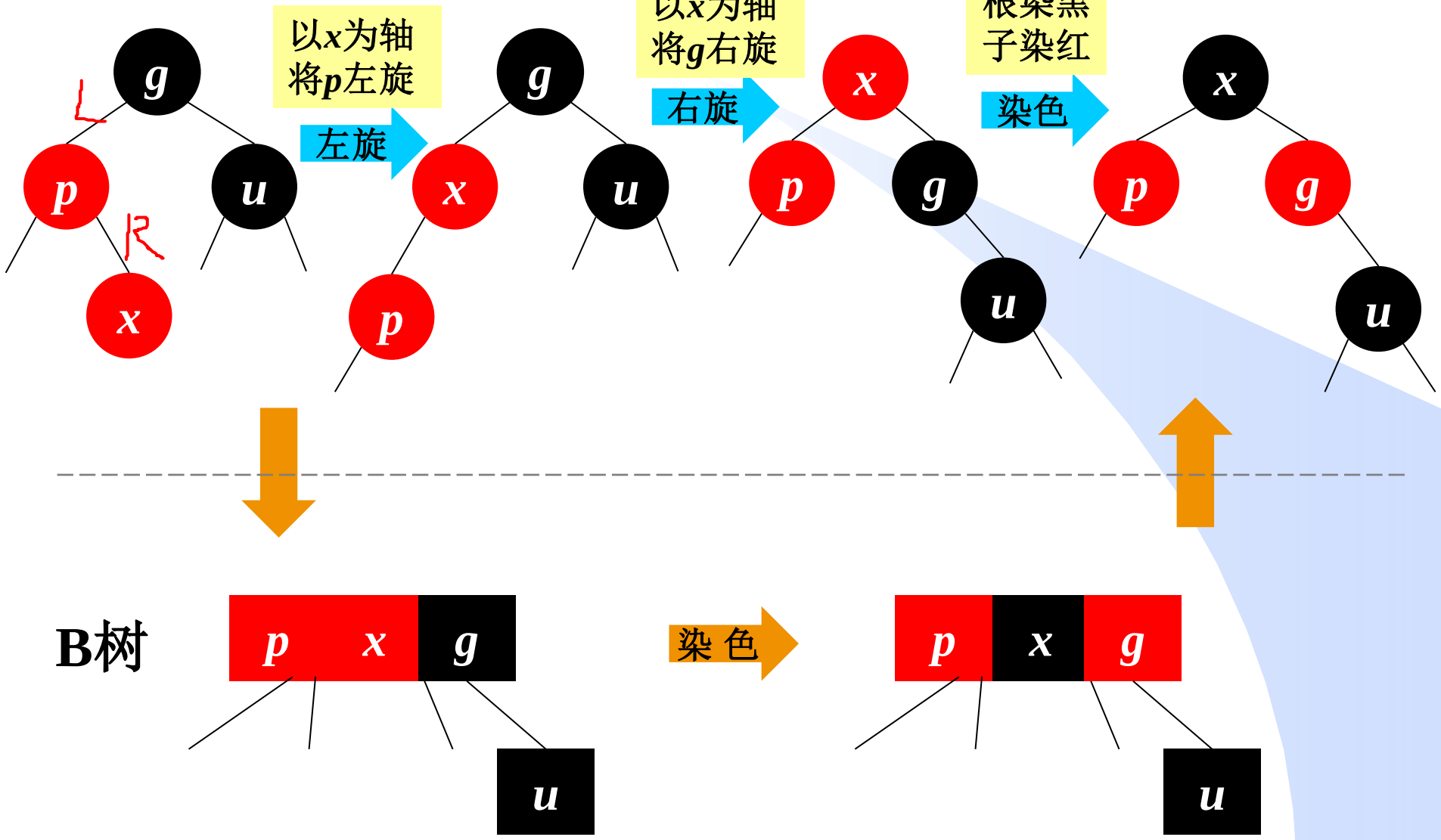


B树



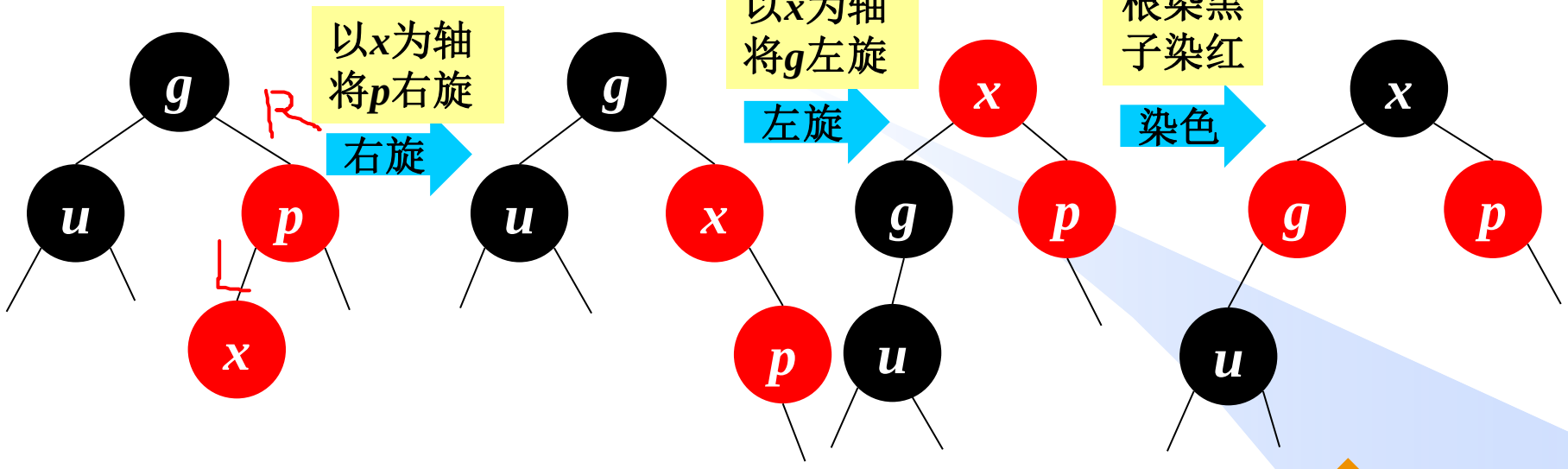
双红修正—情况1：x的叔叔是黑色

1.3 g到x的路径为LR型。

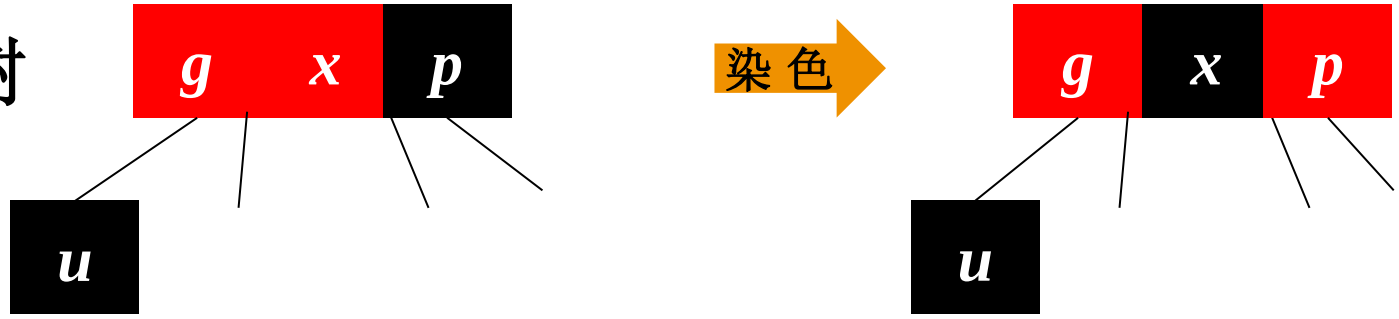


双红修正—情况1：x的叔叔是黑色

1.4 g到x的路径为RL型。

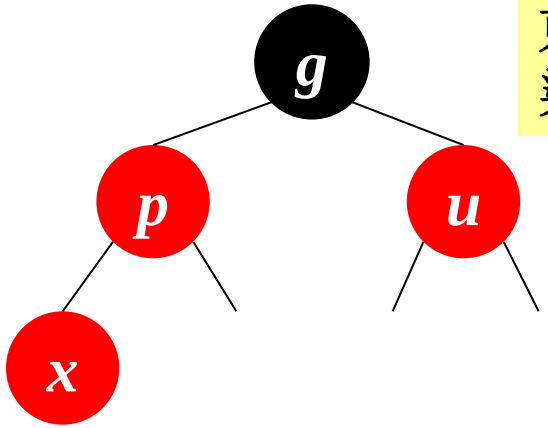


B树

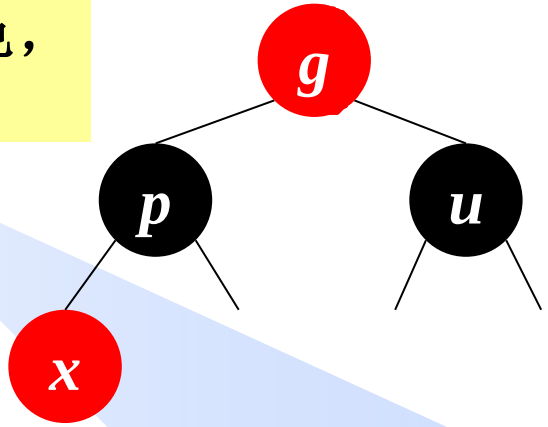


双红修正—情况2：x的叔叔是红色

若g父红，继续向上做双红修正。
若g叔黑，则最多做2次旋转即可结束，若g叔红，继续向上做染色，染色操作可能持续至树根。



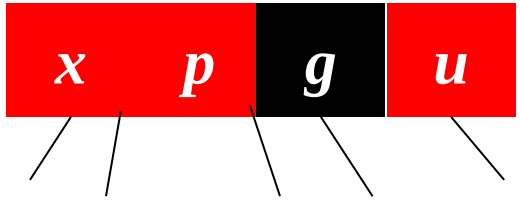
染色

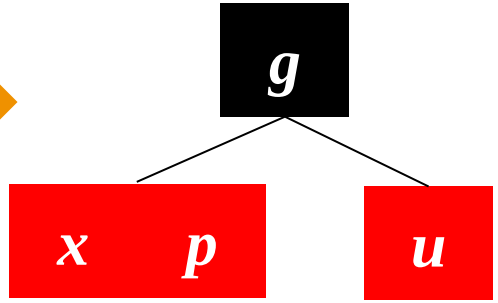


p和u染黑，g染红
若g为根则染黑

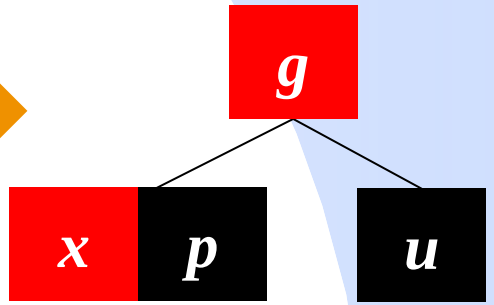
B树

分裂

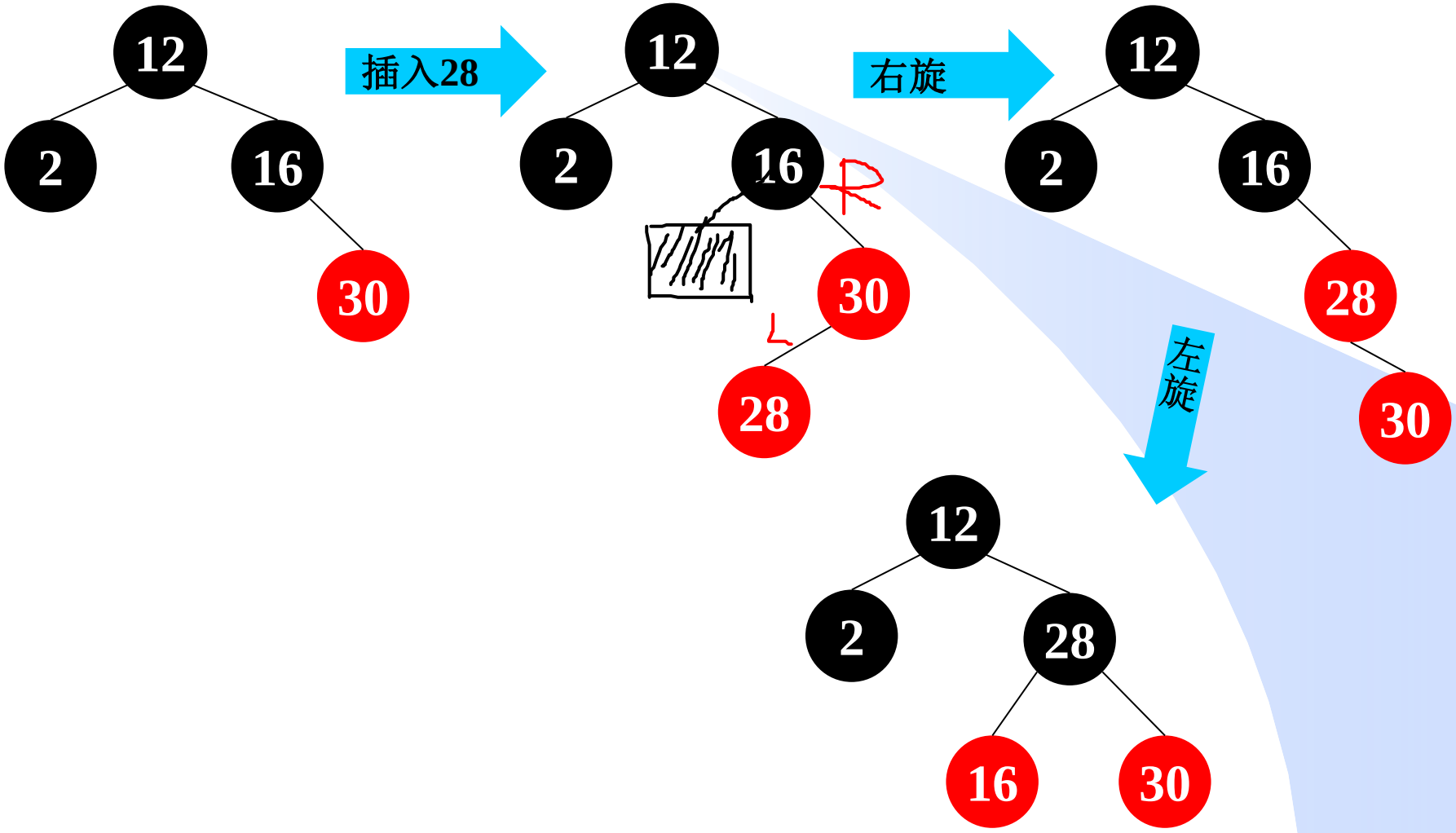




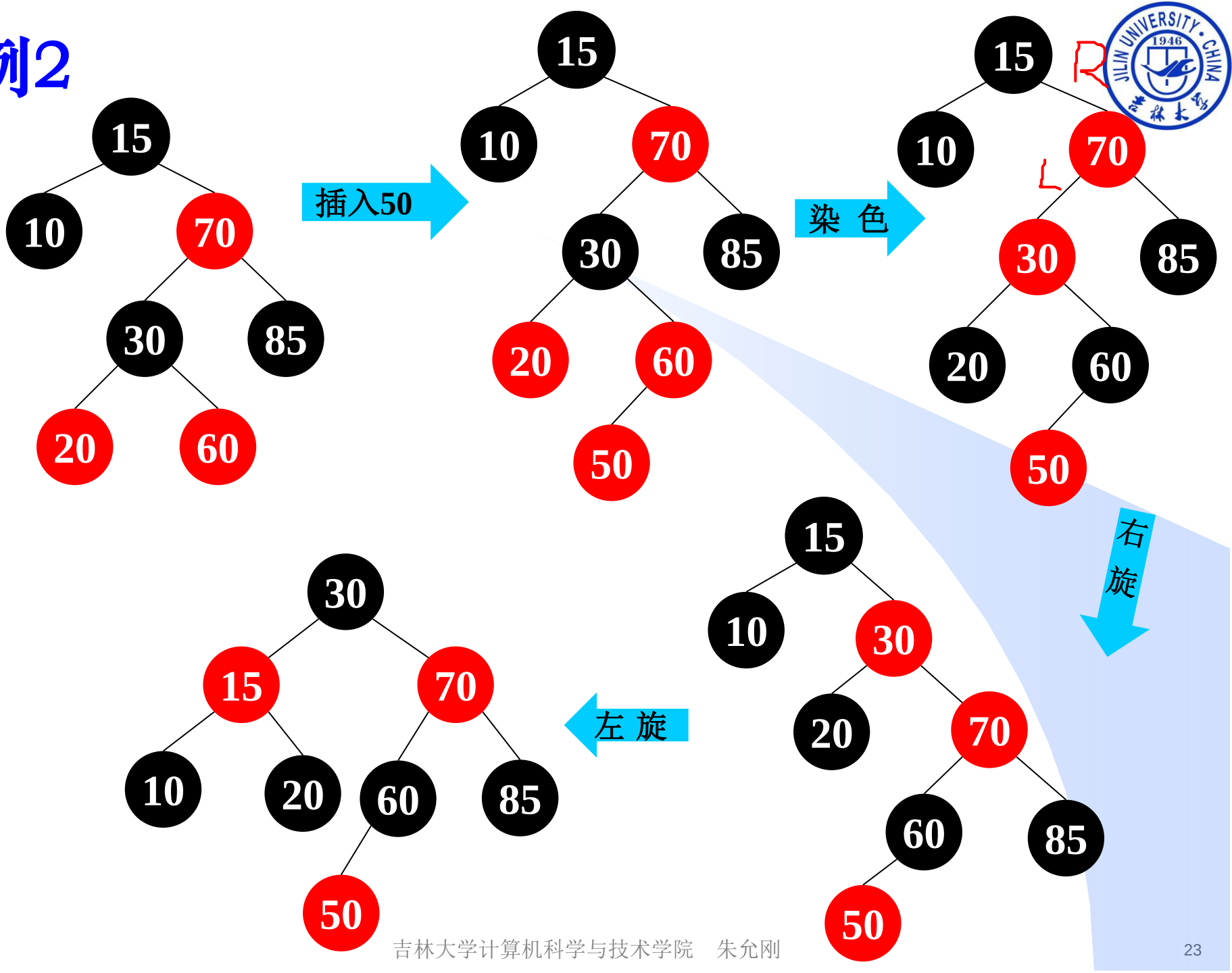
染色



例1



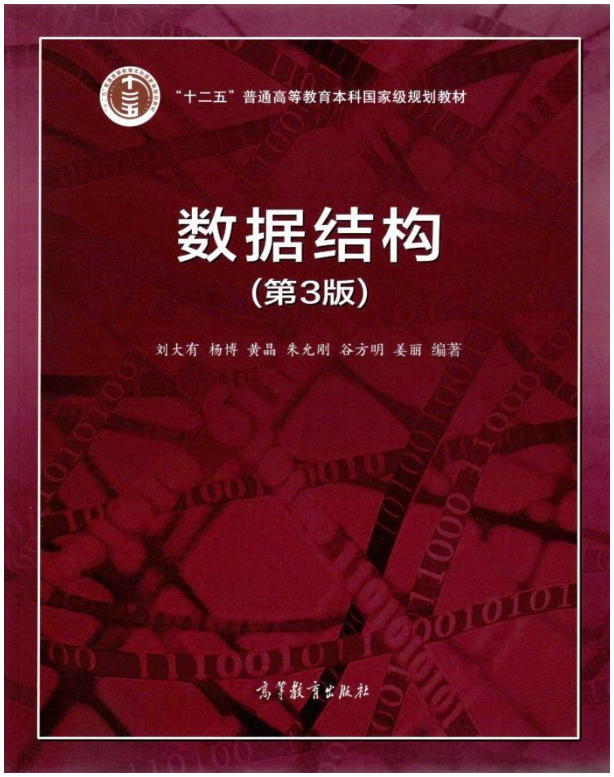
例2



红黑树的插入总结

情况	修正方法 x为新插入节点，p为x的父亲，g为x的爷爷，u为x的叔叔	
父亲黑色	无需修正	
父亲红色（双红）	黑叔叔	根据g到x的路径LL、RR、LR、RL四种情况，执行最多2次旋转，旋转后根染黑，根的孩子染红
	红叔叔	p、u染黑，g染红（若g为根则染黑），将g看作新插结点，继续向上处理

最多涉及2次旋转，O(log*n*)次染色



红黑树

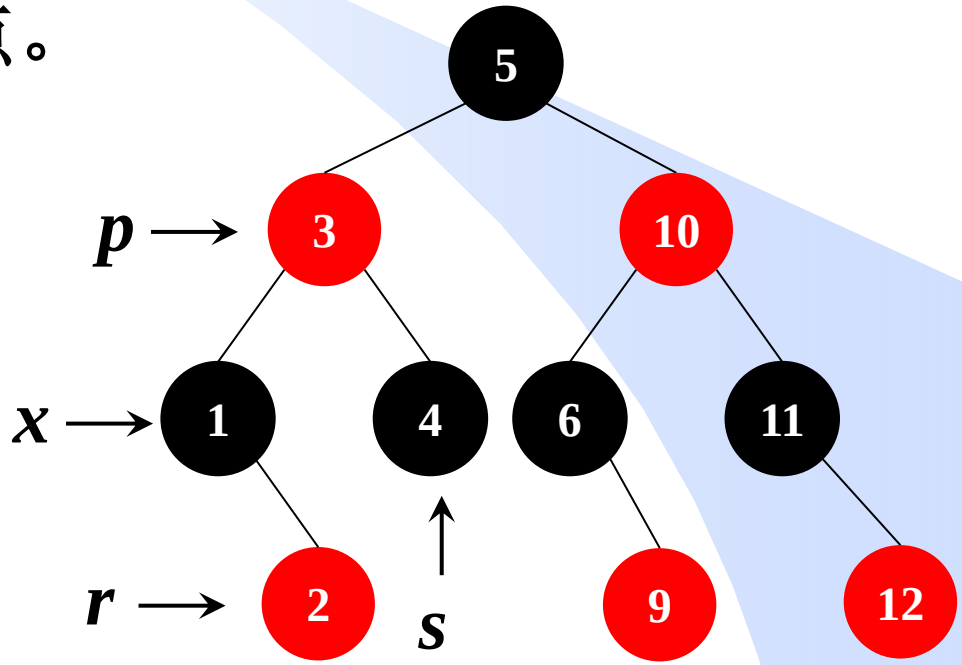
- 概述
- 定义与性质
- 插入算法
- **删除算法**
- 总结

朱允刚
zhuyungang@jlu.edu.cn

红黑树的删除

- 查找，若查找失败则直接返回，若查找成功则删除对应的结点*x*。
- 删除操作最终可归结为两种情况：删除叶结点和删除只有一个孩子的结点。

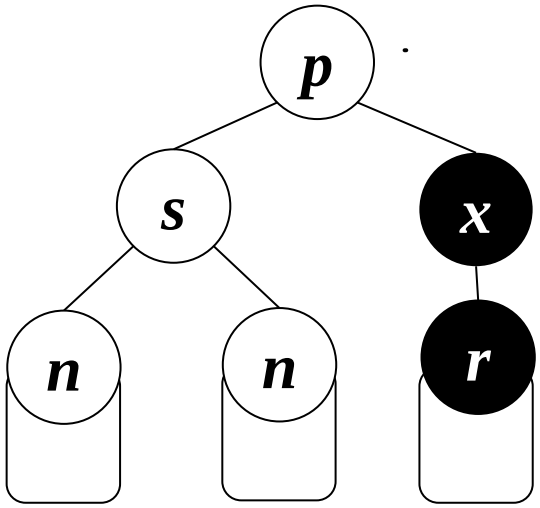
- ✓若*x*为红，则必为红叶子，直接删除。
- ✓若*x*为黑*r*为红，*r*替换*x*，并将*r*染黑。
- ✓若*x*为黑*r*为黑：双黑缺陷…



*x*为实际删除结点，*r*为替换*x*的结点，*p*为*x*的父亲，*s*为*x*的兄弟。

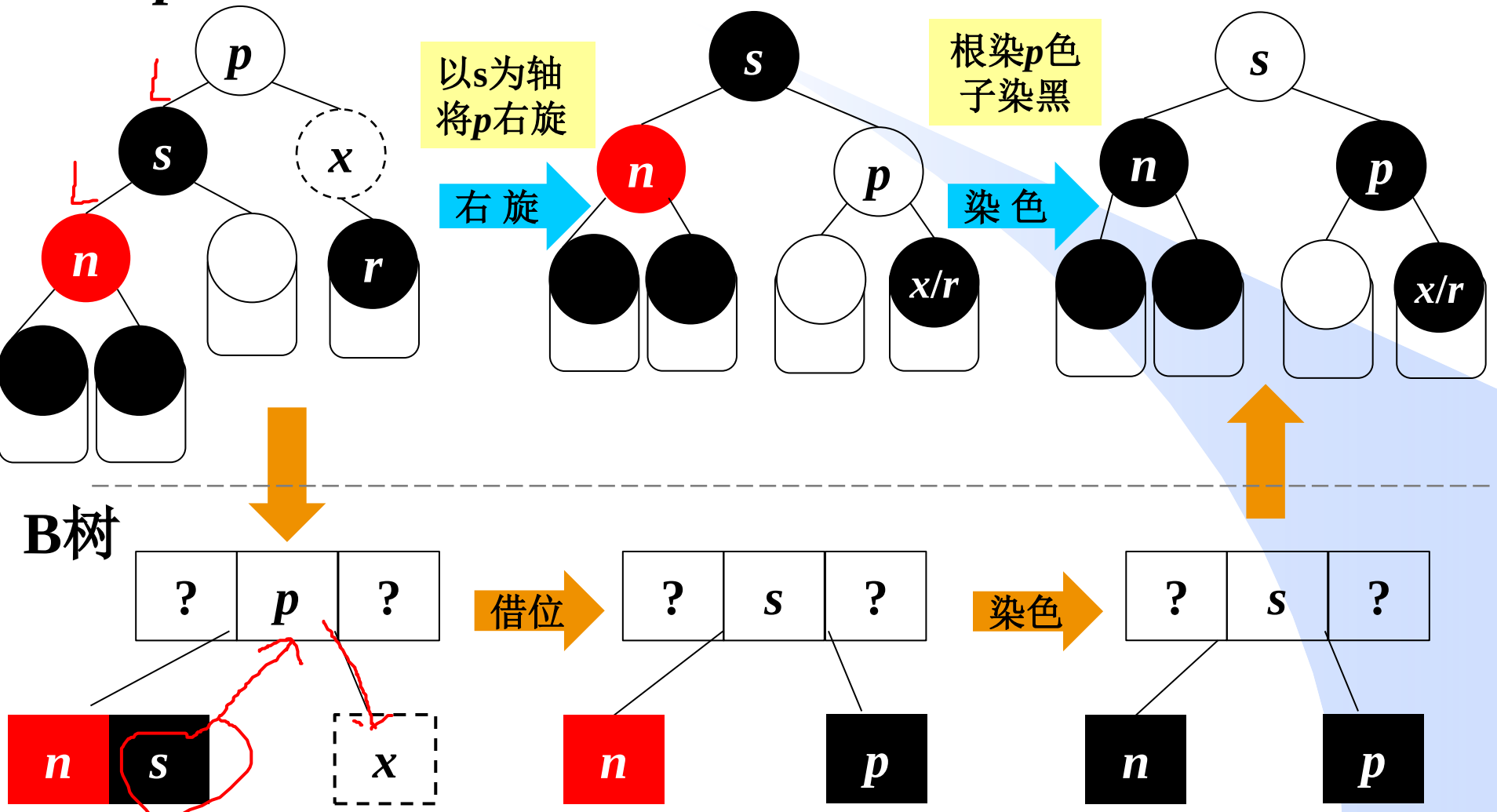
双黑缺陷

- 若*x*为黑*r*为黑（*r*可能为外结点）：双黑缺陷...
- *x*为实际删除结点，*r*为替换*x*的结点，*p*为*x*的父亲，*s*为*x*的兄弟，*n*为*s*的孩子。

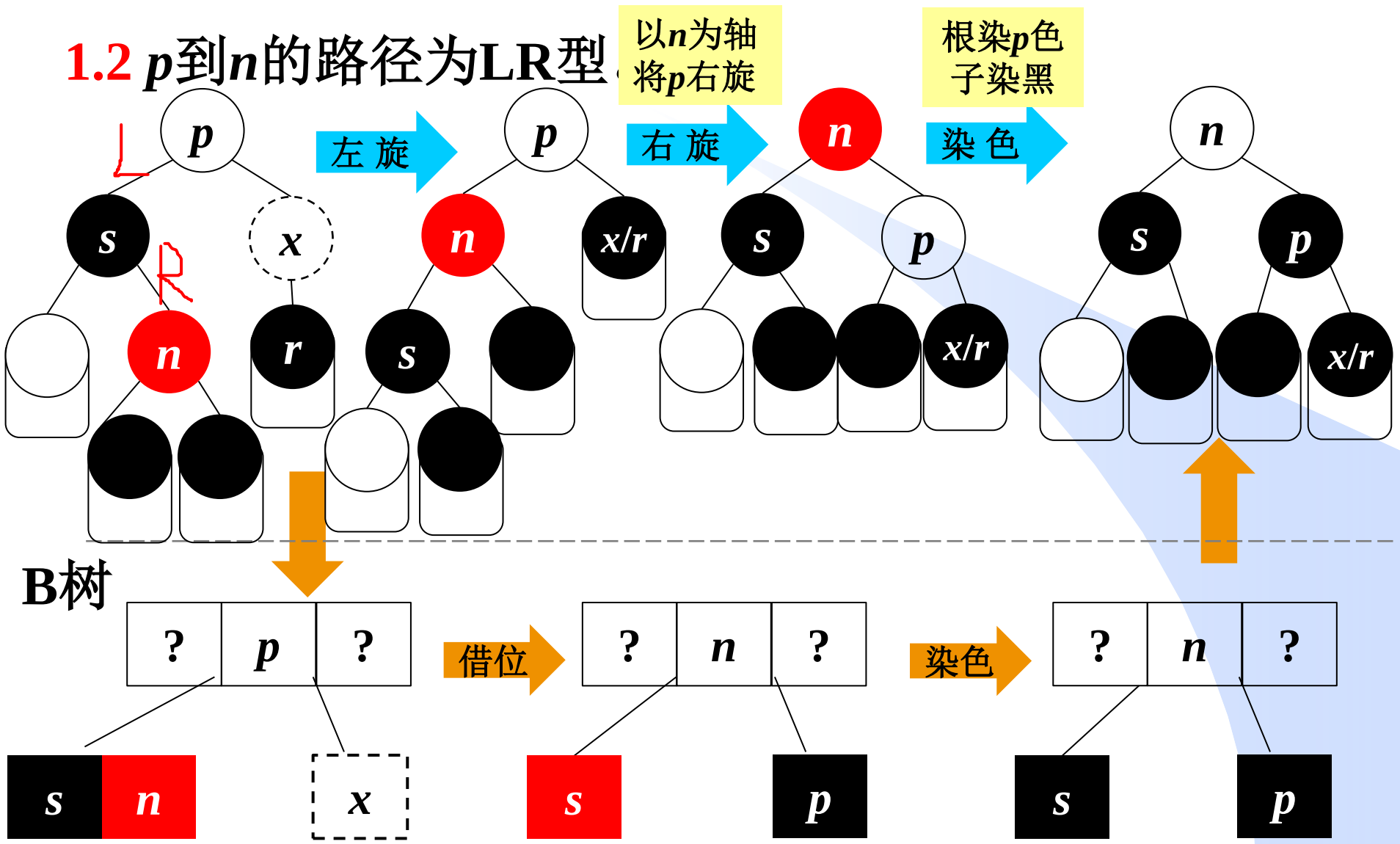


双黑修正—情况1：兄弟s为黑，且有红孩子

1.1 *p*到*n*的路径为LL型。

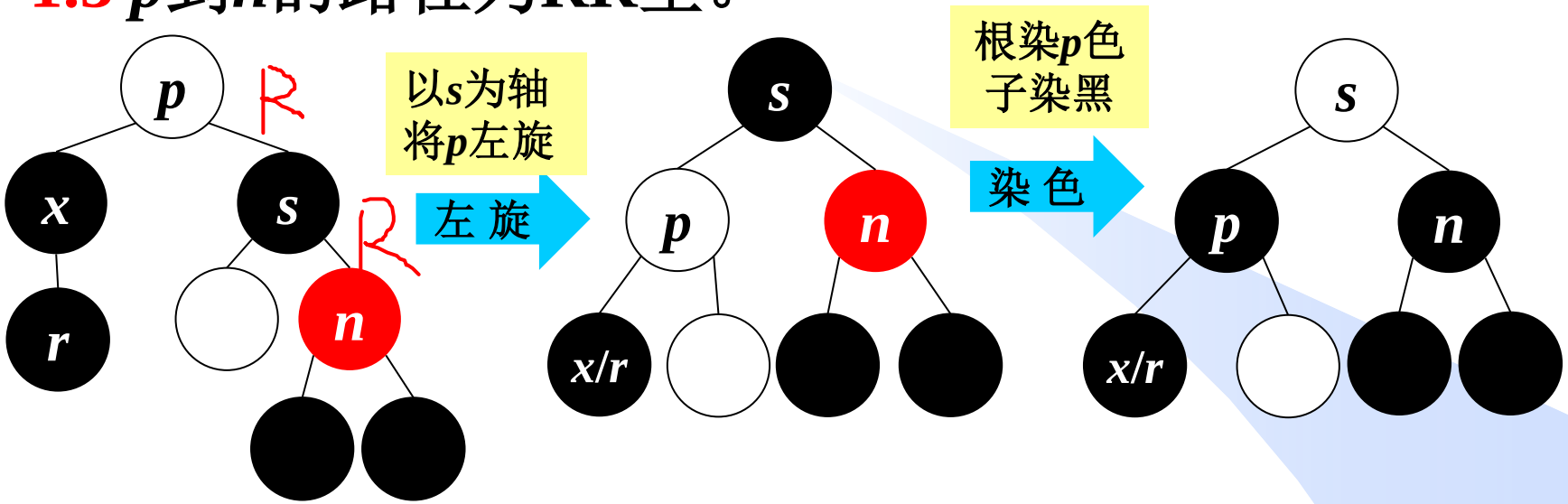


双黑修正—情况1：兄弟s为黑，且有红孩子

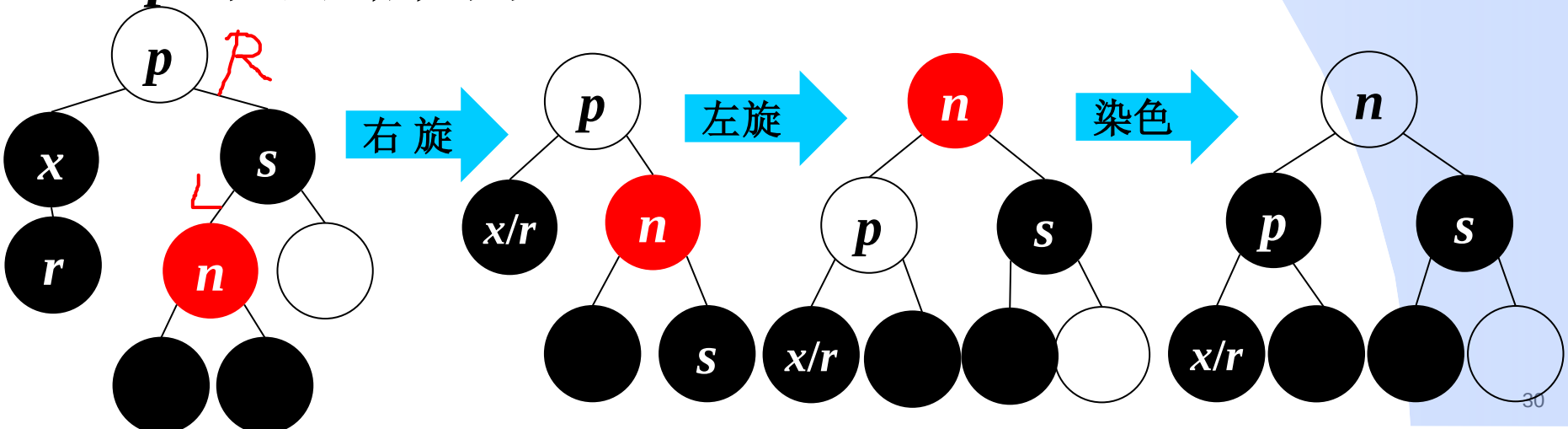


双黑修正—情况1：兄弟s为黑，且有红孩子

1.3 *p*到*n*的路径为RR型。

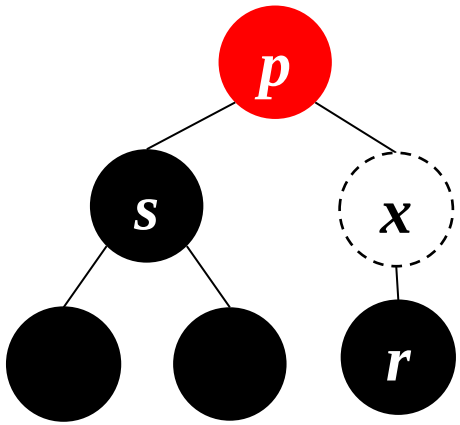


1.4 *p*到*n*的路径为RL型。



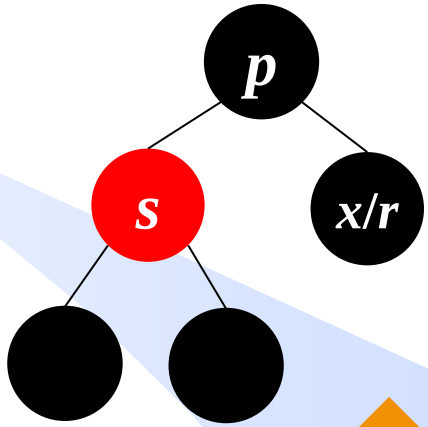
双黑修正—情况2：兄弟s为黑，无红孩子

2.1 父亲*p*为红

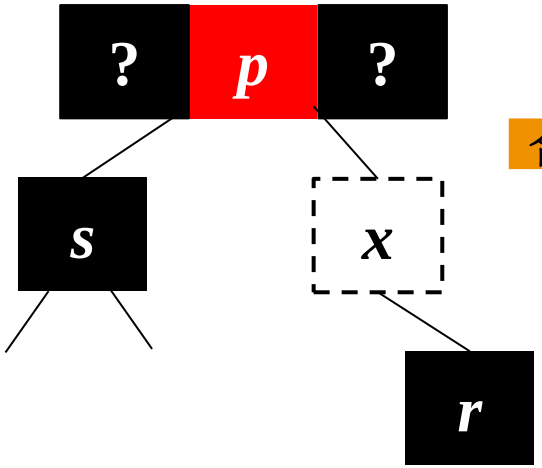


兄染红
父染黑

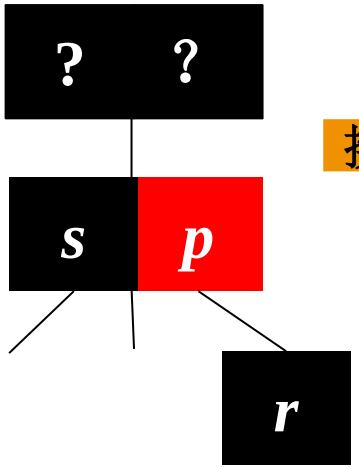
染 色



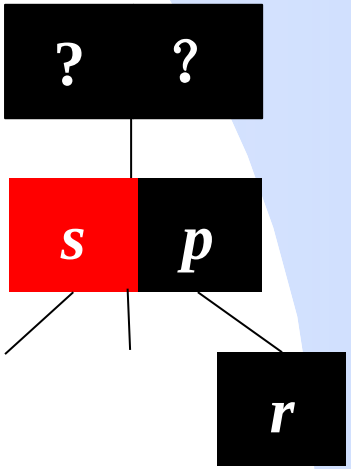
B树



合并



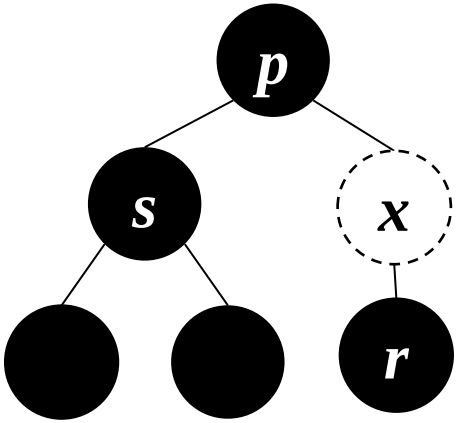
换色



双黑修正—情况2：兄弟s为黑，无红孩子

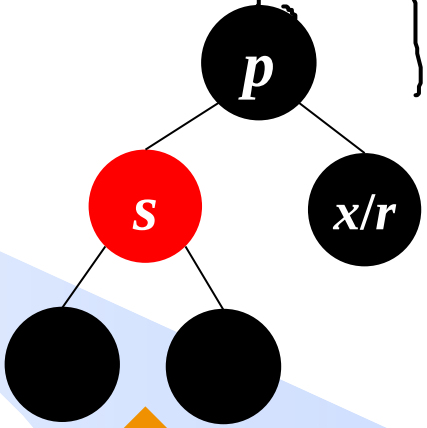
2.2 父亲p为黑

*p*黑高度减1，可能造成*p*父亲黑高度不相等，从*p*开始继续向上双黑修正

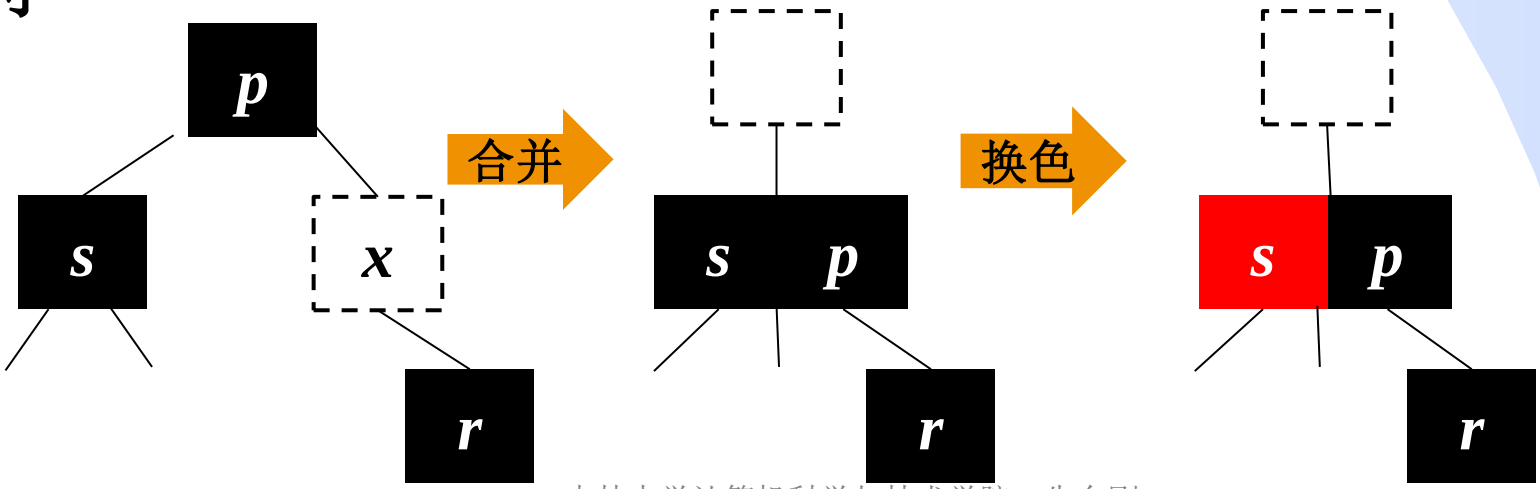


兄染红

染 色

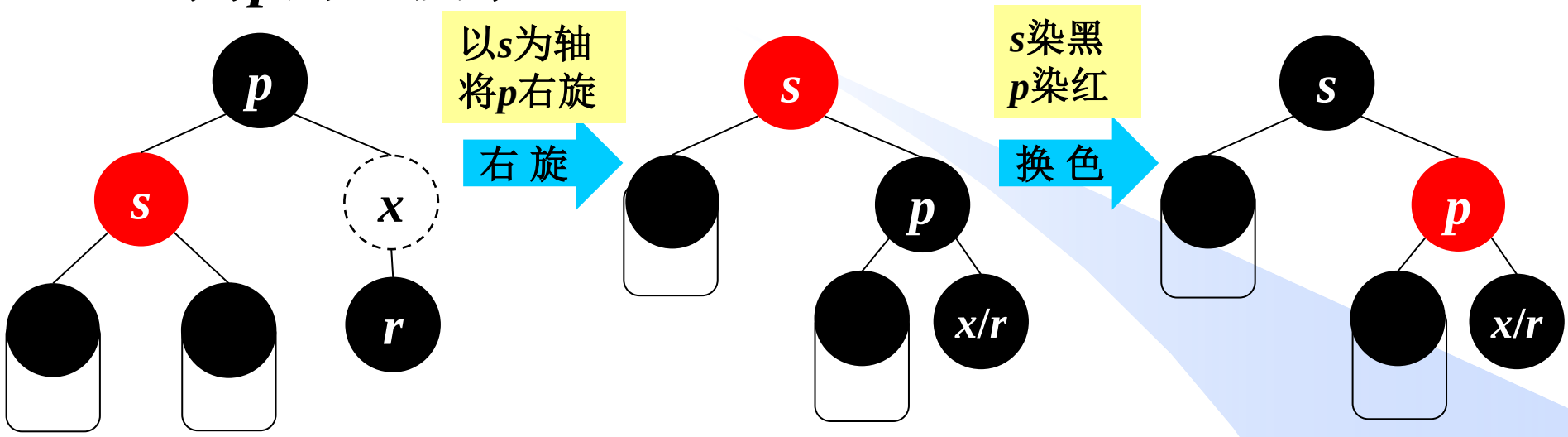


B树



双黑修正—情况3：兄弟s为红

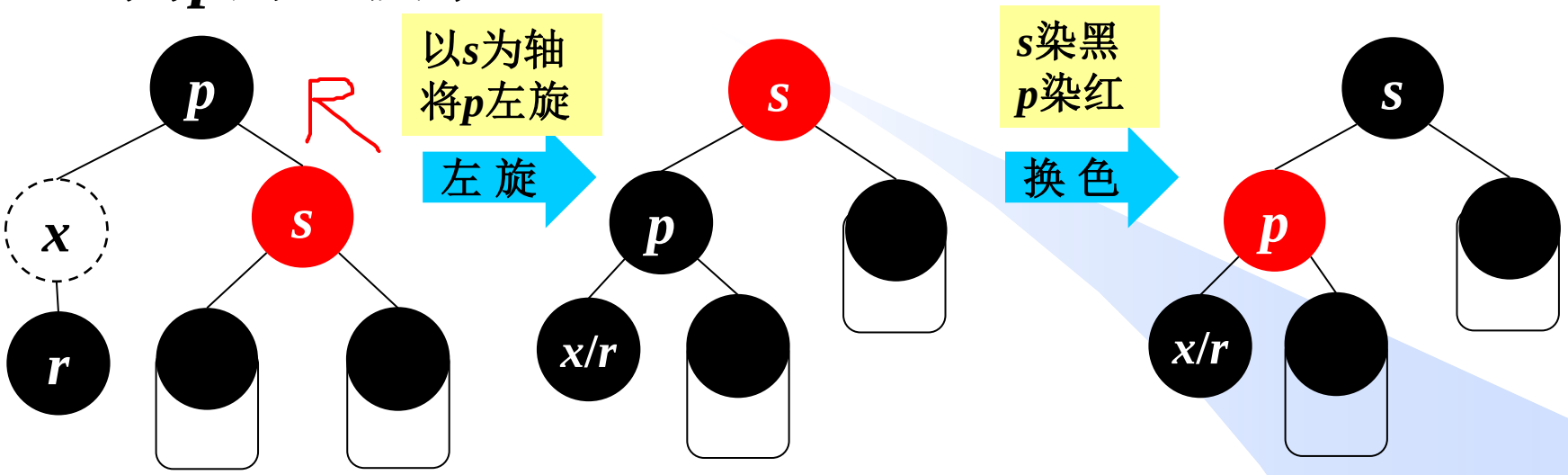
3.1 s为p的左孩子



经过变换后，问题没有解决，但*x/r*的兄弟变为黑色，可能转为黑兄弟有红孩子情况（最多需2次旋转），或黑兄弟无红孩子有红父亲情况，需染色。

双黑修正—情况3：兄弟s为红

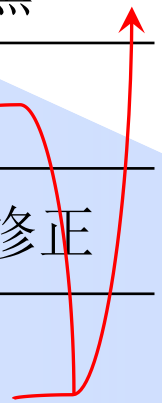
3.2 s为p的右孩子



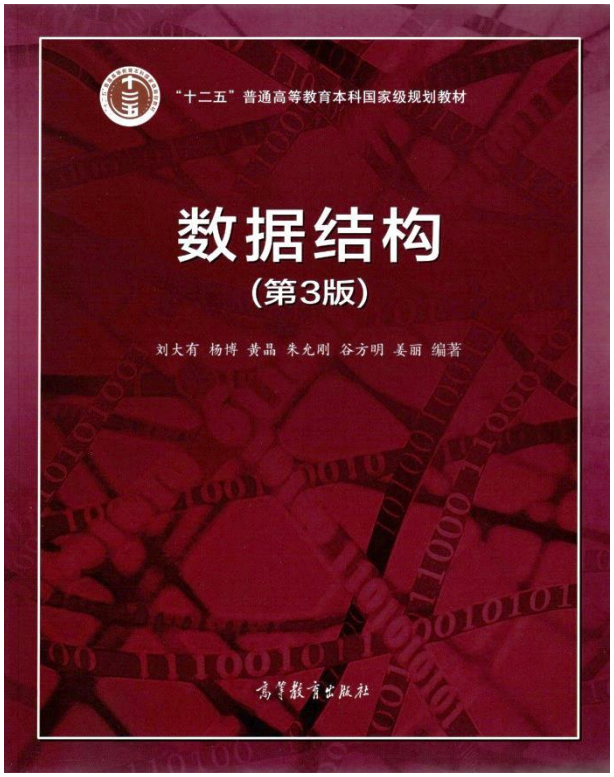
经过变换后，问题没有解决，但*x/r*的兄弟变为黑色，可能转为黑兄弟有红孩子情况（最多需2次旋转），或黑兄弟无红孩子有红父亲情况，需染色。

红黑树的删除总结

情况	修正方法 x为实际删除结点，r为替换x的结点，p为x的父亲，s为x的兄弟， n为s的孩子		
x红色	必为红叶子，直接删除，无需修正		
x黑r红	r替换x，r染黑		
x黑r黑 (双黑)	黑兄有红子	根据p到n的路径LL、RR、LR、RL情况，执行最多2次旋转后子树根染原p颜色，根的孩子染黑	
	黑兄无红子	红父	兄染红，父染黑
		黑父	兄染红，从子树根开始继续向上修正
	红兄	单旋后s染黑、p染红，转为“黑兄有红子”或“黑兄无红子有红父”情况。	



最多涉及3次旋转，O(log$n$)次染色



红黑树

- 概述
- 定义与性质
- 插入算法
- 删除算法
- 总结

朱允刚
zhuyungang@jlu.edu.cn

AVL树 vs 红黑树

- 查找、插入、删除最坏时间复杂度均为O(log*n*)。
- 红黑树平衡性弱于AVL树，故查找性能低于AVL树。
- 红黑树插入删除所需的旋转次数较少，插入、删除效率高于AVL树。