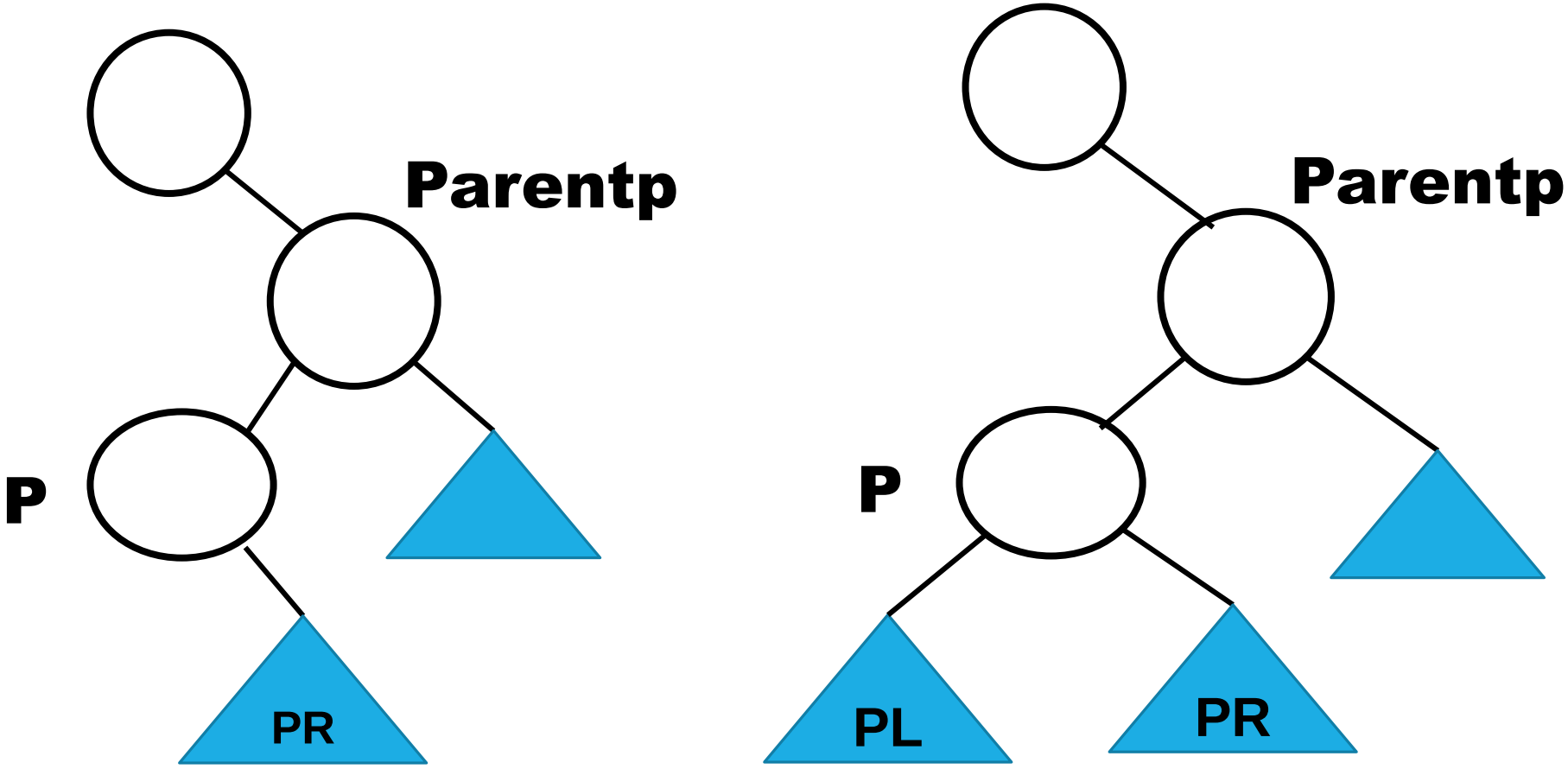
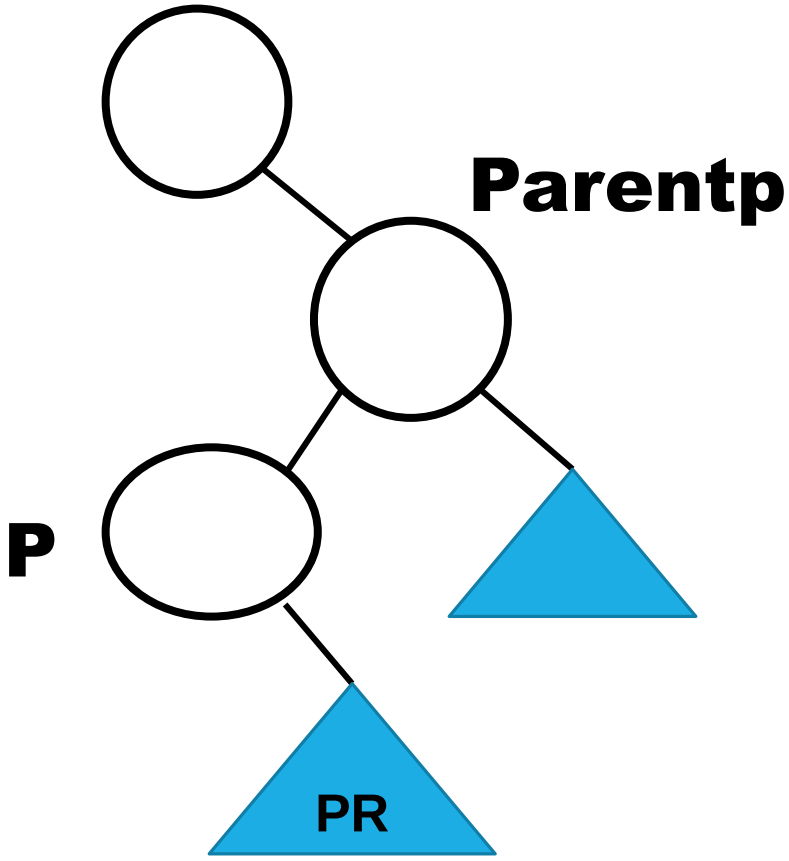
二叉排序树的删除　BST--delete

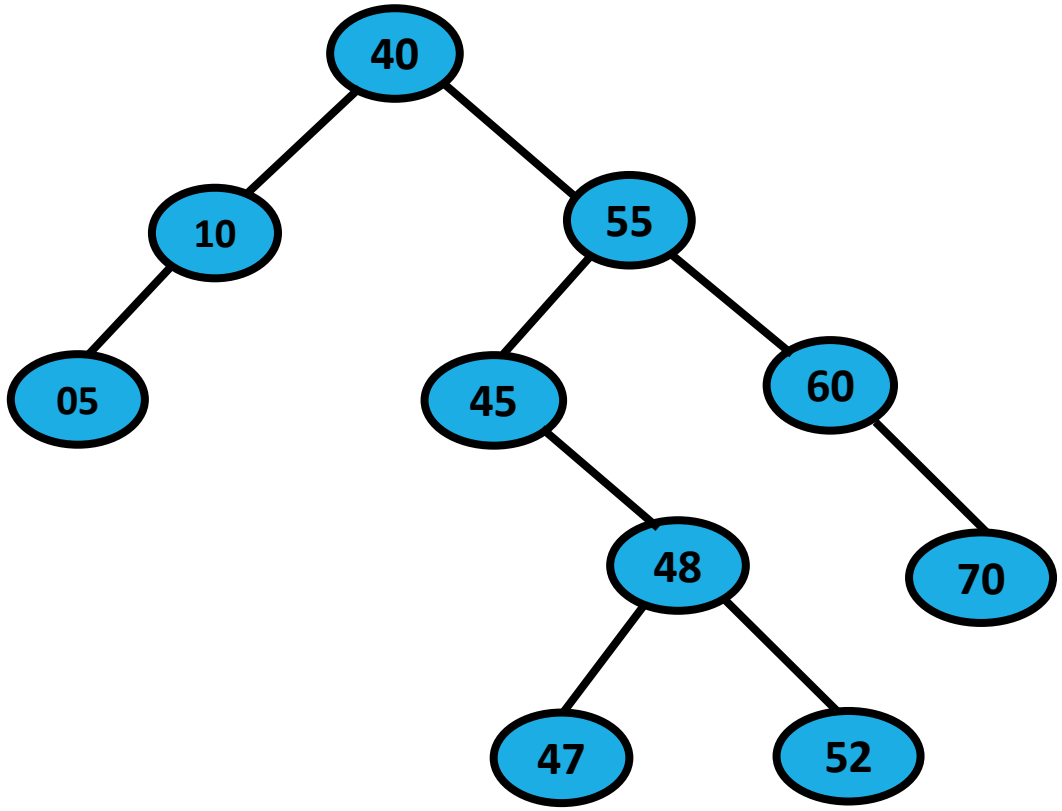


注意：删除二叉排序树的某个指定结点后，仍然要是二叉排序树

① 若p左子树PL为空，此时，只要令PR的根结点直接代替p即可

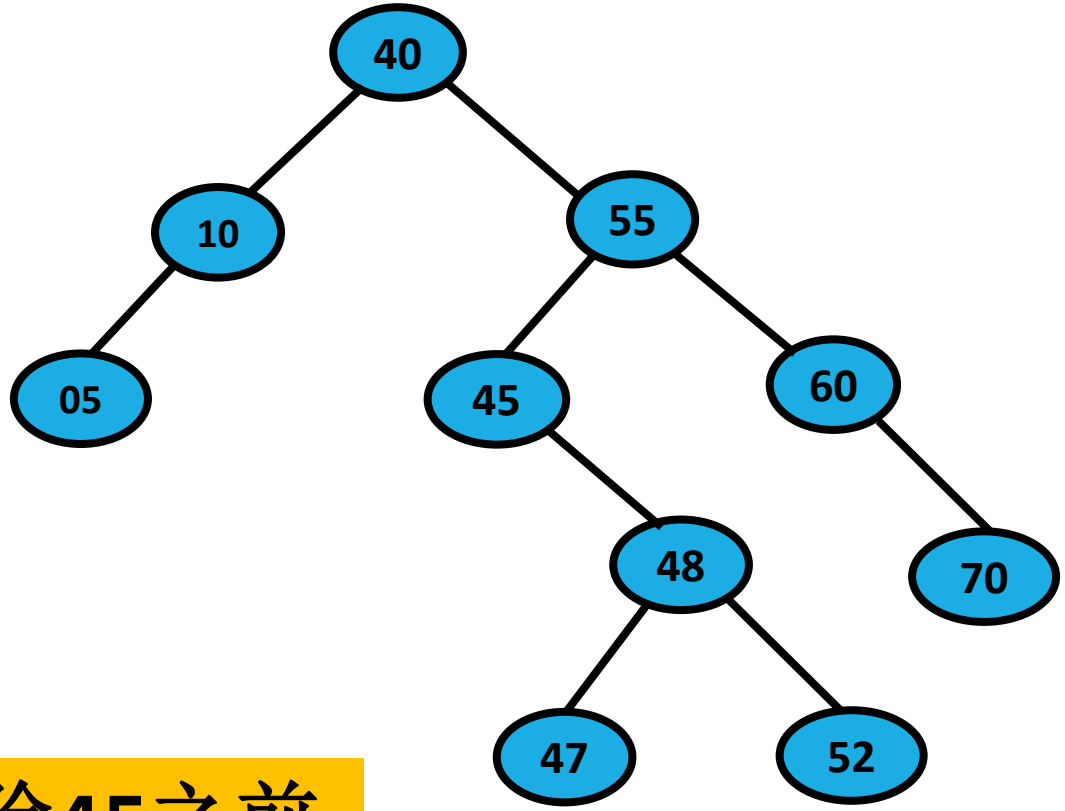
假设指针p指向要删除的结点，指针parentp指向p的父结点



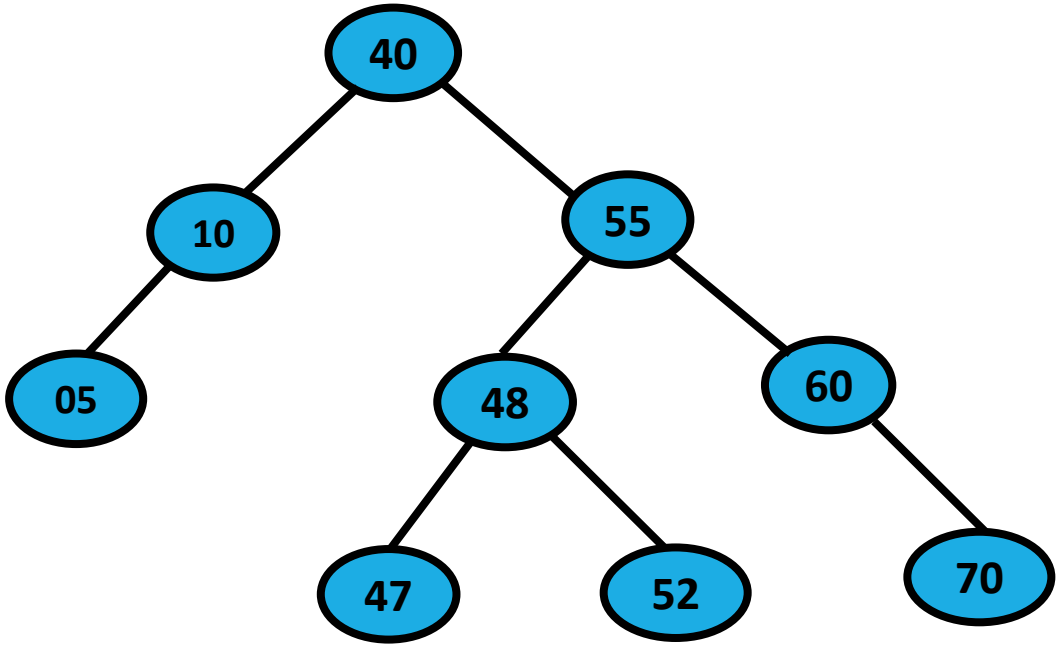


① 若*p左子树PL为空，此时，只要令PR的根结点直接代替*p即可

假设指针p指向要删除的结点，指针parentp指向*p的父结点

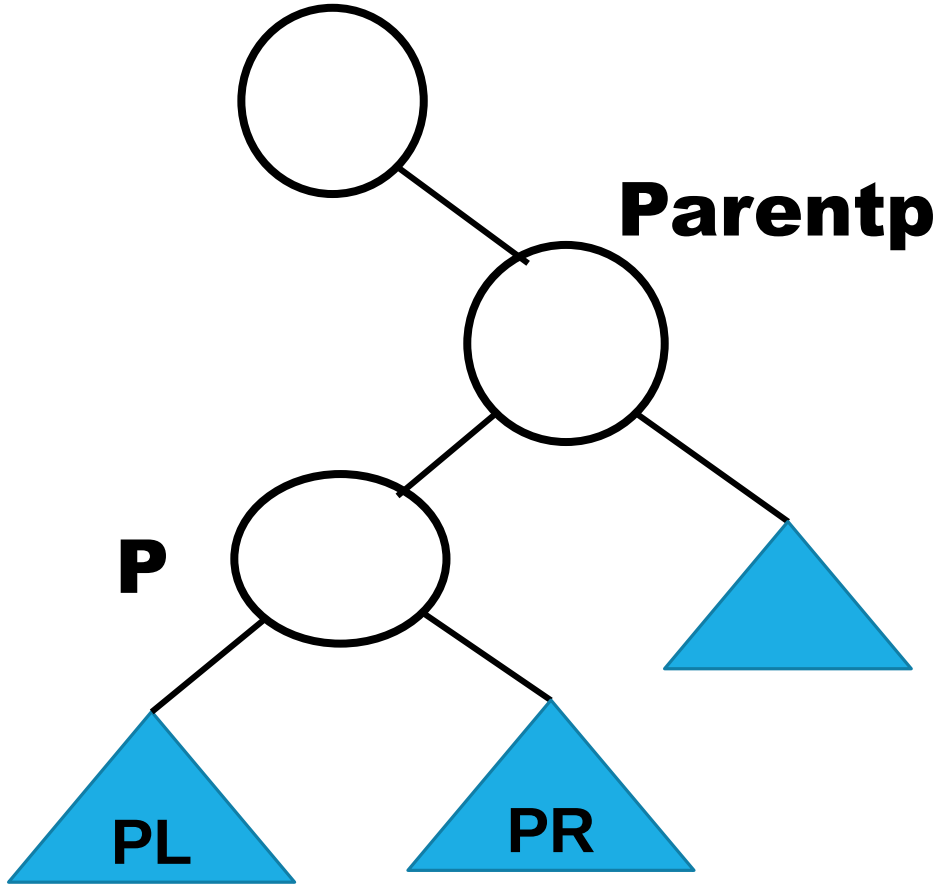


删除45之前



删除45之后

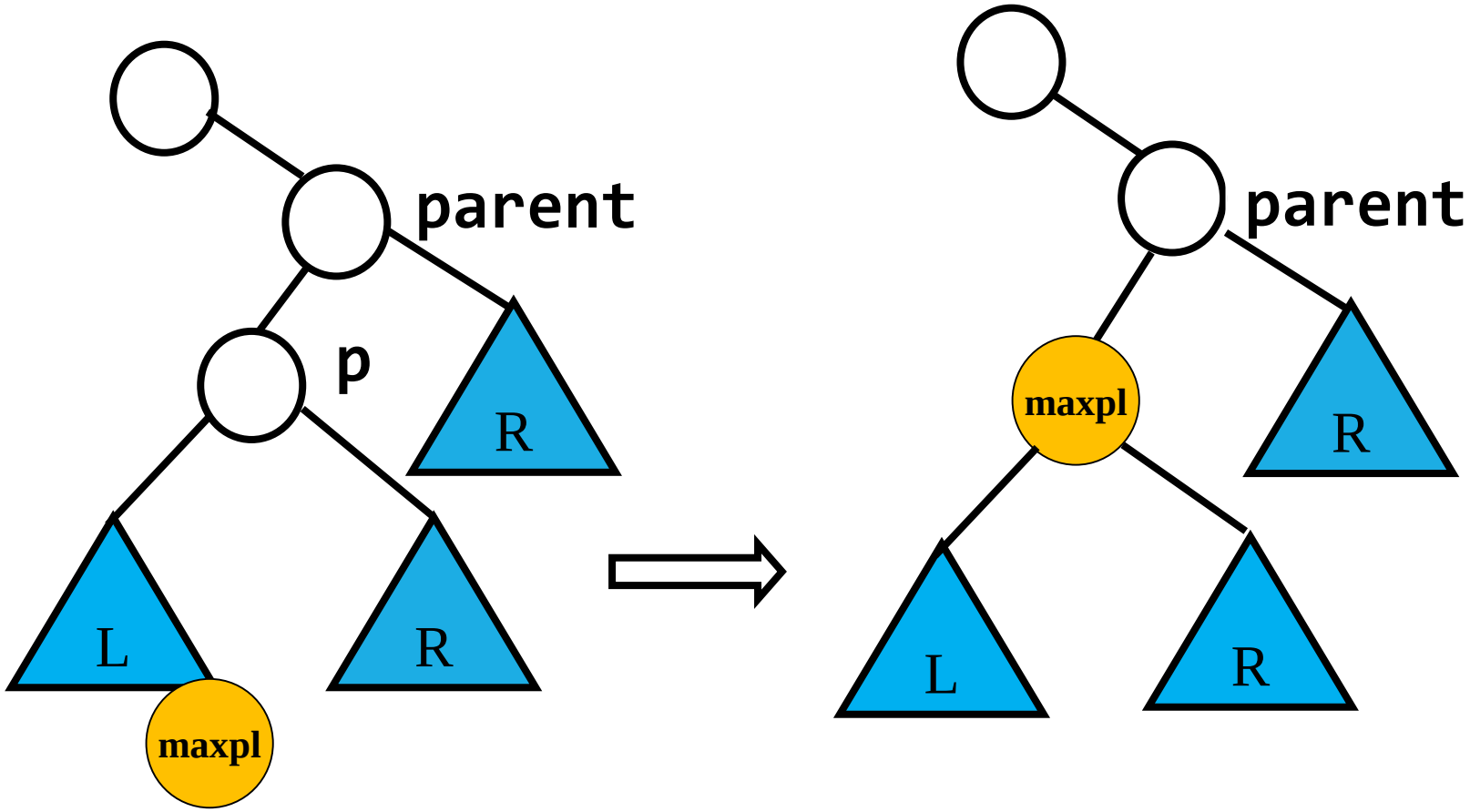
二叉排序树的删除　BST--delete



方法1：

（a）按对称序周游p的左子树，找到关键码最大的结点 maxpl，删除maxpl（用的左子女代替它）

（b）用 maxpl 结点代替被删除结点p

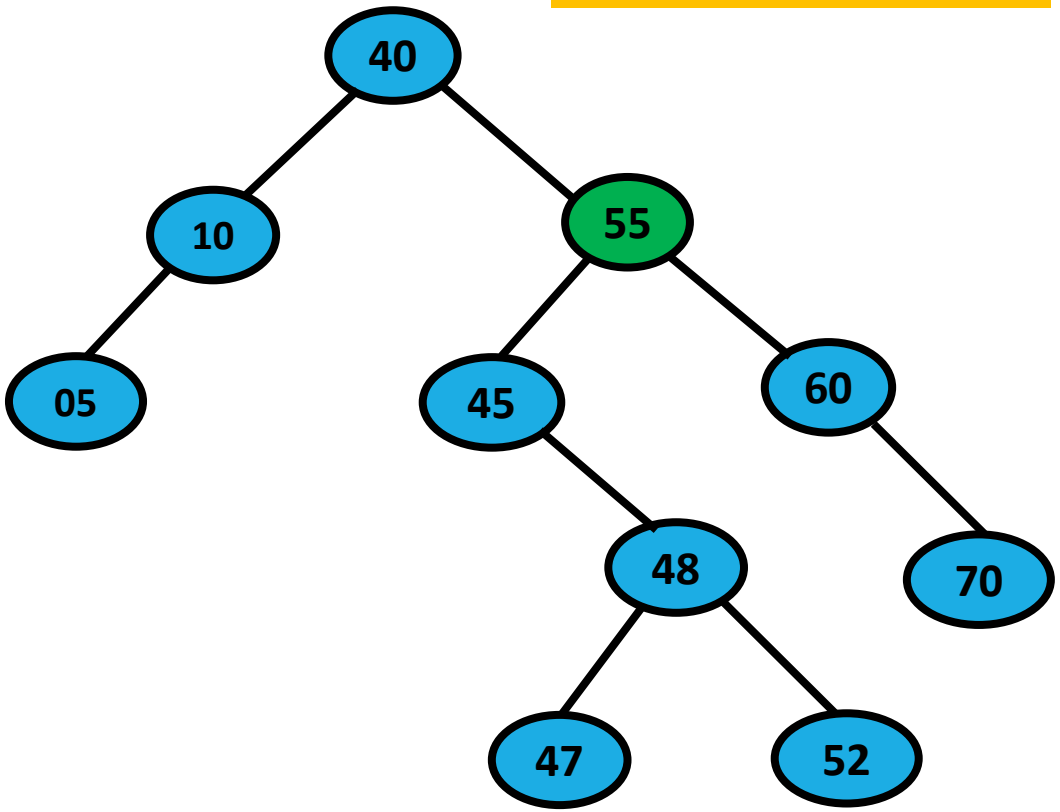


L maxpl p R

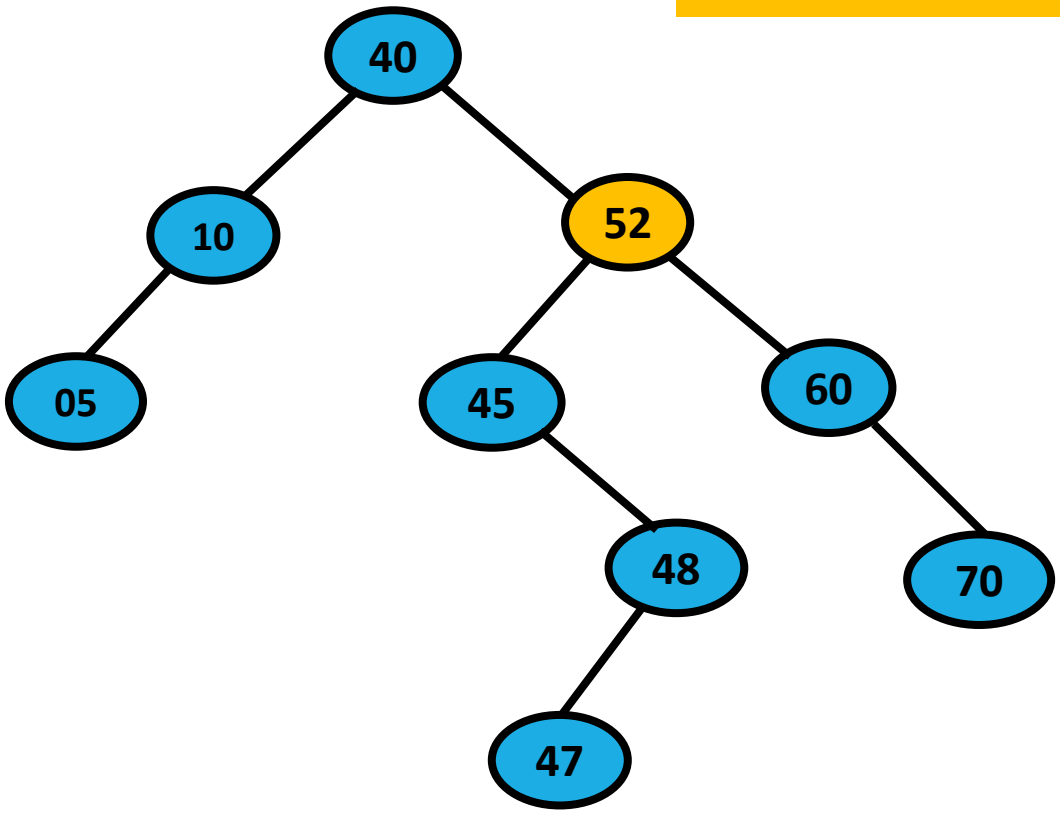
L maxpl R

5.5 二叉排序树的删除 BST--delete

删除55之前

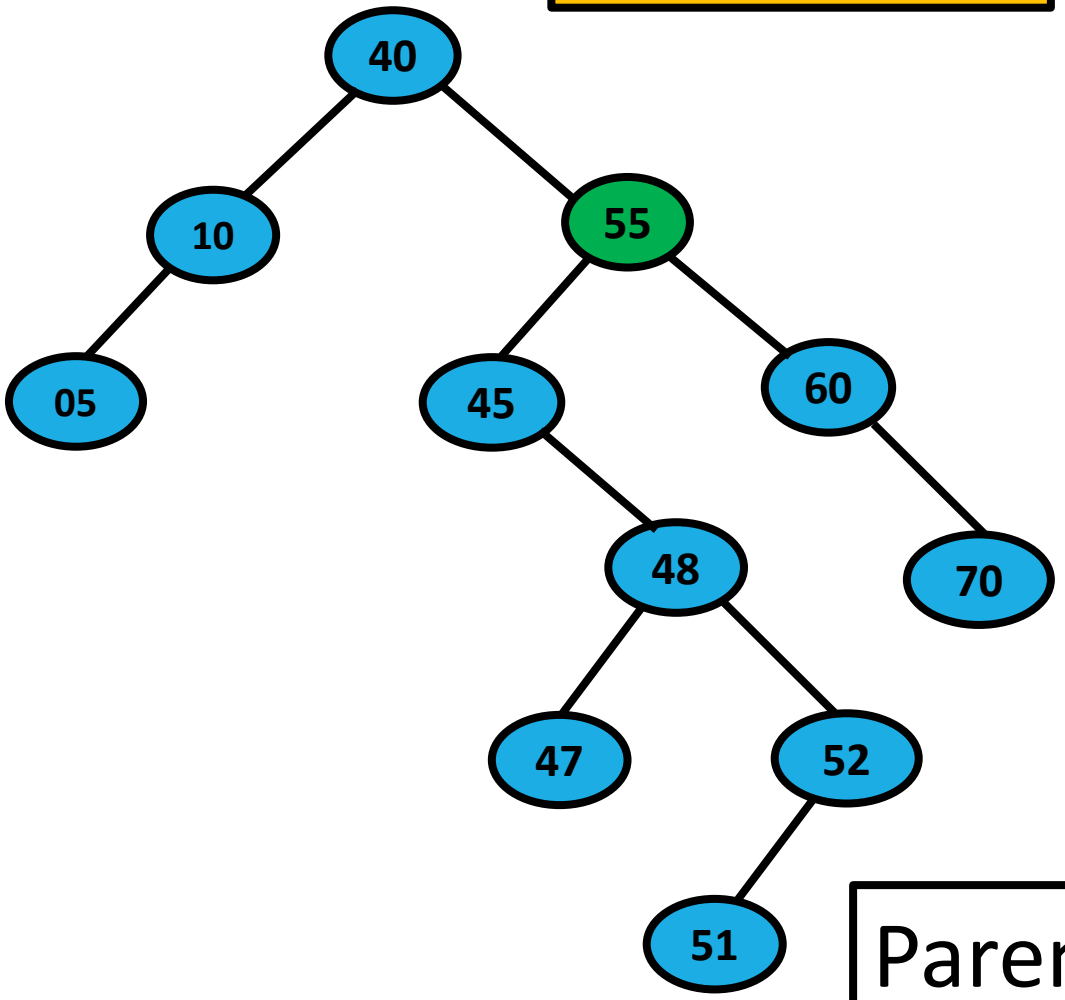


删除55之后

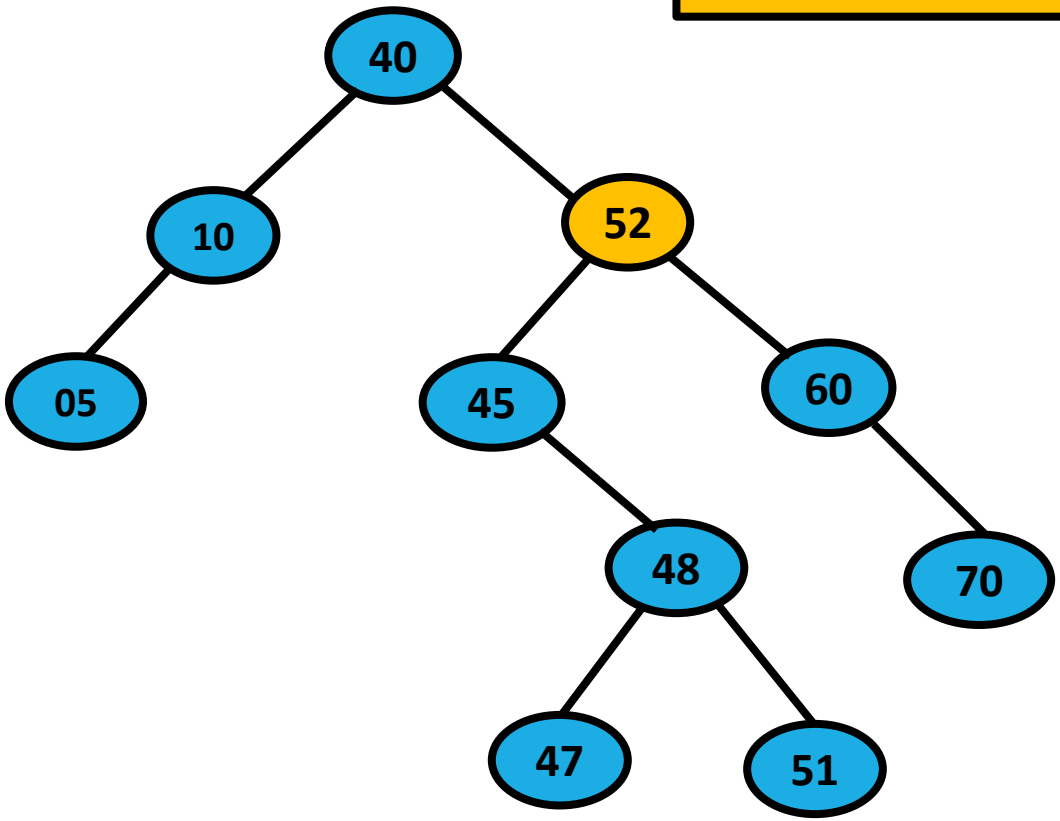


5.5 二叉排序树的删除 BST--delete

删除55之前



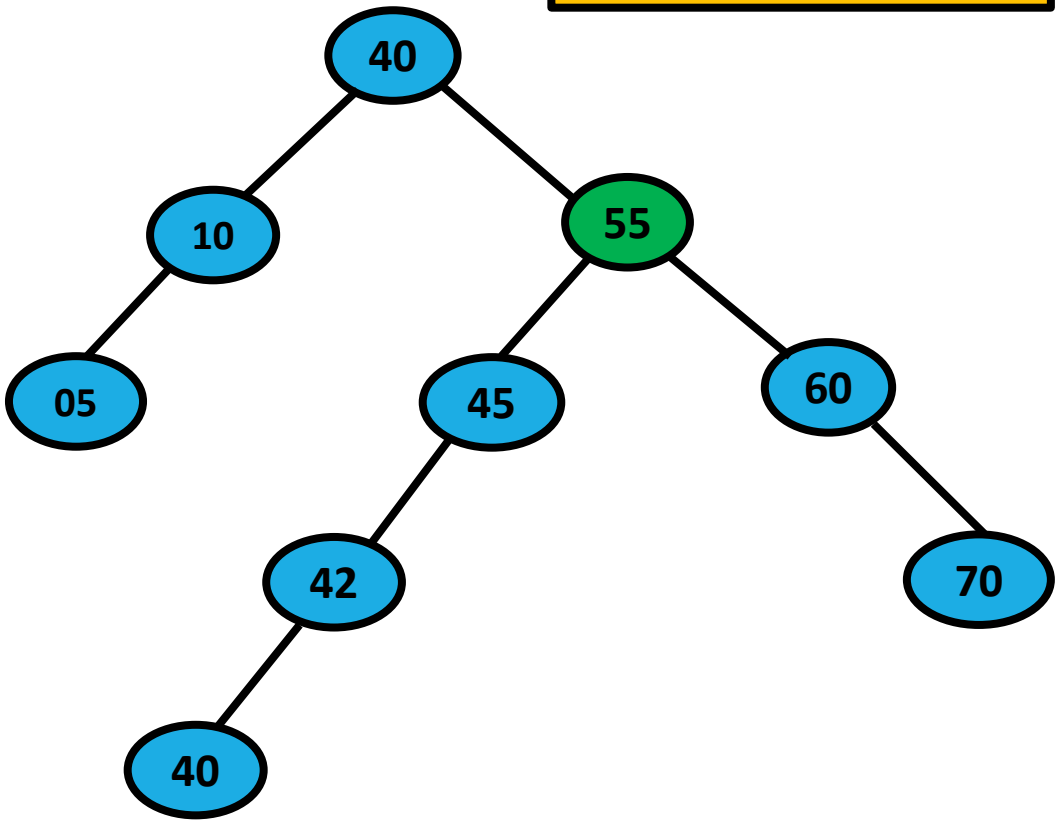
删除55之后



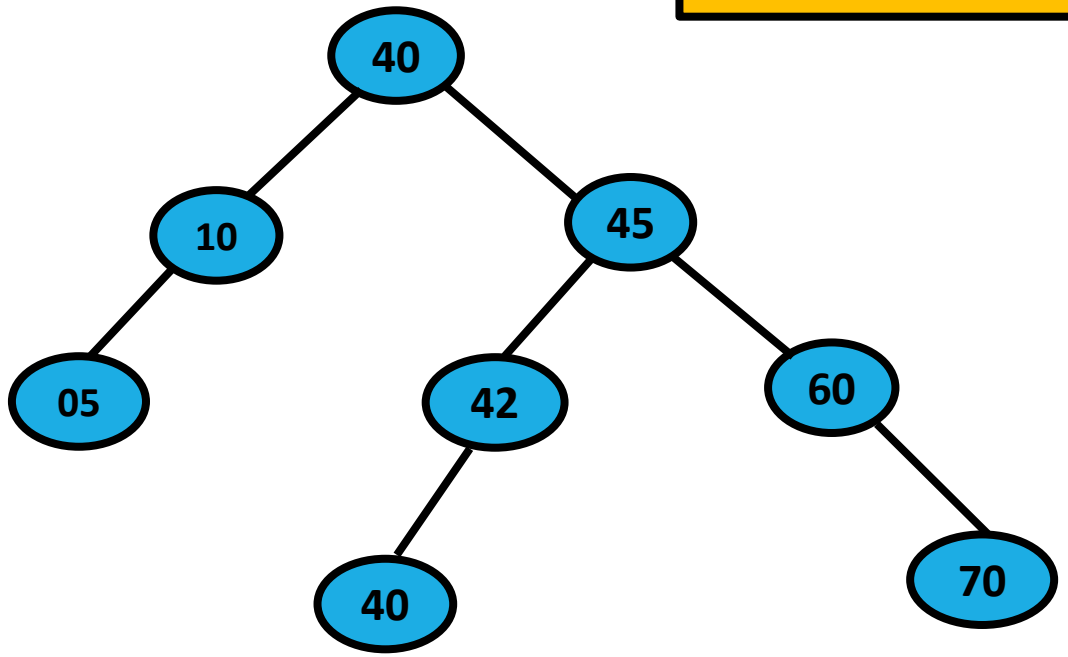
Parentp记录maxpl的父结点parentp!=p

5.5 二叉排序树的删除 BST--delete

删除55之前



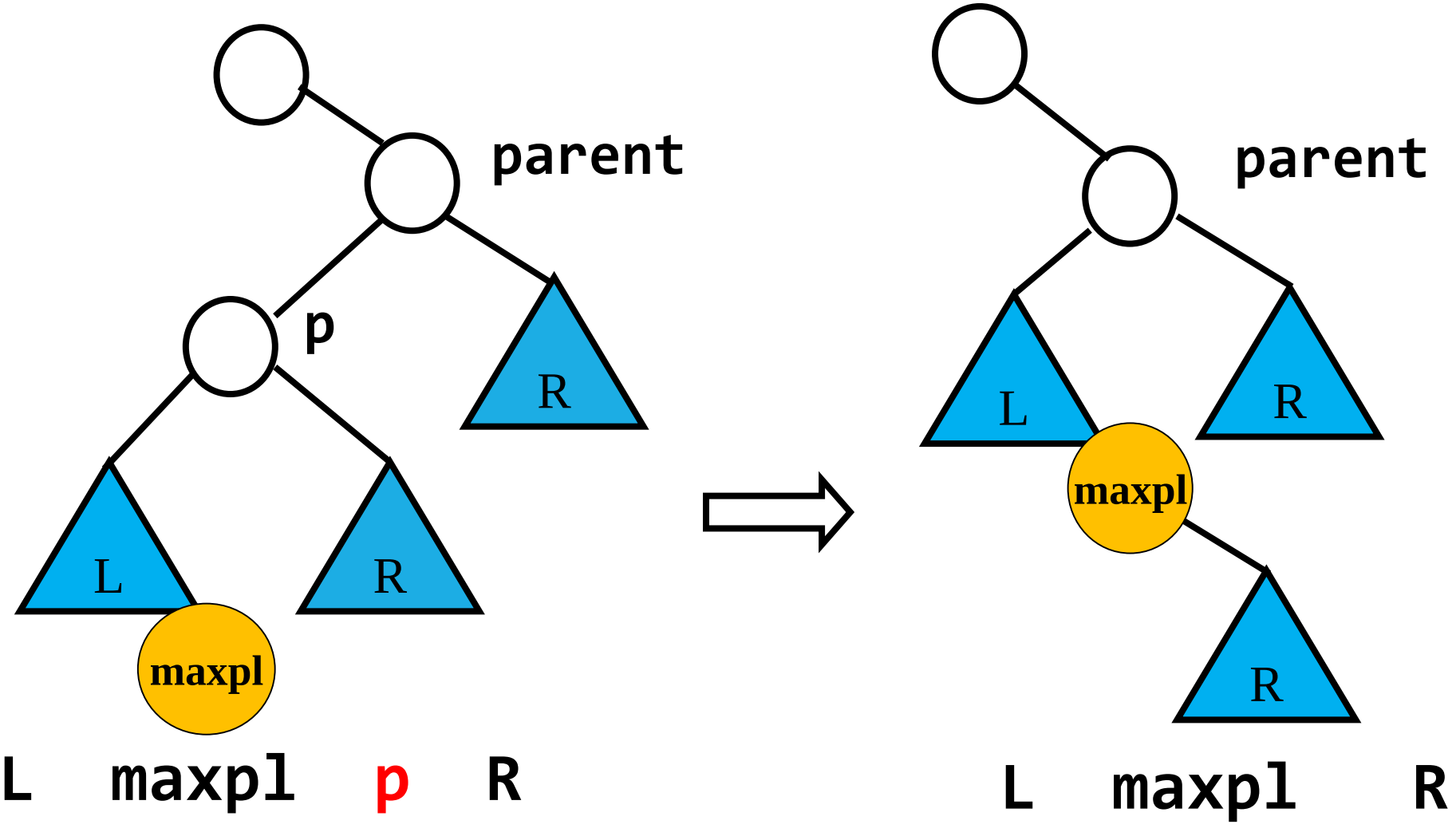
删除55之后



Parentp记录maxpl的父节点，parentp==p

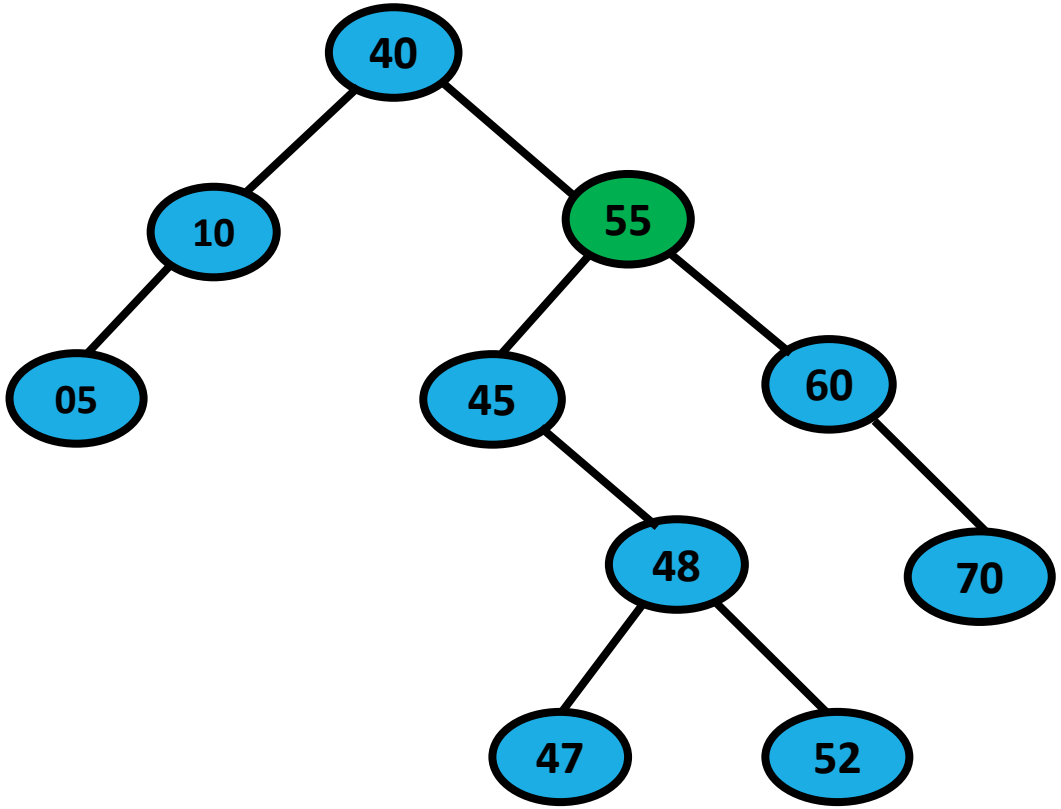
方法2：令p的左子树的根结点代替p，maxpl的右子树是p的右子树

maxpl (p左子树中最大的结点)；

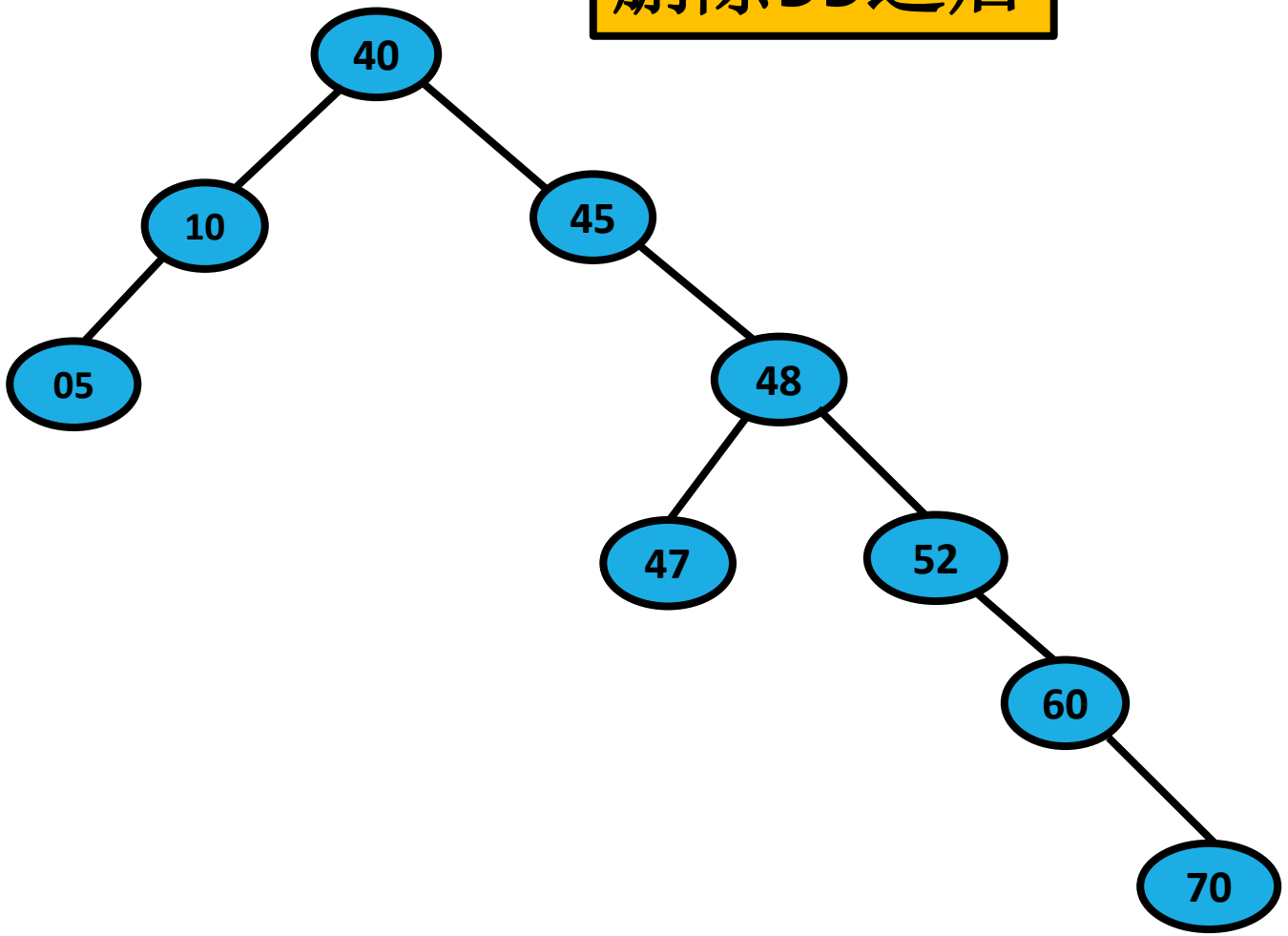


5.5 二叉排序树的删除　BST--delete

删除55之前



删除55之后



查找被删除结点p;

if(p无左子女)

用*p的右子女代替p;

else {

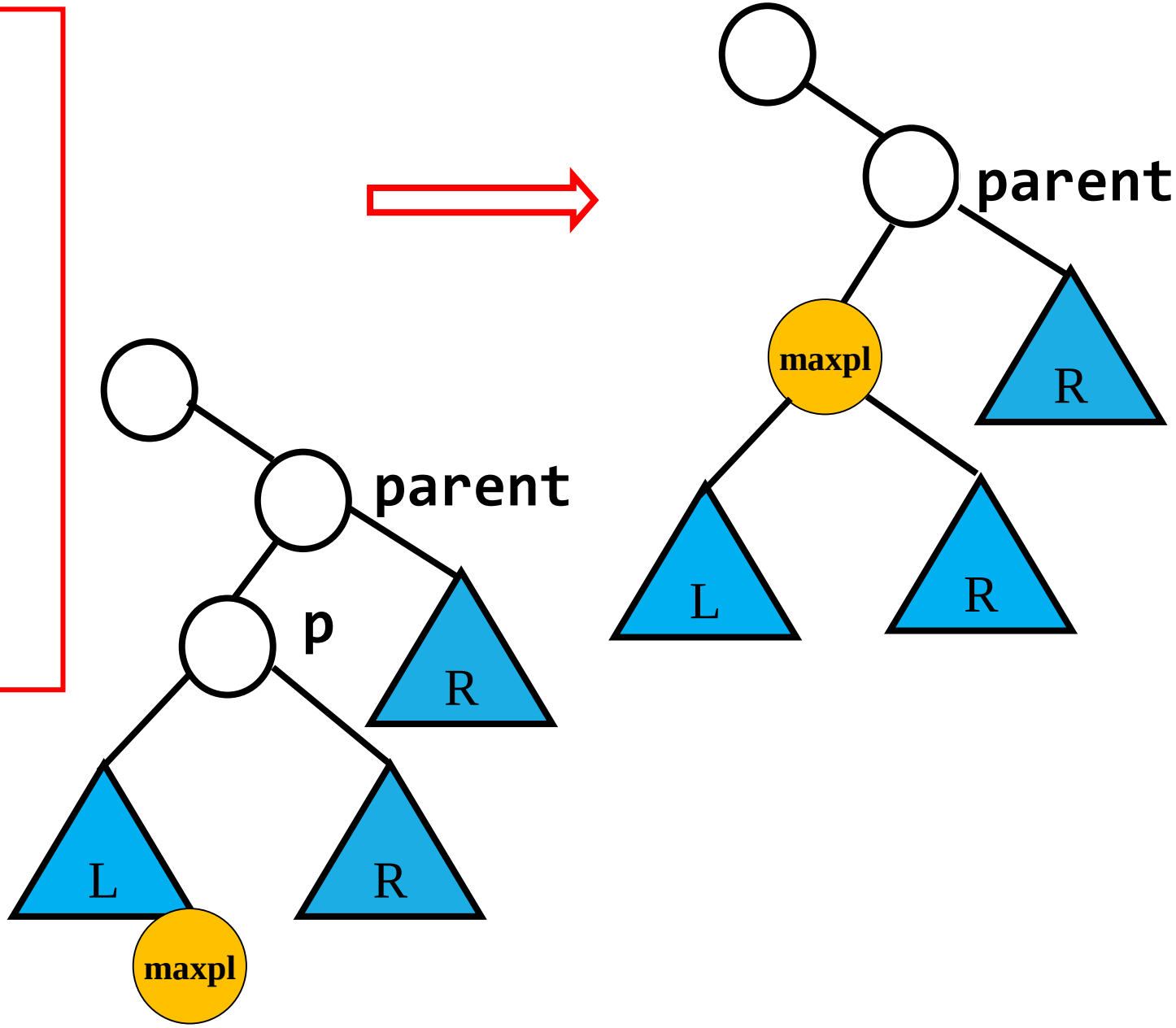
找p的左子树中最右下结点maxpl;

用maxpl结点代替被删除的结点p;

用原来maxpl的左子女代替maxpl;

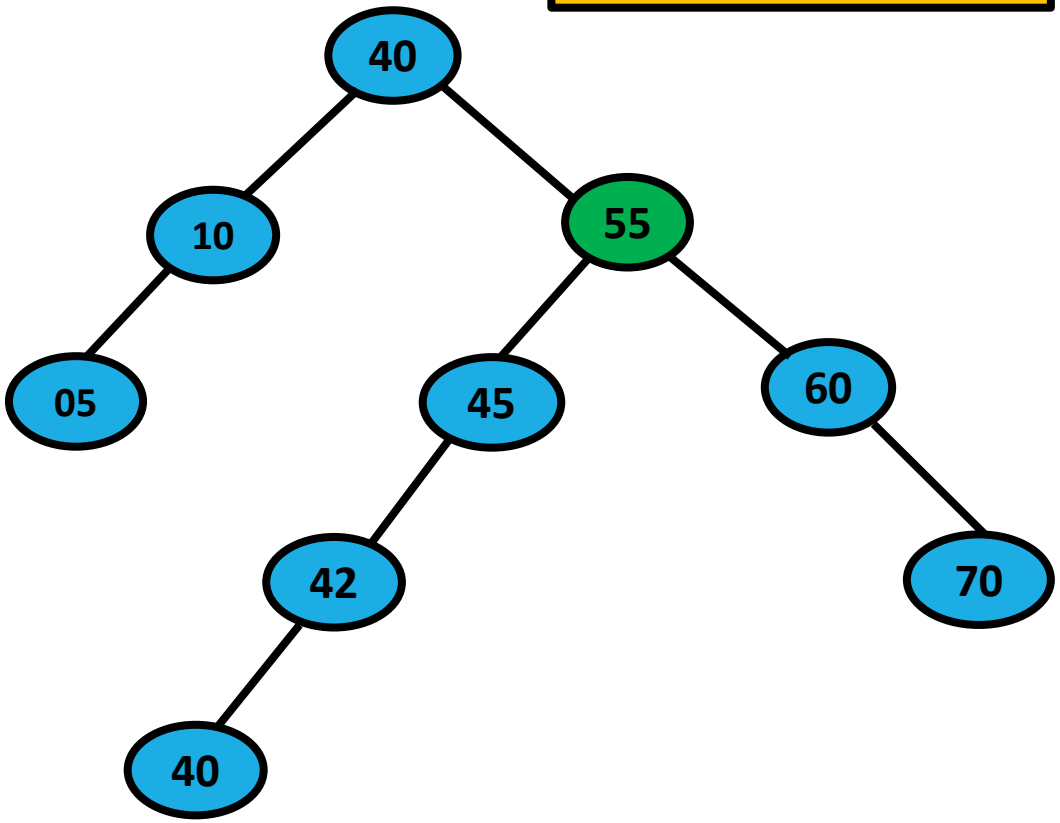
}

具体见算法5-4

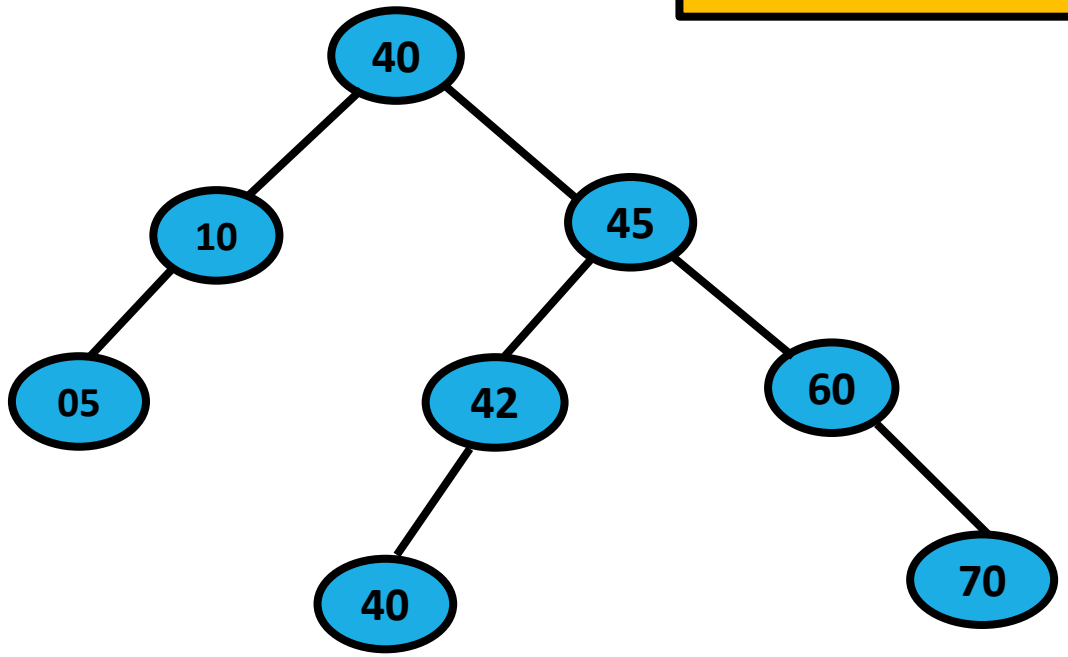


5.5 二叉排序树的删除 BST--delete

删除55之前



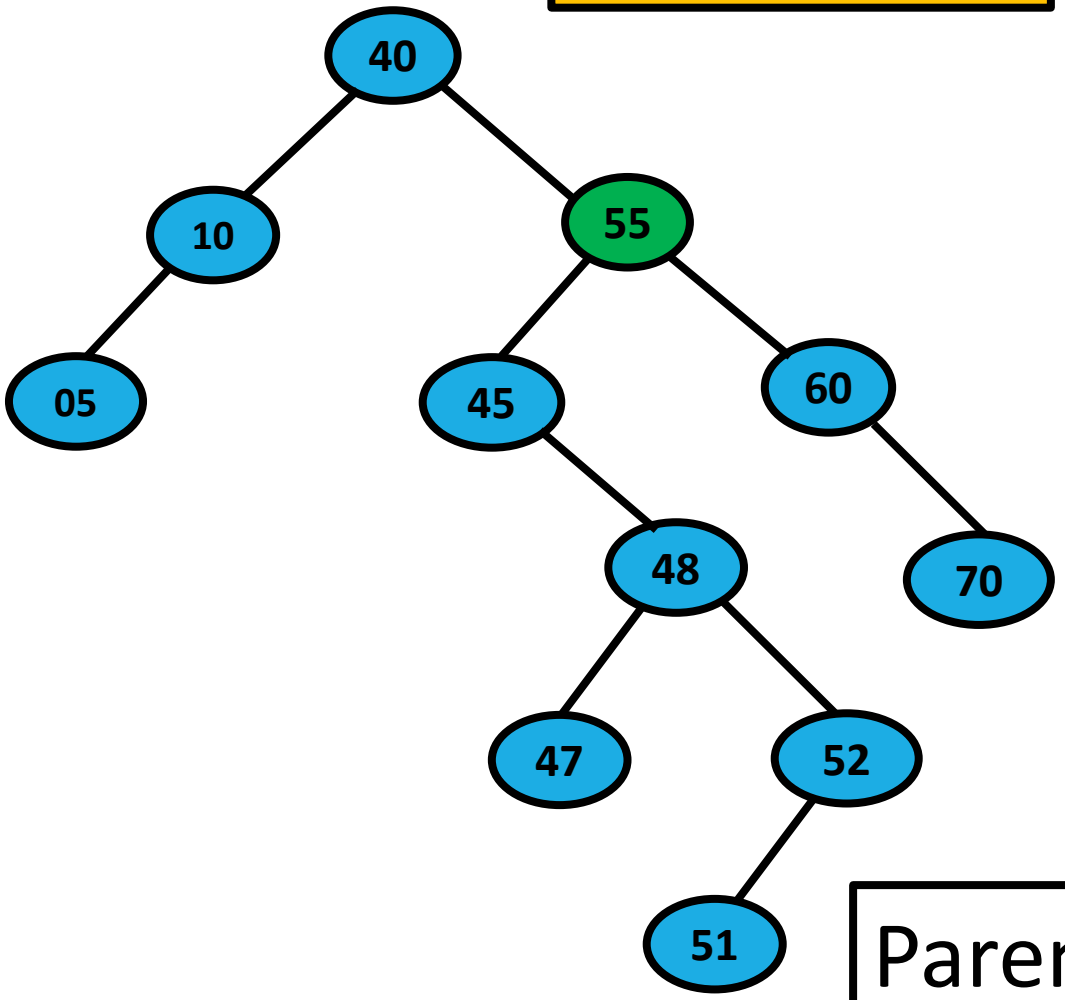
删除55之后



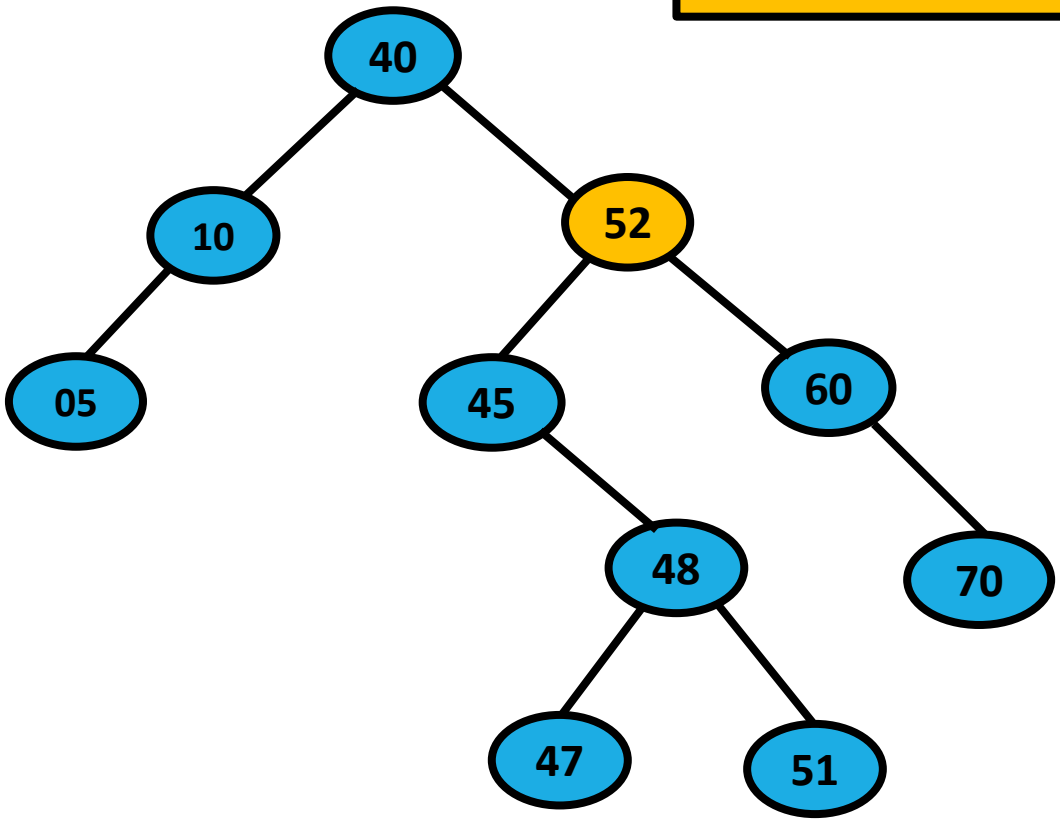
Parentp记录maxpl的父节点，parentp==p

5.5 二叉排序树的删除 BST--delete

删除55之前



删除55之后



Parentp记录maxpl的父结点parentp!=p

查找被删除结点p；

if(p无左子女)

用p的右子女代替p；

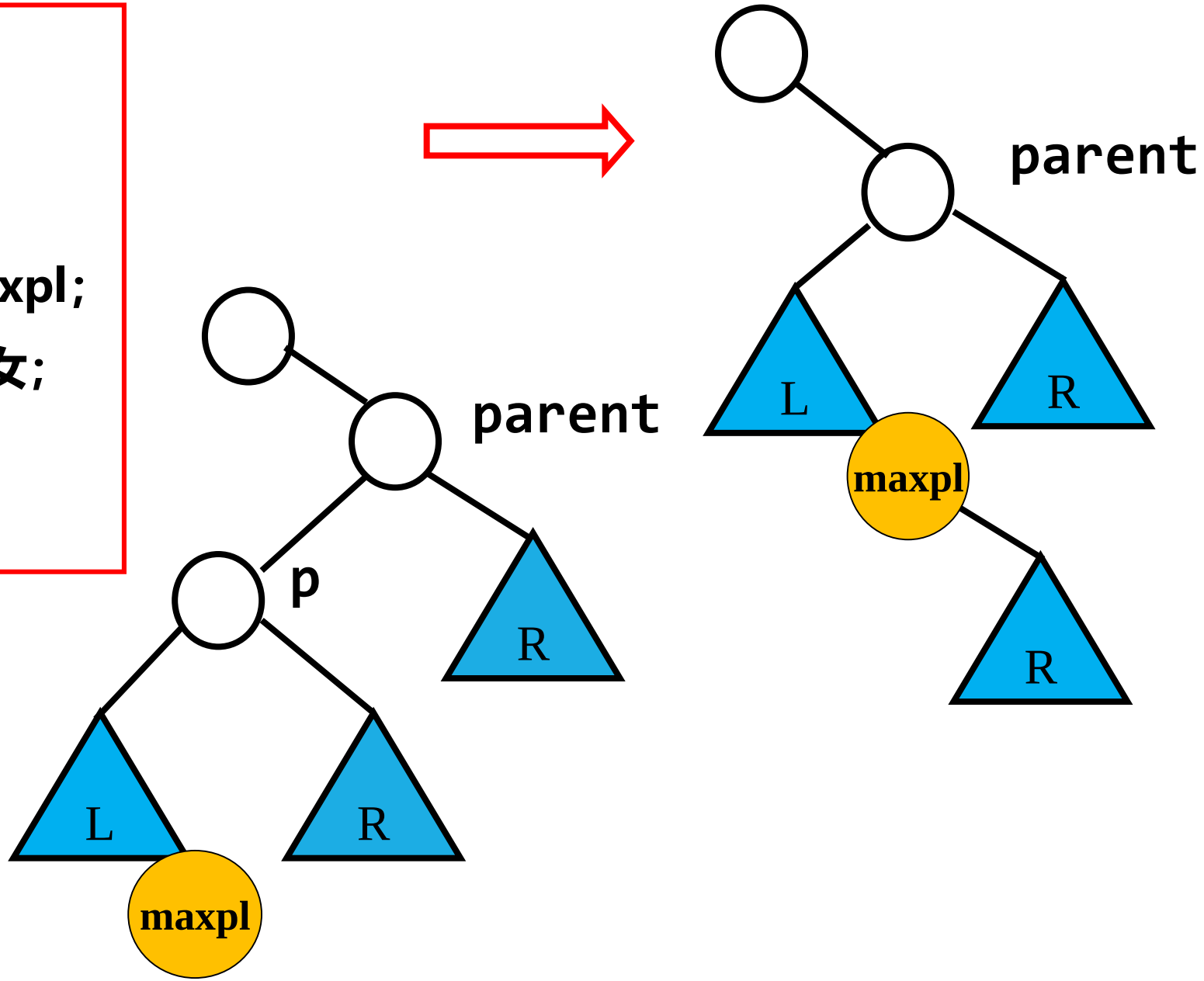
else {找p的左子树中最右下结点maxpl；

用maxpl的右指针指向p的右子女；

用p的左子女代替p；

}

具体见算法5-3

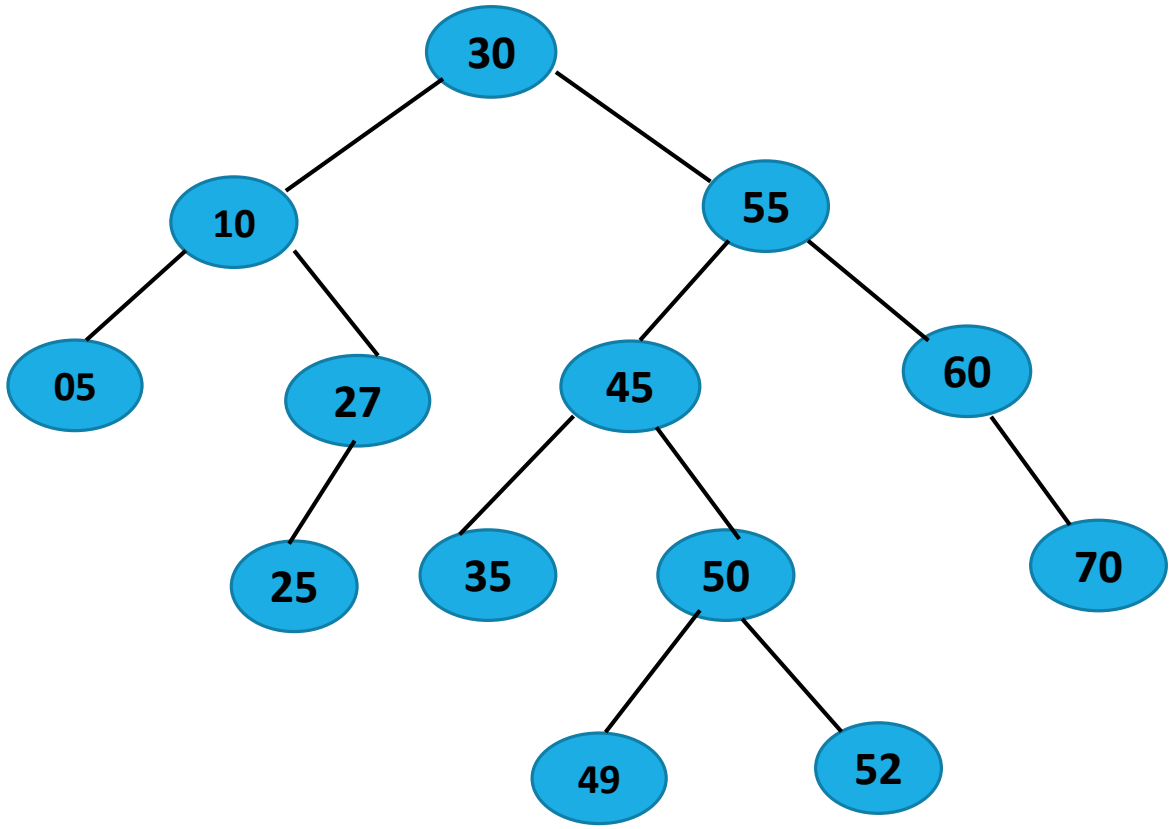


BST算法分析

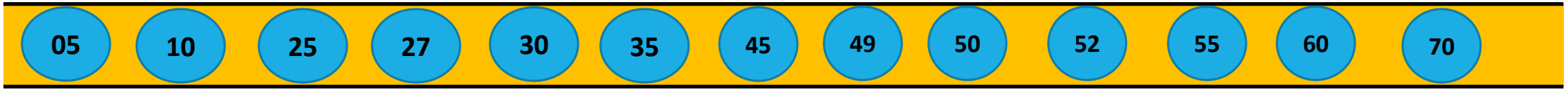
查找search

插入insert

删除delete

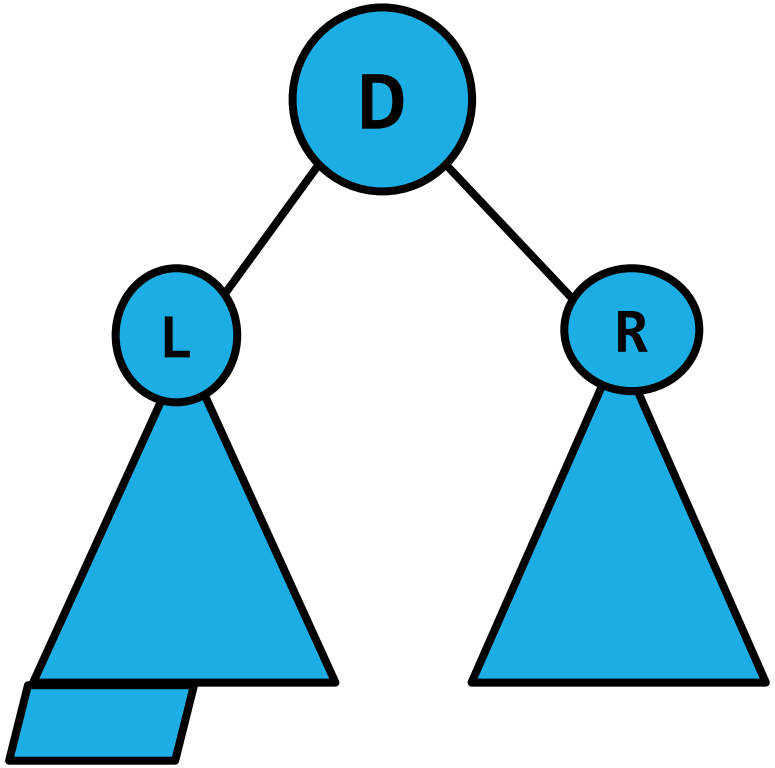


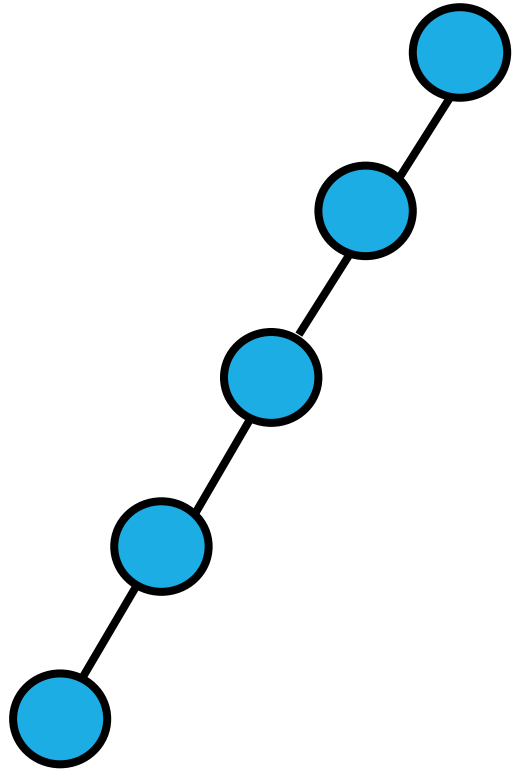
O(h)



BST算法分析

最好情况：完全二叉树 **O(logn)**





最坏情况：单支树 **O(n)**

BST算法分析

折中办法：相对平衡BST